VOTable Future

Mark Taylor (Bristol)

VO-ICE Tech Forum #3
CDS

9 May 2012

`$Id: votable.tex,v 1.4 2012/05/09 08:15:04 mbt Exp $`

Summary

- Issue(s): Null encoding
- Current status
- Decisions for Urbana
- Discussion: Euro-VO position

Issues

- Null encodings:
  - Integer columns: difficult (sometimes impossible) to encode NULLs (esp. streaming)
    - ▷ Need to choose a “magic” in-band value to represent NULL
  - Floating point columns: no distinction between NULL and NaN
  - Array columns: no distinction between NULL and empty array/array of empty elements
- Notes:
  - Issue is quite subtle
  - VOTable has survived for 8+ years without this causing (much) trouble
  - Inherited from FITS BINTABLE data model
  - Raised by Tom McGlynn, seconded by Markus Demleitner; prompted by use in TAP
  - VOTable is widely used - alter with care

VOTable DATA Encoding Refresher

- VOTable has three alternative data encoding mechanisms:
  - TABLEDATA *(widely used)*:
    ```
    <DATA>
      <TABLEDATA>
        <TR> <TD>M51</TD> <TD>202.43</TD> <TD>47.22</TD> </TR>
        <TR> <TD>M97</TD> <TD>168.63</TD> <TD>55.03</TD> </TR>
      </TABLEDATA>
    </DATA>
    ```
  - BINARY *(not much used)*:
    ```
    <DATA>
      <BINARY>
        <STREAM encoding="base64">
           TTUxAAAAAAAAAEBpTcKPXCj2QEecKPXCj1xNOTcAAAAAAAAAQGUUKPXCj1xAS4PX
           Cj1wpA==
        </STREAM>
      </BINARY>
    </DATA>
    ```
  - FITS *(hardly ever used?)*:
    ```
    <DATA>
      <FITS>
        <STREAM href="fcat-2.fits"/>
      </FITS>
    </DATA>
    ```
- These encode exactly the same data

VOTable Rules

Representation of "blank" values in VOTable columns:

- Varies by column data type:
  - ▷ Float scalars (`float`, `double`):
    - ◦ BINARY/FITS encoding: IEEE NaN bit pattern
    - ◦ TABLEDATA encoding: `<TD/>` or `<TD>NaN</TD>`
  - ▷ Integer scalars (`unsignedByte`, `short`, `int`, `long`):
    - ◦ nominated "magic" value (all encodings):

```
<FIELD datatype="short" name="COUNT">
  <VALUES null="-32768"/>
</FIELD>
```

    - ◦ Empty `<TD/>` not permitted! *(but often seen)*
  - ▷ Arrays (including `char[]` ≈ strings), complex, bit:
    - ◦ . . . are more complicated, but less important
- Summary:
  - ▷ No null/NaN/empty array distinction
  - ▷ Need to do work (choose non-data value) to write integer blanks
- Design motivation/benefits:
  - ▷ TABLEDATA ↔ BINARY ↔ FITS encoding transformations are lossless
  - ▷ All makes sense if you think in FORTRAN or FITS BINTABLE!

Current Status

- At Pune (Oct 2011):
  - Raised in TCG meeting:
    - ▷ decided to discuss in special Apps session
  - Special Apps session:
    - ▷ Not very wide participation (mostly Markus, Pat, me)
    - ▷ Made a provisional recommendation for VOTable 1.3 which solves most problems
- Since Pune:
  - Summary and call for comments on Interop mailing list
  - . . . no response
- At Urbana (May 2012):
  - Plenary $\frac{1}{2}$ session scheduled for discussion $\longrightarrow$ decisions

Pune Recommendation (1)

- Issues:
  - (1) Hard (sometimes impossible) to encode NULLs for integer columns (esp. streaming)
  - (2) No distinction between NULL and NaN for floating point columns
  - (3) No distinction between NULL and empty array for array columns
- Proposed Changes:
  - `TABLEDATA` integer columns: empty `<TD/>` element means NULL
    - ▷ Previously illegal, but commonly used with this meaning
    - ▷ Solves (1) for `TABLEDATA`
  - `TABLEDATA` floating point columns: empty `<TD/>` element means NULL
    - ▷ Previously meant NaN — subtle semantic change unlikely to cause problems
    - ▷ Solves (2) for `TABLEDATA`
  - New DATA encoding `BINARY2` — like `BINARY` but with per-cell bitmask marking NULLs
    - ▷ This is a new encoding
      — but for now only encountered by clients explicitly requesting it (from TAP)
    - ▷ Solves (1), (2), (3) for `BINARY` (at least, `BINARY`-like encoding)
- New status for known VOTable encodings:
  - `TABLEDATA`: All pressing issues resolved
  - `BINARY`: All issues resolved by using new `BINARY`-like format
  - `FITS`: is FITS – little motivation/opportunity to resolve issues, rarely used

Pune Recommendation (2)

- Related Change:
  - Add optional `serialization` parameter to VOTable MIME type (RFC 2046):
    - ▷ Example: `application/x-votable+xml; serialization=BINARY2`
    - ▷ Allows optional provision of new `BINARY2` encoding only if explicitly requested
    - ▷ Clarification of VOTable variant useful in some other contexts
    - ▷ Existing (unparameterised) declared MIME types still valid
- Other Changes:
  - We do *not* intend to revisit other areas of VOTable at this time

Rejected Suggestions

Other options were discussed:

- New TD attribute: `<TD null="true"/>`
- Variant empty element types: `<TD></TD>` $\neq$ `<TD/>` *(aargh!)*
- Magic bitmask column:
  `<FIELD name="__NULLCOLS__" datatype="bit" arraysize="ncol"/>`
- Do nothing

Decisions Required

- Null representations:
  - Agree partly/fully with Pune recommendation?
    - ▷ TABLEDATA changes (low impact)
    - ▷ BINARY2 changes (medium impact)
  - If not, what?
- Encourage/deprecate suggestions for other VOTable changes?
  - SKOS field attribute (Hervé)?
  - JSON (Thomas)?
  - . . . ?
- How to proceed if VOTable 1.3 is required:
  - Revive (currently dormant) VOTable WG?
  - Handle through Apps WG?
  - Something else?

Does Euro-VO have a position on these?

Discussion

- Advantages/Disadvantages of proposed changes
  - ☺ TABLEDATA changes make life easier for VOTable producers (esp. streaming)
  - ☺ TABLEDATA changes require very little code change
  - ☺ Becomes possible to represent RDBMS content more faithfully (esp. BINARY2)
  - ☹ BINARY2 requires updates to VOTable I/O toolkits
  - ☹ BINARY2 tables incomprehensible to old software
  - ☹ Some datatypes (arrays, bitmasks) still have no NULL representation in TABLEDATA
  - ☹ Equivalence between different VOTable encodings is lost
  - ☹ Translation between VOTable encodings becomes harder/impossible
  - ☹ New VOTable document version is required
- Considerations
  - What is VOTable for? *(Delivering data to user code? DB$\leftrightarrow$DB communication?)*
  - Who will benefit from the proposed changes?
  - Who will be inconvenienced by the proposed changes?
- Other opinions?

If you ask me...

- TABLEDATA changes:
  - Low impact, little implementation effort required
  - In effect just blesses current common practice
  - Significant benefits for streamed output producers
  - $\longrightarrow$ worth adopting
- BINARY2 changes:
  - Medium impact, requires effort from I/O toolkit developers
  - Complicates VOTable landscape
  - Will it be widely used? (is BINARY much used now?)
  - It is necessary to faithfully represent RDBMS tables
  - . . . but it's not clear (to me) that it solves actual practical problems
  - $\longrightarrow$ is case for adoption compelling?