

Re: Wanted:MAIL.MAI structure definition

Source: <http://unix.derkeiler.com/Newsgroups/comp.os.vms/2006-06/msg00443.html>

- *From:* Dave Froble <davef@xxxxxxxxxxxxxx>
 - *Date:* Thu, 08 Jun 2006 11:47:51 -0400
-

Hoff Hoffman wrote:

Dave Froble wrote:

JF Mezei wrote:

Hoff Hoffman wrote:

More efficient, and less flexible -- it's a trade-off. There are serious limits within the current design, such as the inability to generally search the mail file, or to have a message in two folders, or to handle the information in a transactional format, etc.

Funny, ALLIN1 does all of this and it uses ISAM files for in indexes.

Yeah, through the years we did lots of things. We also learned some lessons. New ideas and products grew out of such.

Bottom line, a relational database is more flexible in the retrieval of data. Note that I'm not a big fan of relational databases as a cure-all for all purposes. But for retrieval and searching, they're pretty good.

I have to assume that there is a lack of familiarity with relational databases here.

Relatively, very possible.

Relational databases are very powerful, and very easy to add new data and new tables, and allow the programmer to provide the end user great flexibility -- in terms of data

Re: Wanted:MAIL.MAI structure definition

organization, cross linkages, action routines, transactional integrity, etc.

That's sort of what I thought I was trying to write.

However, relational databases have significantly higher overhead than, for example, something like RMS. Perhaps this impact has been softened as hardware has gotten faster, but, overhead is overhead, and if you weren't using the advances in hardware for overhead, those designs that have less overhead would also gain from the faster hardware.

Could you do this with RMS? Ayup. But by the time you're done, you are maintaining your own semi-relational database

Goes back to what I wrote about having learned some lessons through the years. One would have a real hard time convincing me that for any general capability that custom code is better, or even as good, as a reusable tool designed for that capability. We progressed from addressing cylinder and track on disk to file systems and then to database systems. Lessons learned and applied.

— and there are features of database products that your upgraded RMS would still lack — and then you get to maintain your database, support and upgrade it, and all the effort that entails?

Yep! In most cases why do so, when it's not a core part of what you're trying to do? Use the tools appropriate to the job.

The problem with mail these days is that RFC822 headers are quite variable and new fields are added and not always consistently used. And one really needs to keep that header "intact" because it is like a postmark on a letter. So even if you were to parse the RFC822 header into some fancy XML structure, you'd still need to retain the original RFC822 header because your XML parser wouldn't be able to understand new fields being added to RFC822.

Wanna bet?

I have to assume that there is a lack of familiarity with XML here.

Re: Wanted:MAIL.MAI structure definition

Again, possibly so, but I can envision some designs that could 'learn' about previously unknown types of data and make some reasonable guesses for adding such without need for re-programming. Definitely not a 100% solution, but could handle minor variations.

Remember, those things that 'cannot be done' remain so only until the first time they are 'done'.

Looking past the hype (not an easy task :-), XML is very powerful and very portable, and it gets the application out of the business of parsing the data. (There are trade-offs here too, as the XML libraries have substantial overhead. It doesn't scale as well as a more traditional database.)

In OpenVMS terms, XML is a portable text-based itemlist-like construct -- and one that can be nested, provided with attributes and tags, structurally verified, displayed, embedded, transferred, and extended as required. I'm using XML within OpenVMS V8.3, and for all of these reasons.

I would also propose storing the data in a database, and allowing the external software to access the data via XML; to allow data imports and exports using XML. This gets the other software out of the business of processing RFC-compliant headers. (And, going full circle, if SMTP mail were to be (re)implemented today, it is exceedingly likely that XML would have been used as the basis of the wrappers.)

--

David Froble Tel: 724-529-0450
Dave Froble Enterprises, Inc. E-Mail: davef@xxxxxxxxxxxxxx
DFE Ultralights, Inc.
170 Grimplin Road
Vanderbilt, PA 15486

.