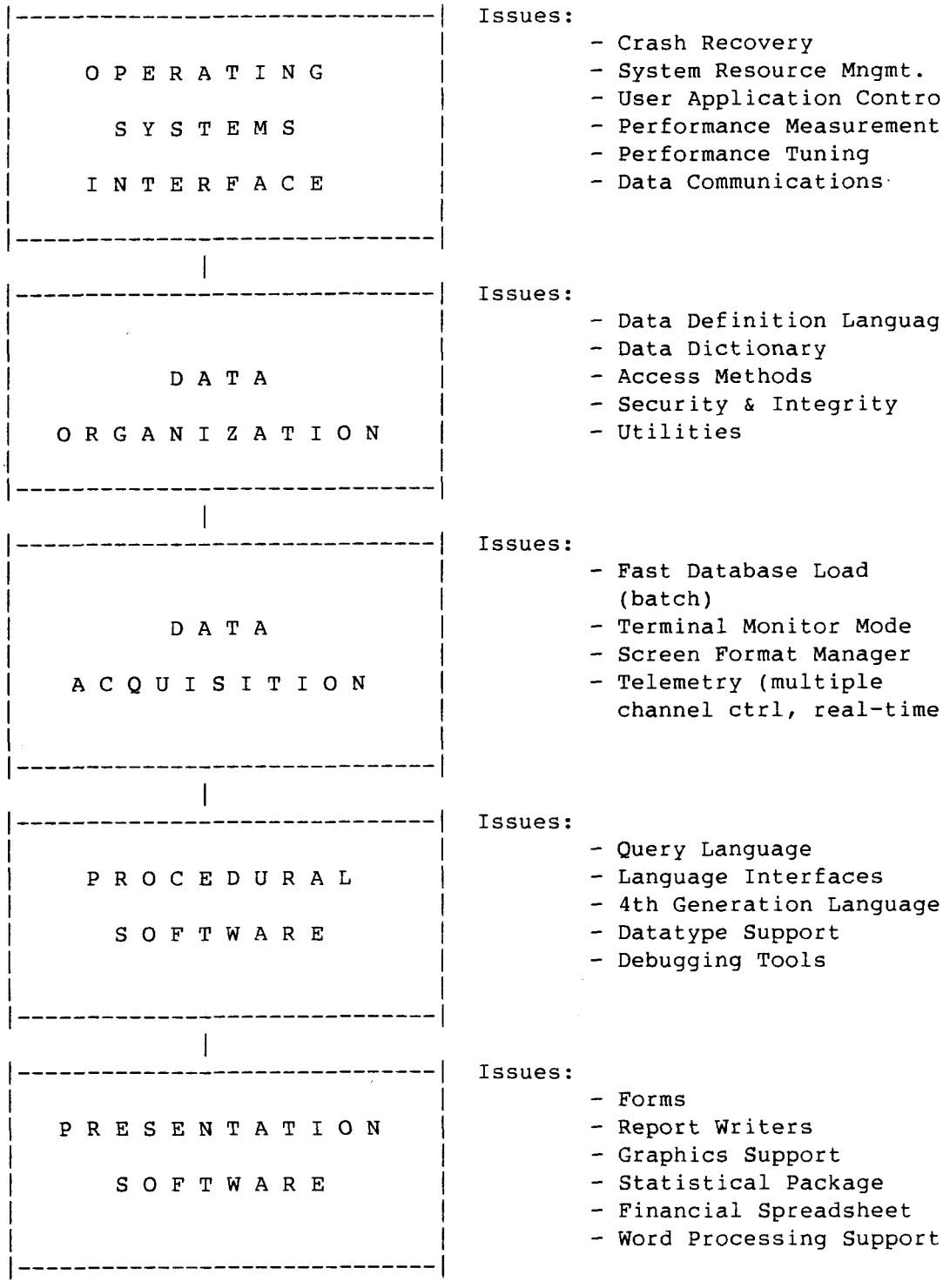
From: MIKEB 2-NOV-1984 15:30
To: GARY
Subj: Overview of Application Development Issues re: Ingres

Marty,

Here is, as promised, my list of needed Ingres enhancements. I have tried to classify all enhancement and/or extensions according to their proper place in one of the functional areas traditionally recognized when one speaks of a DBMS

DEFINITION OF FUNCTIONAL AREAS



ENHANCEMENTS BY FUNCTIONAL AREA
---

1. Operating Systems Interface

1.1 Crash Recovery

Automatic crash recovery is not presently supported. This feature is a "must" for a trouble free applications environment. Currently the system has no notion of application "run-unit" and its status at the time of a crash. Although Multi-Statement Transactions are supported, it is not enough to control standard applications which must deal with "tasks" or "jobs", rather than just sets of queries. The issue of whether a job has gone to completion and need not be restarted vs. the question of whether a given transaction has been sucessfully processed represents a step up from concurrency control to the kind of functionality normally found in a teleprocessing monitor.

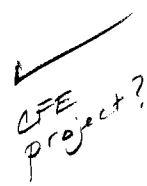
At the moment TP monitor functions have to be emulated within each individual application program and no communication between separate programs running against the same database is possible.

Since nothing is running in a supervisory state on top of a group of applications active against the database, it is impossible to know the exact impact of the database becoming inconsistent on any individual task. Presently when the database becomes inconsistent, we just issue a system level message for everybody to exit out of ingres and then run "restoredb" to bring the database back to life. Where it is restored to with respect to previously active tasks is currently a matter of guesswork.

1.2 System Resource Management

Ingres does not have the ability to allocate system resources dynamically to guarantee acceptable levels of application performance under all conditions. Control of system priority levels, fixing memory pages for specific often performed tasks, job scheduling with maximum load levels, etc. are all application oriented features which would be nice to have in a production environment.

Re-entrancy is also a big issue since at the moment no guidelines exist for writing re-entrant application code using the ingres front-ends. The front-ends themselves are not re-entrant and, as I understand it, separate executable images are activated each time a different user calls on a particular front-end. Since at last count the report writer, for instance, required over 512K of memory to run, multiplying that factor by the possible number of users shows the kind of overhead one can create.



It must be pointed out that re-entrant applications imply the existence of a TP monitor to manage buffer allocation and the database interface for each user who signs on.

The current approach leads to a squandering of system resources in terms of memory usage and as a corollary, to a loss of performanc , since additional image activations cause higher levels of operatin system overhead.

1.3 User Application Control

We are again talking about TP monitor facilities tied to a central version of Ingres serving a number of application "run-units" in a transactional environment.

A catalogue of re-entrant applications available at a given terminal would be displayed on-line and users could select the one they wish to run. Hierarchical password control would be provided and user or application controlled usage modes (shared, exclusive, etc.) along with applicable journaling schemes would be supported.

The way things stand right now, the journaling function is either enabled at database creation time and will remain in effect for the life of the database (if I understand this correctly), or it is not set and cannot be invoked at a later time.

Not true!

I'm assuming that the "server" version of Ingres will handle such functions, but in what release and how thoroughly?

1.4 Performance Measurement

For applications to be viable in production mode performance must not only be adequate but also predictable. Tools must exist which allow programmers to measure the behavior of an application as the number of concurrent users varies and the volume of transactions fluctuates.

A version of Ingres with embedded performance measurement tools would be a plus. Such tools could be selectively activated using a "toggle" approach and should be able to give statistics on all phases of query execution (CPU time through each phase, i/o stats, memory allocation stats, etc.) or selected front-end activity.

1.5 Performance Tuning

At the moment we don't have a way to model an application, trying different types of access methods, indices and key structures. Such a model would be useful for application tuning and could be used on an ongoing basis to monitor production applications. This model would naturally have to be tied to performance measurement tools for performance feedback.

Performance tuning with respect to VMS system quotas is also a problem. How should various quotas be set to mirror application requirements(?). Multi-Statement Transactions, for instance, require minimum queue block allocations which may exceed default standards. If the quota is not properly set, and this is application dependent,

Needs to be fixed. (?)

a failure may result in the lock escalation process.

1.6 Data Communications

Since Distributed Ingres is on the schedule, it would be productive to review its application features which I have frankly never seen.

Among issues of interest (at least to me) would be:

- naming & addressing resources
- routing
- message processor features
- transaction center functionality

2. Data Organization

2.1 Data Definition Language

Support is needed for additional data types, namely:

- binary
- packed decimal
- extended date (short, julian, european, etc.)
- money (with associated edit mask + currency conversion)
- case control

The ability to add or delete a column to or from a table would be a definite plus. Combining several columns into one or breaking up a column into separate columns, when feasible, would also be a useful feature (text or character data types).

Definition of permits is done at the table level. It would be nice to have a more general facility which would allow the DBA to grant permissions based on classes. Permission could then be granted to a class of users, defined by a user list, to perform certain operations on a given class of tables defined by a table list. This would provide a much needed shorthand way of granting permissions.

2.2 Data Dictionary

Mike needs to talk with Peter K. (Joe & Paul).

As I understand it, there is a data dictionary project on the schedule, but I haven't seem anything regarding its projected features. It is therefore difficult to say whether it will be adequate or not for application needs (?)

2.3 Access Methods

We have a lot of useful things at the moment, but down the road it would also be nice to do well some of the things which hierarchical or network databases currently do:

- linked & doubly linked lists with owners (ordered relations)
- GET PRIOR / GET NEXT within context (portals)

- tree extraction (nested queries)
- time series extraction (telemetry, statistical analysis)

Once users and application programmers are given a choice between several different access methods, modeling tools and performance measurement become a necessity since we must be certain that the best access method has been selected to solve a problem at hand.

2.4 Security & Integrity

A few features are still missing:

- encryption/decryption of data (as an option)
- an independent security control mechanism which would provide protection for different levels within a logical data structure A logical data structure being a hierarchical collection of classes each made up of a set of tables.
- audit trail for security control mechanism
- assertions (data integrity)
- triggers

2.5 Utilities

Our current utilities are adequate.

3. Data Acquisition

3.1 Fast Database Load (batch)

I don't believe that we have a fast batch load facility within Ingres.

I know that one was once written (I believe it was for Fairchild) but it was never made part of the product. This would be a useful feature for a lot of obvious reasons.

3.2 Terminal Monitor Mode

The terminal monitor mode is a most unlikely candidate for data acquisition as things are set up right now. This mode is useful to create tables and set up securities, but is just not adequate for data entry. Of course that is the reason why we have QBF and MQBF, where data can be entered in screen form. The only problem is that tables cannot be created in QBF or MQBF. That, in turn, implies that one must first go into terminal mode to create the table, then exit and enter QBF to do data entry, which is not too convenient when dealing with ad hoc applications.

One solution would be to have some sort of prompting facility (such as the one in Ashton Tate's Database II/III, or QBF's default frame) included in the terminal monitor.

The relational shell is an attempt at solving some of those obvious

problems (i.e. integrating semantically "pure" sub-systems into something useful and practical) and giving the user a single environment where all Ingres features are available to him in a common form (i.e. so that he does not have to learn 3 different subsystems with radically different command forms).

A big question remains in my mind as to whether the relational shell is a practical solution or just a bandaid on a wooden leg.

Personally, I think that the only way to solve this problem is to go to an extended QUEL or SQL, still in interpreter form, which would allow procedures and workspace variables and which could provide screen formatting, reporting and graphics capabilties in the form of primitives. Default procedures for prompting and display (such as those used right now) would automatically be used if a user procedure was not specified.

Yes, this gets us into unpopular topics of conversation, but I must stress again that from an application's development standpoint practicality and economy of systems resources is still a major plus.

3.3 Screen Format Manager

VIFRED could use the following enhancements:

- data type support
- edit masks on both input and output
- "pop-up" form
- ability to box groups of fields or trim
- program control of video attributes
- trim video attributes
- choice of prompts (using edit mask, toggle between "yes" and "no")
- multiple sub-menu lines
- sub-menus should use serial selection with reverse video toggle to highlight selected keyword (line LOTUS 1-2-3), that's simpler than typing a keyword and there is no possibility of an ambiguous entry
- more message options: position on row other than row 24, column 1; sleep till carriage return, etc.
- no echo mode for selected fields
- lock keyboard command with status message (useful when validating field)
- table fields scroll up and down are not symetrical with respect to ACTIVATE COLUMN statement. One should be able to ACTIVATE on both entering and leaving a field (or column). Right now activation occurs only upon leaving.
- automatic scrolling of table fields one screen at a time. There is no reason why the user program should have to provide that facility when EQUEL/FORMS knows the # of rows and the size of the table field and can therefore easily support an option for screen scrolls.
- scrolling a table field one line at a time causes all lines appearing on the screen to be rewritten. This is time consuming and not very elegant. Hardware options such as windowing and scroll control should be used on terminals that have them.

3.4 Telemetry

Clearly we have nothing in this area at the moment. I can think of a lot of applications where the ability to acquire and evaluate data in real-time based on input from multiple channels is more than just useful (point of sale, quality control, robotics & factory automation, artificial intelligence, etc.).

This kind of data is captured in "time series" format, normally represented as a variable size record with a header describing what is being observed, the frequency and number of oberservations already performed, and the maximum number of observations allowed. The rest of the record contains actual observations.

I don't understand this proposal.

Time series data is well suited to rapid analysis by statistical or heuristic models and is a productive way of handling data in real-time decision oriented systems (expert systems).

In the relational environment, we have the basic data structure needed to store this kind of data (a table can be viewed as a collection of time series where columns are objects being observed and rows the corresponding individual observations). Unfortunately, different columns will grow at different rates since frequencies of observation may differ and no provision is made for headers.

A new table type could solve this problem. Joins would still work, but would be conditioned by observation frequencies on different columns.

4. Procedural Software

4.1 Query Language

Heresy. All this belongs in ABE (4GL).

QUEL or SQL, I feel that we still have to extend the query language to support procedural constructs, workspace variables, formatting for input or output (with defaults) and interfaces to the standard file system.

"Nested Queries" is an important extension to QUEL which, since it is on the schedule, will undoubtedly raise questions about some of these issues.

Clearly, once you allow nested queries, there must be procedural statements within the main query loop which select a particular nested query based on variable values for a row of data just retrieved. Extensions to support workspace variables are needed to allow attribute values resulting from different levels of retrieval to be manipulated and/or compared.

4.2 Language Interfaces

We are talking here about the various EQUEL(s) (EQUEL/C, EQUEL/FORTRAN, EQUEL/COBOL, etc.).

My very personal feeling on this is that if we had a complete, extended query language we would need none of these. From my programming experience with EQUEL/C and the financial system, I would say that approximately 70% of my program statements are EQUEL statements (database retrieval or update, forms control, concurrency control, etc.) and the rest are pure "C".

I don't see why we have to maintain this incredible array of interfaces when in fact, in each case, very little of the actual host language is used.

The only things missing in EQUEL to make it a full-fledged language are procedural constructs, variable declarations, interface to the native file system and an environment for calls and inter-program communication. I feel that it is easier to put these in and end up with a full, useful and self contained 4-th generation language than try to be all things to all people and end up with a fragmented set of front-ends none of which can address a real world problem effectively.

We have already talked about some of the problems of extending the QUEL/EQUEL syntax in order to handle nested queries. If OSL's language constructs look unfamiliar and unappealing to you now, you should wait until they've been extended to incorporate nested queries. There'll be nothing but nested braces on the page...

To get back to the obvious objections from "C", FORTRAN, and COBOL fans who would have us believe that they really need their original languages, user exits from EQUEL to the languages and the ability to call EQUEL procedures from any of the higher level languages would answer all needs. Data could be transmitted either through calling sequences or via external variables.

4.3 Fourth Generation Language

I guess this must be ABF/OSL.

I'm not familiar enough with OSL (and ABF for that matter) to offer a full blown critique, but here are some of the things for which I've heard it criticized:

- no facility for source code maintenance
- no interface to standard file system
- no debugger
- no repeat queries
- no global variables
- no return values
- only single master detail relationship possible
- not exactly user friendly

Mike should attend an ABF course!

4.4 Datatype Support

This goes back to what I was saying earlier about variable declarations. If we were to implement all the datatypes on the current request list, we would most certainly run into trouble with some of our EQUEL(s) since some of the languages don't support those datatypes. "C" for instance does not support binary or packed decimal datatypes. FORTRAN may or may not support packed decimal, depending on whose FORTRAN it is. Date and money datatypes, with masks, could only be fully implemented in COBOL in native mode. They'd have to be faked by subroutines in other languages.

Good point!

How do we solve this problem?

Don't know.

4.5 Debugging Tools

It would be safe to say that at the moment we have none. That is none that are available to the general public.

R & D has some "set trace" type options which cause Ingres to dynamically display query information, but these are generally a closely guarded secret.

5. Presentation Software

5.1 Forms

This is RBF, QBF+ and GBF.

Here are some of the features that are missing:

RBF:

- ability to edit page header and footer
- ability to define query (w/o view)
- breakpoint control for user
- "file report" to produce simplified Report Writer code
- generate wide reports through horizontal scroll
- larger maximum report width

QBF:

- add "where" clause to query mechanism
- add "sort" capability
- ability to scroll backwards
- obtain record counts for retrieval
- edit masks for numeric and special character display

GBF:

- will only generate one graph per query
  (query must be written so that only one graph is produced)
  it would be nice to have a more general graphing facility
  which could draw a graph for each break on a specified
  key during data retrieval, automatically changing headings,
  titles and scaling.

- cannot handle dates
- text cannot be added
- no color control

5.2 Report Writers

I'm using the plural form on purpose. There is no reason why a database management package should not have more than one report writer. Optimally, one report writer such as our current report writer could be used for quick tabulations and another more sophisticated, language based report writer would be used for complex reports.

Criticism of our current report writer would have to include the following items:

- not syntax based (very difficult to extend functionality)
- interpreter with finite state machine architecture = large memory requirements
- can only handle single queries
- cannot call another report from within a report
- cannot perform multiple iterations over extracted data
- no cross-tabulation facility
- cannot have multiple detail sections

5.3 Graphics Support

Graphics support should be available beyond the scope of a graphics package such as GBF. Users may want to build custom graphs using language based graphics primitives and an assortment of exotic devices which they would preferably want to control.

The following questions are often raised:

- does the database system implement the actual storage and manupulation of graphics?
- is there any facility to input graphics in graphics form into the system?
- what interactive tools are available besides the keyboard? (light pen, tablet, mouse, stick)
- does the graphics capability extend to the handling of digitized images?
- are there recognition capabilities? (character or pattern recognition)
- what sort of line graphics images are available to the user? (move & draw, mark & polymark, polyline, polygon, rectangle, circle, sector, analytical geometry curves specified by equations, text)
- what are the primitives of the language interface?
- are clipping and other image manipulations available?
- are color and shading supported?
- what about digitized image compression?

5.4 Statistical Package

We don't have one of those interfaced to the system yet. That kind of capability would be very useful particularly in support of financial and other models.

When looking at a package of this kind the following questions ought to be asked:

- how many variables are allowed?
- what is the maximum number of observations?
- extrema?
- means, percentiles, standard deviation?
- correlation?
- linear regression analysis?
- multiple linear regression?
- stepwise multiple linear regression?
- polynomial regression analysis?
- factor analysis?
- discriminant analysis?

- analysis of variance?
- tests of estimate? (Chi-square, significance)
- time variable statistics?
- time series analysis?
- creation of distributions? (linear, exponential, power, hyperbolic, binomial, others)
- exponential smoothing and forecasting?
- interface to external files?

5.5 Financial Spreadsheet

There has been long standing talk of us doing something in that area but I haven't seen any specs or schedules.

I don't believe that I need to sell you on the virtues of a spreadsheet and the wonders it could accomplish as an available addition to the financial system.

The ability to extract financial data from the data base into a spreadsheet package would greatly enhance forecasting and budgeting activities.

5.6 Word Processing Support

I don't recall there ever being any talk about this one, but I think it is something to consider seriously given the sorry state of word processing in both the UNIX and VMS world. Besides what else would we do with our text datatype...

Obviously, these are not all 4.0 enhancements. I haven't ranked anything in order of priority on purpose so that a total picture of what remains to be done could be better conveyed.

If I only had nested queries, edit masks in forms, some improved graphics capability and good performance I could survive writing applications for a while. The rest would be nice...

Mike.