Inter-Office Memorandum

To	Interest List	Date	April 2, 1982
From	Rick Cattell	Location	Palo Alto
Subject	Database Application Goals for calendar year 1982	Organization	CSL

XEROX

Filed on: [Indigo]<Squirrel>Doc>DBApplications.memo, .press

Introduction

This is a summary of my plans for the Cedar database system and its applications this year, to organize my thoughts and to get some feedback and ideas.

My short-term goal is to test the functionality of CedarDB and to demonstrate it to the CSL audience. The underlying goal is to generate enough interest to recruit manpower for other database applications and, more importantly, manpower for basic database work that is needed to adequately support database applications. My long-term goal is interesting database reasearch in CSL.

The vehicle we chose for an initial application is the message handling system Walnut. I will briefly describe the design of Walnut, as an example of an application design and to illustrate application needs. After that we will discuss Cedar database applications in general.

Walnut phase one

Let's call Walnut 1.0 the version of Walnut that Willie-Sue Haugeland and I want to have generally usable by June. Walnut 1.0 provides four kinds of Cedar Viewers visible to the user:

1. *Control window*: This window is the one that appears when Walnut is started. It provides menu commands to fetch new mail, create a new message form (item 4 below) and create a new "mail file" in the Laurel sense (item 2 below, which we call a *message set*). The control window also reports when new mail is available, the user's password is invalid, or any error is recognized by the Walnut system.

2. *Message-set display window*: This window roughly corresponds to the top window of the three in Laurel. It displays a list of messages, their senders, dates, and subjects. Like Laurel, there is a special active message set for each user into which new mail is initially inserted. This viewer for this message set is automatically put up, along with the Walnut control window, when Walnut is started. Unlike Laurel, one may have many message sets on the screen at the same time. Pointing at one of the messages and red-buttoning produces an instance of item 3 below, a message displayer. A menu of commands at the top allow adding and deleting messages from the message set, as well as the standard viewers commands (Grow, Destroy, etc.).

3. *Message display window:* This window roughly corresponds to the middle window in Laurel. It displays a message in the database. The menu of commands at the top include Answer and Forward, which produce an appropriately initialized instance of a message form (item 4 below), DeleteMsg, which deletes the message itself and removes it from all message sets, and the standard viewers commands. The message display window, like

Laurel, does not allow editing. Unlike Laurel, the user can point and red-button a message field to see a database item named there. For example, selecting the "In-Reply-To" field causes the message to which this replied to be displayed (another message display window). Selecting the "Subject" or "Message-Set" items causes the corresponding subjects or message sets to be displayed (which enumerate messages on that subject or in that message set, which can in turn be selected). Selecting persons in the "To", "From", and "cc" fields causes the person to be displayed (phone number, office number, picture, or whatever the database happens to contain). I will discuss this "point to see it" idea in the next section, on the Squirrel system.

4. *Message editor window:* This window roughly corresponds to the bottom window in Laurel. It displays a message form which the user can edit with Tioga. Some of the fields will be initialized according to whether the message editor window was created by the "Answer", "Forward", or "New Form" commands. The message editor window was designed to be almost independent of the rest of Walnut, so that we could make a stand-alone tool for sending messages. Its commands include "Send" and "File"; the latter simply puts the message into the database without sending it, e.g. for the user's messages to himself (this feature is disabled for the stand-alone version).

Walnut also includes a mechanism for logging all user operations on messages and message sets, and replaying the log if some hardware or software failure destroys the database. This feature is primarily necessary due to shortcomings of CedarDB:

1. We have no reason to believe CedarDB will be reliable enough to entrust with one's mail on a long-term basis. For that matter, we have no reason to believe that Walnut or the rest of Cedar will be reliable enough for routine mail handling, either, since the components are experimental and not always protected from one another.

2. CedarDB does not currently deal efficiently with very large character strings, such as one might encounter in the bodies of messages. We therefore store pointers to message bodies in the log file rather than the message bodies themselves in the database.

When one or both of these shortcomings is remedied, the need for the log mechanism will probably disappear.

The Walnut log replay program, due to clever design of the log format to be upward compatible with Laurel mail file format, can also double as a mechanism to read Laurel mail files into a database.

Unfortunately, the log has its drawbacks: it will grow without bound as the user reads messages, even if they have been deleted. This is particularly a problem when users delete an order of magnitude more messages than they keep, as the old messages still exist in the log. A dump program is necessary to periodically dump the database into the log, discarding the deleted messages. We may also find it necessary to implement a special optimization, that truncates the log file after every user session.

There are other difficulties in implementing Walnut. Two major ones are a result of its implementation as multiple processes:

1. Portions of Walnut may asynchronously call the CedarDB, CedarGV, and Viewers packages. In some cases it is not clear where monitors should best be inserted to avoid anomolous behavior. I want to be able to run more than one database application at a time, which suggests that the DBView interface to CedarDB will have to be the synchronization point. Although for independent reasons we are making *most* database access in Walnut go through a single interface where synchronization could be done, it appears to be about as hard to do synchronization there as DBView. Unless a better solution comes along, I therefore propose to do the synchronization in DBView.

2. When one component of Walnut modifies the database, e.g. a message is removed from a message set, other component's displays may be rendered invalid. This problem does not arise in Laurel since only one window of each type is allowed. It looks as though any solution to this problem will eventually be forced to handle the general case of notifying all database windows of the update. I therefore propose to add this notification facility to Squirrel, which already keeps a list of application viewers on the database.

Walnut phase one is partially implemented at the time of this writing. We expect to have all of the basic features implemented and reliable enough for wide use (but no wider than Cedar!) by June.

We have deferred our less modest aspirations to Walnut phase two, which is described later.

Squirrel phase one

Squirrel is a package to make database applications easier to build, and to make multiple database applications act in an integrated way from the user's point of view.

The development of Squirrel has been broken into two phases: phase one, largely completed in January 1982, which provides the basic facilities crucial to the development of Walnut; and phase two, to which we deferred functions less important to getting Walnut phase one to a minimally usable state.

Squirrel provides integration of database applications in two ways:

1. CedarDB currently allows only one database open at a time. Squirrel opens the database, and notifies application programs sharing the database when it is closed or re-opened. Also, application programs are notified when a change is made to the database, to allow them to update their data structures or screen display.

3. Squirrel makes it easy for applications to follow a uniform user interface paradigm, in which the user sees one Cedar Viewer on the screen per database entity, an object-oriented menu of commands upon a database entity, and a consistent interpretation of user selections on the screen.

Squirrel recognizes three primary types of application viewers: *displayers*, *editors*, and *queryers*. A *displayer* displays a database entity, provides a menu of commands upon it, and shows the information the database contains about the entity. For example, for a message displayer, the message's header and body would be displayed, along with commands such as Answer, Forward, and Delete. An *editor* looks much like a displayer, but allows the user to modify the information about the entity, and usually has a different set of commands. For example, a message editor is created in response to a NewForm or Answer command; the user fills in the fields and invokes the message editor's Send or File command. A *queryer* also looks like a displayer and editor, but the user may fill in the fields with values, boolean expressions of values, or other application-interpreted information; the queryer represents all entities in its domain which satisfy the constraints. When the user invokes the query, the entities satisfying it are displayed and may be browsed or printed.

A Squirrel window on the screen provides the user application-independent database functions, such as commiting or aborting a transaction, dumping or loading the database, or erasing portions of the database. The Squirrel window also allows the user to explicit invoke a displayer, editor, or queryer on a particular entity type (a *domain*). If no application has registered itself for the given domain, Squirrel provides a default application-independent displayer, editor, and queryer, the Palm browser/query system. Squirrel and Palm have proven indispensible in debugging application programs, as there is no other simple way to examine and modify the database a program is using. The Palm facilities have also completely obviated the need for some application programs. For example, for some personal databases (a wine database, and a database of phone numbers and addresses) we have found the Palm tools not only adequate but quite useful.

In summary, Squirrel provides the application program:

1. a library of handy procedures to set up Viewers, particularly those that fit the standard paradigm,

2. a library of handy procedures for database access, again particularly for using the standardized paradigm,

3. a central registration mechanism for coordinating opening and closing of the database, database updates, and calls from one application to another dealing with an entity the former has no mechanism to deal with, and

4. application-independent tools for debugging, dumping, loading, editing, and viewing databases.

Cedar DBMS

The CedarDB system of 1Q81 had proven somewhat lacking in two aspects: (1) the performance of the underlying file system, Juniper, and (2) the machinery each application needed on top of CedarDB (to do adequate integrity checking and their routine data operations). Alpine running under CedarDB is designed to remedy the first problem; the DBView level running on top of CedarDB is designed to remedy the second.

The development of the DBView interface to CedarDB has been broken into two phases in a fashion similar to Squirrel. The first phase, which includes all of the design save two more complex features not immediately necessary to Walnut, was completed in January 1982. It provides integrity checking of the data in a variety of ways, facilities to make it feasible for more than one application to share a single database, and some more powerful operations than present in the old DBTuples interface.

The main two features deferred to phase two are *views* and *environments*. Views make application sharing of databases easier by allowing procedurally defined relations, and also allow encoding of operations on entities as procedures stored in the database. Environments allow databases that contain both public and private portions to appear as a single database to the owner of a private portion. Environments will also provide a simple form of "layers" in the PIE sense, to encapsulate updates to a database in a useful form. Environments will necessitate changes below the view level of CedarDB, to allow database segments to be logically independent files, as well as changes to the DBView implementation. Views only require changes to the DBView implementation. There exists an implementation of environments that allow public/private databases and requires only DBView changes, creating two instances of the lower level database code; I may consider this if the pressure for combined public/private databases continues to increase.

The lack of environments is a serious problem right now. There is no way to open both a public database (e.g., whitepages) and a private database (e.g., user mail) at the same time and have the two interact in a reasonable way. We can only copy whitepages into private mail databases or be willing to store our mail in public (unprotected!) databases. The lack of multiple transactions on the same machine is similarly a serious problem with respect to concurrent database access by multiple applications. Squirrel can notify applications of database updates or transaction open/close, or do some user interface coordination between applications, but there’s no way we can do much more than experiment with really sharing databases between applications until many man-months have gone into these problems and into Alpine. I therefore view Walnut and other applications as interim experiments; I’m hopeful that these experiments will help rather than compound the problem by bringing in more motivated manpower. The DBView interface will make it a little easier to retrofit applications to these extensions than the old DBTuples interface.

Some other shortcomings of CedarDB we are coding around in application programs for now (e.g. via the log mechanism in Walnut). I hope to reduce the need for these short-term measures in the second phase of CedarDB development.

The motivation and design of changes to CedarDB are discussed in [Indigo]<CedarDocs>Database> ViewLevelDesign.press, readable but not yet completed at the time of this writing, so no more details of the DBView interface will be given here.

Walnut phase two

A variety of facilities more powerful than Laurel are made possible by the storage of mail in a database. They are not essential to the use of the system, so are deferred to a second phase. Two facilities that come immediately to mind are:

1. *Message Queryer*: A query-by-example-like viewer which lets a user fill in fields and produces a message set of messages satisfying the query. This tool can be built at various levels of aspirations, allowing the fields to be boolean expressions or even nested viewers which describe the entities desired recursively as queries. The simplest form of this tool can be implemented directly with the primitive DBView DomainSubset operation, so this is a simple task. The properties of querying message entities are little different from querying other application entities, so a variant of this code can probably be copied into Squirrel for use by other applications.

2. *Mail Sorting:* There are two times when the user might desire to sort mail on some criteria. One is on receipt, when the user might like to see messages sent directly to himself before those sent to distribution lists (unless they are from his boss). Or something like that. This might better be implemented by simply routing selected messages to a special message set representing higher-priority mail. The other case of sorting is on old mail, perhaps to order a message set in some nice way or to produce nice output for printing.

The features of phase two will largely be a result of feedback from phase one, so I will say no more now.

Squirrel phase two, and other database applications

A query tool is one of the most notably lacking facilities in Squirrel. For most application I envision, including Walnut, we need a query-by-example tool or some equivalent thereof. Producing the query response as an entity set on the screen nicely integrates it with the other facilities, e.g. the user can select the entities to view them with the application displayers. It is also important to do some kind of simple report generation. A simple but powerful way to do this is to provide yet another entity-based form in which the user marks the entity properties to be printed, and the output formatting. This can be the same form as the query, although a separate one may be easier for the user to understand.

The other major facility Squirrel lacks is the abilty to view and update the data schema. This facility can be added naturally by registering displayers and editors for data schema entities. The data schema editors should modify not only the schema *but also all existing data* (to fit the new schema). The ability to make data schema changes to an existing database is already becoming an important need, as the time has become prohibitive to dump and reload a database in text form to do updates with a text editor. The prohibitive time seems primarily to be in reloading, so dumping is still a viable mechanism for backing up a database. I believe performance improvements could be made in reloading, but it is an intrinsically expensive job to lookup and type-check incoming data so potential improvements are limited.

A major topic not even touched by Squirrel phase one and two is use of graphical display of databases. I believe there is an order of magnitude improvement possible in database access by browsing, zooming, moving, and thinking in a spacial dimension in database display. This will have to wait for a Squirrel phase three, I'm afraid, but I can see it as upward compatible with the less ambitious initial Squirrel and application displays. It isn't clear whether the performance of Cedar Graphics is yet adequate to make graphical database display (say, as boxes and lines on the screen and the ability to move and zoom around) a usable product. But I have the seed of this idea in my head and would like to see it pursued.

Another dimension in which Squirrel phase two could move is procedural storage. Application viewer implementations can be stored as bcds referenced in the database itself; this is actually much easier than it sounds. This makes coordination between different application programs even easier, and simplifies the task of building applications with active components like alarms. We'll have to see how important this becomes.

The three applications I am most interested in seeing after Walnut, particularly because they all integrate with Walnut as a single useful database, are:

1. *Whitepages*. A database of people, organizations, phone numbers, addresses, perhaps even photos. Note there are both public private versions of whitepages, and the user would like to see both superimposed.

2. *Calendar/alarm*. A database of events a person would like to schedule. There might be "day" entity viewers, and "month" entity viewers, for example. Note there can be both public and private databases of events, too.

3. *Super notebook*. A database of one-sentence to one-page ideas, interconnected and indexed in a database of bibliographic references, project notes, etc. I have a lot of ideas on this from trying to build such a thing at CMU. Note yet again that this application has both public and private sections.

Enthusiasm is great for such applications; several lab members have already spoken to me recently about undertaking these. The right combination of facilitators are in place: Viewers, the new CedarDB, Squirrel, and Cedar. All of these have only recently become really useful systems, and my plan and my prognosis is that more than one very successful database application will take off this year. It is even more satisfying to me that as a result of the recent CedarDB and Squirrel work, Walnut and other applications will be unlikely to become separate monolithic programs that are dead ends with respect to uniform user interaction paradigms, sharing databases between applications, and upward compatibility with database system enhancements.