INTEGRATION OF TELIDON WITH BASIS --

A QUERY-DRIVEN BIBLIOGRAPHIC UTILITY

Charles P. Cohen

INFOMART
164 Merton St
Toronto, Ont.

Current affiliation:

Wood Gundy
Royal Trust Tower, Box 274
Toronto-Dominion Centre
Toronto, Ont. M5K 1M7

ABSTRACT

It is difficult for untrained users to search large or highly-structured databases. The standard tools available have been complex query languages (e.g. BASIS) or structured database traversals (e.g. Telidon trees). We suggest and give examples of a search technique which enables naive users to answer their questions without knowing anything about the underlying structure of the database or its native query language. This technique has been applied successfully to the Canadian Record Catalog and the Secretary of State Terminology Databank.

L'intégration de Télidon et de Basis: un système bibliographique de type «Question-réponse» Revue.

Il est difficile pour des utilisateurs non-entraînés de consulter des bases de données importantes ou hautement structurées. Les outils standards disponibles ont été des langages de «demandes» complexes (ex. BASIS) ou des bases de données à structures «transversales» (ex. «l'arbre» Télidon). Nous suggérons et donnons des exemples de techniques de consultation qui permettent aux utilisateurs naïfs de répondre à leurs questions sans rien connaître sur l'infrestructure de la base de données ou de son langage de «demandes» originel. Cette technique a été appliquée avec succès au Catalogue Canadien des Procès-Verbaux (Canadian Record Catalog) et à la Base de données de terminologie du secrétariat d'état.

I. Rationale

As computers have become more powerful, they have also become easier to use (this is sometimes called "overhead" by old-line programmers). The user no longer needs to know how a computer works -- a day's training in BASIS, Inquire, Dialog, ... is sufficient to achieve minimal competence. But per must still:

- understand the structure of the database being searched
- be trained in (or otherwise learn) a weird language.

We were faced with a large online catalog of Canadian recordings, with a potentially large group of utterly unsophisticated (that is, untrained in searching) users. The state of the art was not good enough, and had to be changed.

II. Telidon

Telidon is two things:

- an excellent standard for the transmission of gra-phics and text (now called "NAPLPS")
- a method of structuring data into static "trees", where the nodes are "index pages" or "menus" with choices on them, and the leaves are "data pages" with information on them.

The static structuring makes it difficult to add new data, since index pages must be changed to lead to the new data pages. The tree structure also means that a wrong choice at any node (and there are lots of chances for wrong choices) means the user doesn't get per's desired information.

Most important, though, the fact the the tree is the only possible search mechanism makes many queries impossible. A tree of restaurants, in which

Location: 1) Central 2) East End 3) West End 4) Other

is an early choice, makes it impossible to find all pizzerias. And no way can a user ask for places which serve both shrimp and veal. In general, in a tree-structured database, a page has one index-term -- the concatenation of the choices which lead to it. This is like having a library with only one card catalog -- a subject catalog -- and no cross-references.

But it is so easy to use -- no training required . . .

III. BASIS

(Note -- the comments below, about BASIS, will apply mutatis mutandis to most other sophisticated retrieval packages. I am finding fault with the technology, not with a specific product within it.)

BASIS is a big, powerful, complex IR package -- term-switching, subfield indexing, free-text indexing with or without delimited keywords, sorting, relational-type traversals -- they're all included. The only problem is, to get a list of John Lovesin's LP's (this is in the Canadian Record Catalog), one must say:

```
> FIND FS=A AND ART=LOVESIN
    11 documents found
> SORT DR/D,TI/A
> PRINT TI,DR,LBL,CAT (for title, date of release, label,
                       and catalog number)
```

This is fine for a trained searcher who works with BASIS every day, but not very nice for a record-store clerk. But the clerk needs this information, not the searcher. How to make the computer so smart that it can even understand a clerk -- that's the problem.

BASIS can save and re-execute search strategies (the saved strategy is called a "profile") and let the user specify arguments to them -- so a profile for the above search would look like:

```
> FIND FS=A AND ART=[Enter artist's last name:]
> SORT DR/D,TI/A
> PRINT TI,DR,LBL,CAT
```

and it would be invoked when one said:

```
> /x PERFORMER SEARCH   (or some similar sensible name).
```

Now, our problem is reduced -- we write and store a bunch of profiles, and just have to teach the clerk the names of profiles, and twenty-two pre-packaged search/sort/display combinations, and what each one does. Still not simple enough.

III. Search Strategy Trees

Our twenty-two search strategies (Why twenty-two? List all the ways a clerk might be asked for an album, or a single, or a cut on an album -- by performer's last name, performer's full name, composer, . . .) do fall into a kind of schema:

What do you want -- 1)album 2)song 3)biography

What do you know -- 1)composer 2)performer 3)title

How much of it -- 1)exact 2)some words 3)parts of words

We could document our profiles for the clerk by drawing a tree of choices -- if per followed the branches of the tree, per would arrive at the name of the profile that would search for the right thing, using the right locators, and per would enter that profile name to do the search within BASIS.

A "tree of choices" is precisely the way Telidon retrieval works -- but in Telidon, it's the data that's organized as a tree, not the search strategies.

As it turns out, "tree" is not quite the right structure -- "network" or "graph" is required to handle things like looping back to the main choice menu after a display.

IV. Implementation

We did not supply all users with paper flowcharts of choices and profile names. Instead, we built a user-friendly, fully-prompting front end which was general enough to handle a wide variety of IR applications. It included:

- a network-type database of search-strategy menus -- questions to ask the user to figure out what per was looking for and how per wanted to find it
- the ability to prompt the user for arguments to profiles and automatically send the search requests to BASIS -- the user, having defined per's search strategy, finally sees:

  Please enter performer's last name: LOVESIN

  which is sent as an argument to the appropriate profile.
- display procedures for records selected by a search
- routines for following pointers embedded in the database records -- that is, a user could use keyword searches and Telidon trees in the same database
- graphic (NAPLPS format, naturally) displays for search menus and data, and stored graphics.

This has recently been extended with:

- testing/setting "system variables" -- e.g., number of records selected by previous search, user language, terminal type, etc.

The network database of questions which define searches and prompt for their arguments (one needs about 30 "question pages" for the Canadian Record Catalog) is, itself, a BASIS database, and can be edited and extended using packaged tools.

This software has been applied to the Secretary of State TERMIUM III databank, a 600,000-record multi-lingual terminology file.

V. Extensions

The technique suggested here is useful in situations in which:

- each record has many keys, and
- combinations of keys are often necessary, and
- common search strategies are predictable.

During development, we realized that our friendly structure was not restricted to BASIS databases -- the underlying information could be in a formatted DBMS, and the "profiles" could be replaced by packaged sequences of DBMS instructions -- things like:

```
LIST ALL TITLES VIA TITLE_ARTIST_SET FOR ARTIST WITH
     ARTIST.LAST_NAME EQ "LOVESIN"
```

(that's imaginary, but not by very much). It is likely that a large retrieval system for an unnameable client will be developed using search-strategy menus as the front end of a VAX DBMS-based database.

VI. Conclusion

We made a powerful retrieval system easier for untrained people to use. Our method works for different data structures and supporting software. It doesn't involve much runtime overhead, and it doesn't require any extra indexing of the underlying data. The framework of searches is stored in a database, and is easy to modify and extend. We think it's neat.