The Use of WWW in Biological Research

R.Doelz, Biocomputing Basel T.Etzold, EMBL Heidelberg

Introduction

Information in Biology grows rapidly. Initially, biological retrieval systems used conventional methods of searching and processing large datafiles. Networked versions of the retrieval were sufficiently supported by browsing methods like GOPHER or full–text searches as implemented in the WAIS type of retrieval systems.

However, upon even larger growth in volume, scope and complexity, which is partially caused by the the data arising from Genome projects, today's data in biology are found to be structured, and therefore may be analyzed by more sophisticated retrieval systems. Already earlier, the Sequence Retrieval System (SRS) [1] was established which allows to build indices on any flat file database. The key element of this retrieval system is a query form which allows to query selected fields individually, thus allowing all kinds of very sophisticated queries in individual fields. There is an on–line version of SRS wich can be used to view the mask as presented if the SRS program runs on a local computer system . (Usage: *telnet* to `embl-heidelberg.de` and use the username **`SRS`** as login identification.)

Established Work

The need to have several consecutive forms follows from the complexity of the query. E.g., a typical sequence database entry (see next page, figure 1) is very different in format if compared with a literature reference with respect to possible fields to query A simple query in biology databases, therefore, requires at least three different input parameters:

- Database to be queried (E.g., literature, sequence databases, additional data)
- Field to be searched (E.g., Authors, Titles, Descriptions, etc.)
- Keyword to be searched (E.g., names of Authors, Genes, Organisms)

We have already set up a GOPHER gateway implementation of the SRS system. This (*gopher://bioftp.unibas.ch:12999/1g2getz*) method uses a PERL script type of gateway to compose a query to a server which listens to a proprietary SRS server.

Within the GOPHER system, we communicate data elements from one query to the next by adding those to the selector string in GOPHER, and parse the resulting line with the PERL gateway at the remote server. Consecutive input masks allow for a

```
ID   MMUVOM     standard; RNA; ROD; 2486 BP.
AC   X06339;
DT   17-NOV-1988 (Rel. 18, Created)
DT   12-SEP-1993 (Rel. 36, Last updated, Version 7)
DE   Mouse mRNA for uvomorulin cell adhesion molecule
KW   calcium binding protein; cell adhesion molecule;
KW   cell surface glycoprotein; transmembrane protein; uvomorulin.
OS   Mus musculus (mouse)
OC   Eukaryota; Animalia; Metazoa; Chordata; Vertebrata; Mammalia;
OC   Theria; Eutheria; Rodentia; Myomorpha; Muridae; Murinae.
RN   [1]
RP   1-2486
RA   Ringwald M., Schuh R., Vestweber D., Eistetter H., Lottspeich F.,
RA   Engel J., Doelz R., Jaehnig F., Epplen J., Mayer S., Mueller C.,
RA   Kemler R.;
RT   "The structure of cell adhesion molecule uvomorulin. Insights into
RT   the molecular mechanism of Ca(2+) dependent cell adehesion";
RL   EMBO J. 6:3647-3653(1987).
DR   SWISS-PROT; P09803; CADE_MOUSE.
FH   Key             Location/Qualifiers
FH
FT   source          1. .2486
FT                   /organism="Mus musculus"
FT                   /cell_line="F9"
FT                   /clone_lib="lambda NM-641"
FT                   /clone="F5"
FT   CDS             <1. .2136
FT                   /note="uvomorulin (711 AA)"
XX
SQ   Sequence 2486 BP; 637 A; 648 C; 631 G; 570 T; 0 other;
        1  CCAAAGAACC TGGTTCAGAT CAAATCCAAC AGGGACAAAG AAACAAAGGT TTTCTACAGC
       61  ATCACCGGCC AAGGAGCTGA CAAACCCCCC GTTGGCGTTT TCATCATTGA GAGGGAGACA
      121  GGCTGGCTGA AAGTGACACA GCCTCTGGAT AGAGAAGCCA TTGCCAAGTA CATCCTCTAT
      181  TCTCATGCCG TGTCATCAAA TGGGGAAGCG GTGGAGGATC CCATGGAGAT AGTGATCACA
      241  GTGACAGATC AGAATGACAA CAGGCCAGAG TTTACCCAGG AGGTGTTTGA GGGATCCGTT
      301  GCAGAAGGCG CTGTTCCAGG AACCTCCGTG ATGAAGGTCT CAGCCACCGA TGCAGACGAT
      361  GACGTCAACA CCTACAACGC TGCCATCGCC TACACCATCG TCAGCCAGGA TCCTGAGCTG
      421  CCTCACAAAA ACATGTTCAC TGTCAATAGG GACACCGGGG TCATCAGTGT GCTCACCTCT
      481  GGGCTGGACC GAGAGAGTTA CCCTACATAC ACTCTGGTGG TTCAGGCTGC TGACCTTCAA
      541  GGCGAAGGCT TGAGCACAAC AGCCAAGGCT GTGATCACTG TCAAGGATAT TAATGACAAC
      601  GCTCCTGTCT TCAACCCGAG CACGTATCAG GGTCAAGTGC CTGAGAATGA GGTCAATGCC
      661  CGGATCGCCA CACTCAAAGT GACCGATGAT GATGCCCCCA ACACTCCGGC GTGGAAAGCT
      721  GTGTACACCG TAGTCAACGA TCCTGACCAG CAGTTCGTTG TCGTCACAGA CCCCACGACC
```

***Fig. 1**: Typical entry of a biological database (in this case, the EMBL sequence database storing genetic information). The entry is labelled uniquely in the database via an '**ID**', and can be cross−referenced via the Accession number (**AC**). Note that there is also a cross−reference to the SWISSPROT protein sequence database (**DR**).*

step-by-step assembly of the query. Additionally, the complex query can easily be modified by stepping back instead of retyping all elements.

Other implementations (e.g., *http://golgi.harvard.edu/gilbert-bi.html*) use similar gateway functionality and create separate hypertext elements for each database and field to be searched. This is partially based on hypertext interpretations of the ExPASy at Amos Bairoch's Machine in Geneva. The ExPASy system (*http://expasy.hcuge.ch/*) pioneered the hypertext representation of biological information in hypertext.

The need to store data in order to requery particular items has also been addressed by the Entrez system (*http://www.ncbi.nlm.nih.gov/Search/Entrez/index.html*), which is produced at the National Center for Biotechnology Information. (NCBI). The well-known Entrez, however, is usually a standalone client-server application which does not utilize a WWW environment.

The Challenge

Most of the query systems today allow to assemble a query only in a single step (figure 2). After evaluation of the query, a next query is phrased, which can possibly originate from the previous answer. However, detailing the query might be necessary if there are dependencies within the query (figure 3).

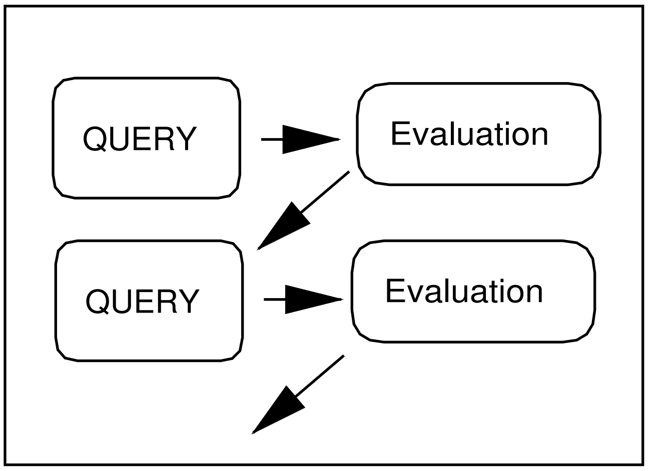


Fig. 2*: Sequential processing of queries as observed with WWW/WAIS type of searches. The query is asked and formulated in a single step.*

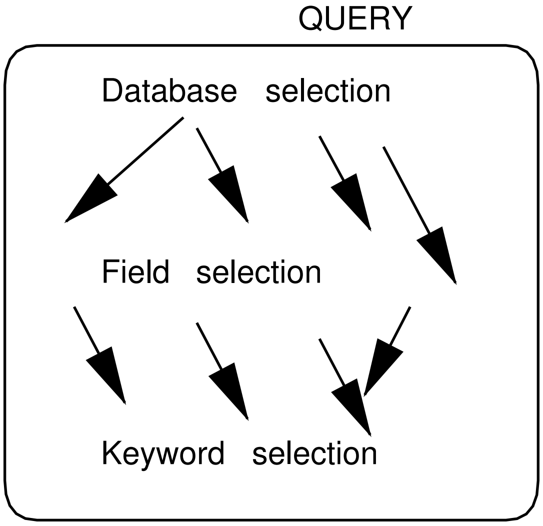


Fig. 3*: A more complex query will have dependencies based on the questions. The database selection might restrict the number of fields to be queried, however, the keyword to be asked will be the same question.*

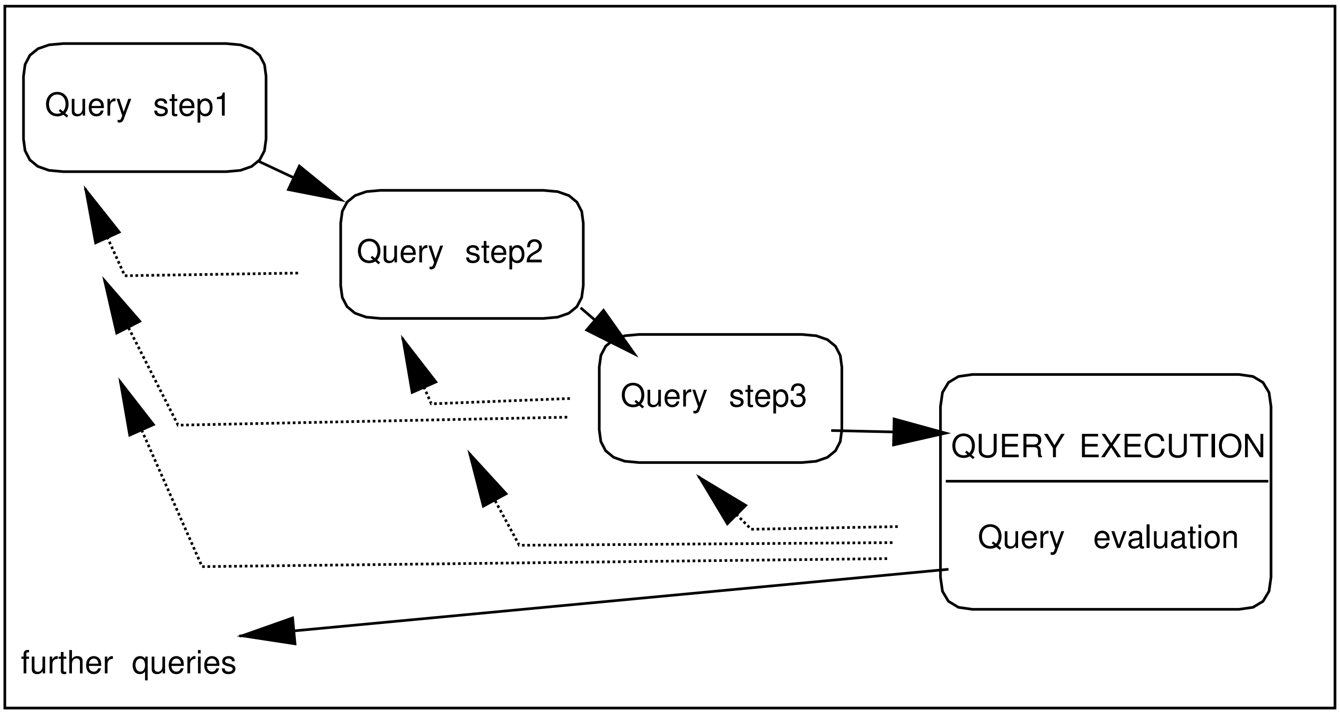


Fig. 4: Advanced model of composing a query via WWW. Instead of allowing the query to be assembled in a single step, intermediate forms allow conditional branching depending on previous input. The key step in the process is to allow the httpd server to carry information from one FORM to the next. (Details see text)

If we had only an option which could remember the previous query form, we would be able to have these queries individually posted in dependence of the database. This requires that the database type, and the fields which are to be queried, are well–defined. Biological databases can be quite diffrent, therefore, a generic rather than a static form generation were necessary. We achieved this type of setup with the model depicted in figure 4.

Why a Special System like SRS?

Forms Definition

Queries in the SRS system can be combined, linked, manipulated with a large variety. All input files are described in a language named "ODD", which can be used to parse any text structured flat file with great ease. A server definition to present forms specific for a given database, therefore, can be achieved easily using the Application Programmers Interface of SRS in order to access the necessary fields.

Flexibility of Links

A very challenging feature of SRS is that it allows to transform the flat file libraries to a fast access network. This is accomplished by links which are either embedded into the data or can even be computed from the data itself [2]. By allowing a transparent linking, a user query can travel along paths not directly specified e.g. if database A is linked to B, and B is linked to C, the end user can directly perform a link from, e.g., C to A without knowing that this is passing B, and even a reverse link in both directions (figure 5).

The work presented in this paper is dedicated to the implementation of a prototype interface which allows to access this SRS system via WWW. We have achieved a working preliminary version using the POST type of method to call routines which, in turn, run a command–line interface of SRS. The output of this is transformed into HTML and presented to the user as WWW page, showing possible links as hypertext (figure 6).

There is a fundamental difference to the usual hypertext link generation: Instead of a single entry or item being linked to another database, SRS links a set of entries to other databases. This is a very fast and useful step, as, in this example, the result can be immediately viewed as result of a subsequent database query.

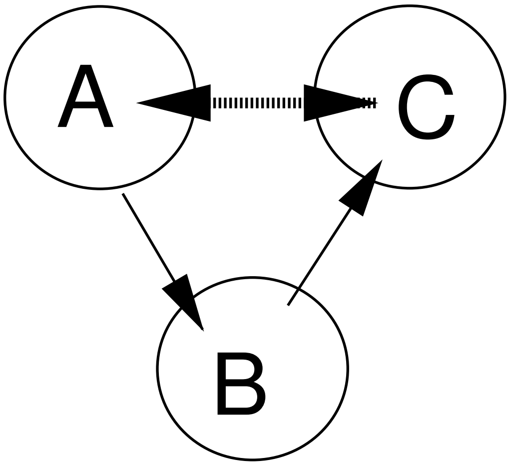
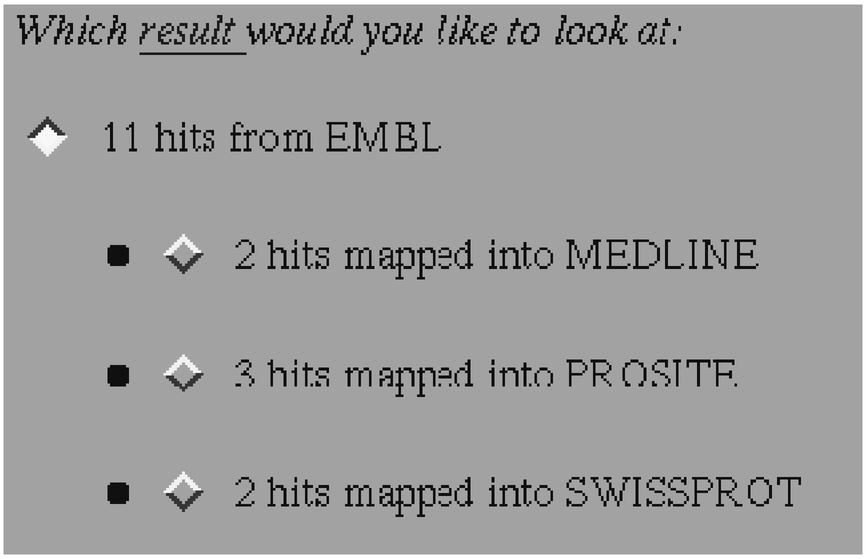


Fig. 5: Virtual links in SRS: As both the links A to B, and B to C are bidirectional and transparently imple– mented, A can access C by 'travelling' along B. This path is known to SRS and does not need to be specified by the user.



Which result would you like to look at:

◆ 11 hits from EMBL

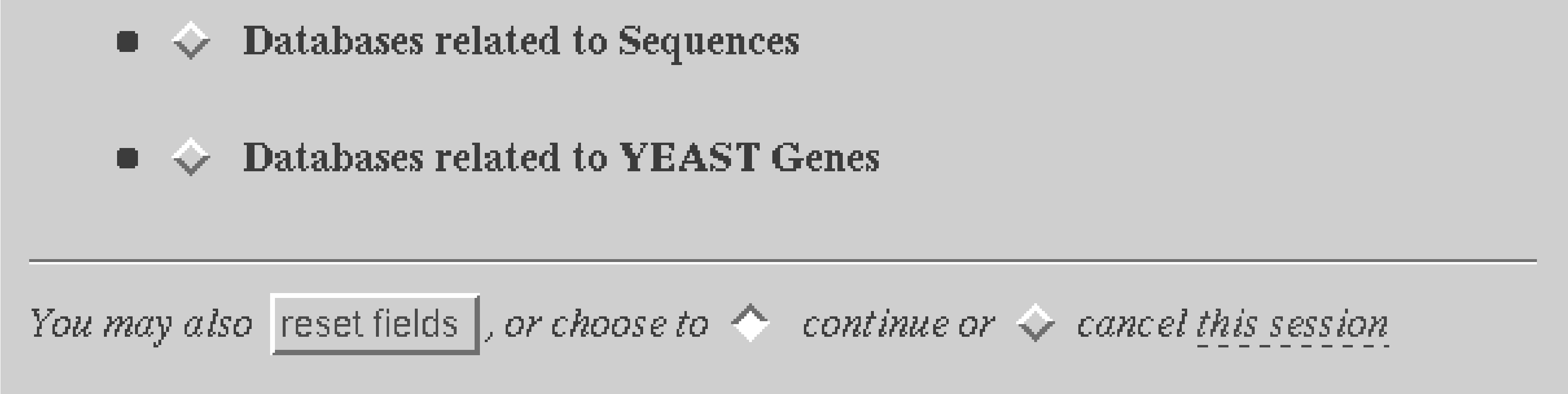
- ◇ 2 hits mapped into MEDLINE
- ◇ 3 hits mapped into PROSITE
- ◇ 2 hits mapped into SWISSPROT

***Fig. 6**: A snapshot from the result evaluation of a SRS query in WWW. The primary question concerned EMBL, but SRS automatically outlines possible answers in databases accessible via links.*

WWW Aspects of the Work

The problem to be solved in this implementation include the ability to use forms in sequential order. In contrast to complex FORMS, we need to present several input pages as several screens during the query process: The user input determines the forms for the steps that follow. Secondly, we found that scrolling FORMS of a certain complexity might not perform as expected and therefore try to present all input on a single page in MOSAIC.

We achieved the solution of these sequential inputs by allowing the WWW server to carry an '**ID**' of a query from one page to another. The typical screen of a SRS WWW screen is shown below in figure 7. The Code which does the conversion is hidden in the last line: The **ID** of the job is the VALUE of a radio button pair. If the continue is chosen, the HTTP daemon will try to access a file with this name in a scratch directory.



- ◇ Databases related to Sequences
- ◇ Databases related to **YEAST** Genes

You may also reset fields *, or choose to* ◇ *continue or* ◇ *cancel this session*