# Chapter 10
# SRS—Access to molecular biological databanks and integrated data analysis tools

D. P. Kreil and T. Etzold
EMBL-European Bioinformatics Institute, Wellcome Trust Genome Campus, Cambridge, UK.

## 1 Introduction

This first section gives an introduction to SRS. Section 2 (A user's primer) is a tutorial that demonstrates basic tasks: simple database queries, exploiting links between databases, exploration of results, and launching analysis tools. Section 3 (Advanced tools and concepts) builds on the skills imparted by the tutorial, and shows how to refine queries, create custom views on data, and use distributed SRS resources. Section 4 (SRS server side) introduces aspects of using a local SRS installation, and outlines how to take advantage of one. Section 5 (Where to turn for help) suggests where to turn to, if this chapter does not address a particular question or problem.

### 1.1 SRS fills a critical need

Everyone who has faced the problem of finding particular information (e.g. experimental results) in the printed literature knows to value computer-readable storage of data. Besides allowing advanced methods of data retrieval and visualization, computerized storage also facilitates systematic combination of data from multiple sources, comparative studies, and methodical application of analysis tools to large sets of data. The rapid growth of available molecular biological databases now gives us access to an unprecedented amount of information by computer.

The increasing specialization of research and the resulting fragmentation of molecular biology is clearly reflected in the choice of databases, which are many and varied in content type and form. The wide differences that can be observed in the database formats and technologies employed are partly caused by the different nature of the results gathered, and cannot solely be blamed on an indifference on the part of the database designers towards efforts at standardization

While adhering to present standards is certainly to be advocated, it would often significantly impede the progress of individual research groups. Also, many databases, and consequently the respective data formats, have grown over history.

The large number of different databases alone is a serious problem for knowing where to turn for specific information. DBCAT, a manually maintained catalogue of databases, currently lists around 400 databases relevant to biology (1). The variation in database formats and technologies that need to be dealt with yet adds significantly to an already complex task of fruitfully accessing the available data. The technical difficulties encountered particularly obstruct more sophisticated leverage of data, such as systematic application of analysis tools, or bringing data from different areas together. Even though many databases directly or indirectly reference data stored elsewhere, these links are difficult to exploit, owing to the large differences between individual database implementations. While a standardization of formats, or at least an agreed upon interface for database interconnectivity, would greatly alleviate these problems, none of the attempts so far has achieved the hoped-for degree of acceptance (2).

SRS has evolved to overcome many critical technical difficulties in database access, and the integration of databases, and analysis tools. Already in earlier versions SRS was considered a promising approach to the problems it set out to solve—respected as 'the paragon of connectivity' (3). The current version SRS-5.1 now offers an automatically maintained catalogue-database of over 350 different databases and their documentation to help users identify databases of interest and locate an appropriate server (4). Users can access all these databases through the uniform SRS query interface on the Web. The interface is simple to use, yet allows complex queries, including the use of logical operators, combining fields from multiple databases, and following implicit and explicit links between databases. Results can be browsed in various views, combining data from several databases. Database entries can be passed to analysis tools such as sequence similarity search programs, restriction map analysis, or phylogenetic algorithms. The results generated can be used in further queries or analyses.

## 1.2 History, philosophy, and future of SRS

The SRS system started out as a Sequence Retrieval System (5) that employed sophisticated parsing and indexing of database text files (a parser is a computer program that breaks down text into recognized strings of characters for further analysis). Plain text is the lingua franca of data exchange. Whether a researcher has entered experimental results into a spreadsheet, or a database centre maintains a repository of complex data structures using an advanced database management system, everyone can provide a dump of their data in the form of plain text files. SRS does not convert these files, but rather leaves the original data in place and reads items directly from there, preserving the original context when the complete entry is viewed. The parsed items are used for display of selected details of a database record, and for the construction of index files that allow efficient data retrieval (6) and searchable links (7) between database

entries. SRS was first applied to databases with information on protein or nucleotide sequences. It has since developed into a much more general data-retrieval tool. It is used, for example, for bibliographical databases, hierarchically structured databases like taxonomies, or clinical data on mutations.

Besides giving researchers the freedom to store information the way they want to, an approach centred on the use of plain text files benefits from a medium that is portable across different computer architectures and is usually easy to read for humans. In the integration of distributed resources, this can be quite helpful in resolving conflicts that may be triggered by the individually evolving components.

SRS parsers, definitions of database structures and interfaces to analysis tools, and a considerable amount of the SRS core functionality itself are written in *Icarus*, a language especially designed for that task. Icarus is used to define descriptive data structures (*meta*-data) which are extensively used to represent SRS concepts like database structures. Icarus code in parsers is interpreted. These two design features allow rapid development: using meta-data reduces the amount of code to be written, and interpreted parsers can be modified and tested fast—it often takes just a few hours to integrate a new database starting from scratch. Not only has this approach proven to scale very well with the number of databases to integrate, the flexible combination of a recursive breakdown of structure and the powerful in-place processing of parsed data that is characteristic of SRS parsers handles even very complex database formats elegantly. Developers who have used an interpreted language before will have little difficulty adapting to Icarus. We think this is reflected by its ready acceptance by SRS server maintainers around the world who have added well over hundred databases to the system.

Icarus is evolving towards a general-purpose object-oriented programming language with special support for recursive lazy parsing and definition of meta-data structures. We expect that it will play a central role in scripting and large-scale data analysis in the future of SRS.

SRS integrates and interfaces to databases and analysis tools of other researchers, and it is also often used as an engine for database access in other systems like OPM (8), and BioKleisli (9)—'wrap and be wrapped'! Communication with an SRS system can be through the World Wide Web (the most popular form of access), from the Unix command line, from within an Icarus script, or using the C application programming interface (API). Current new developments include prototypes of a Corba server, and of a Perl API. Adding support for other language API's (e.g. for Java via JNI) would be straightforward.

## 2 A user's primer

This section will introduce core SRS functionality with simple step-by-step instructions. It concludes with an overview of the screens covered in the primer. We suggest that new users follow the examples given below. The concepts are much more readily understood with some hands-on experience.

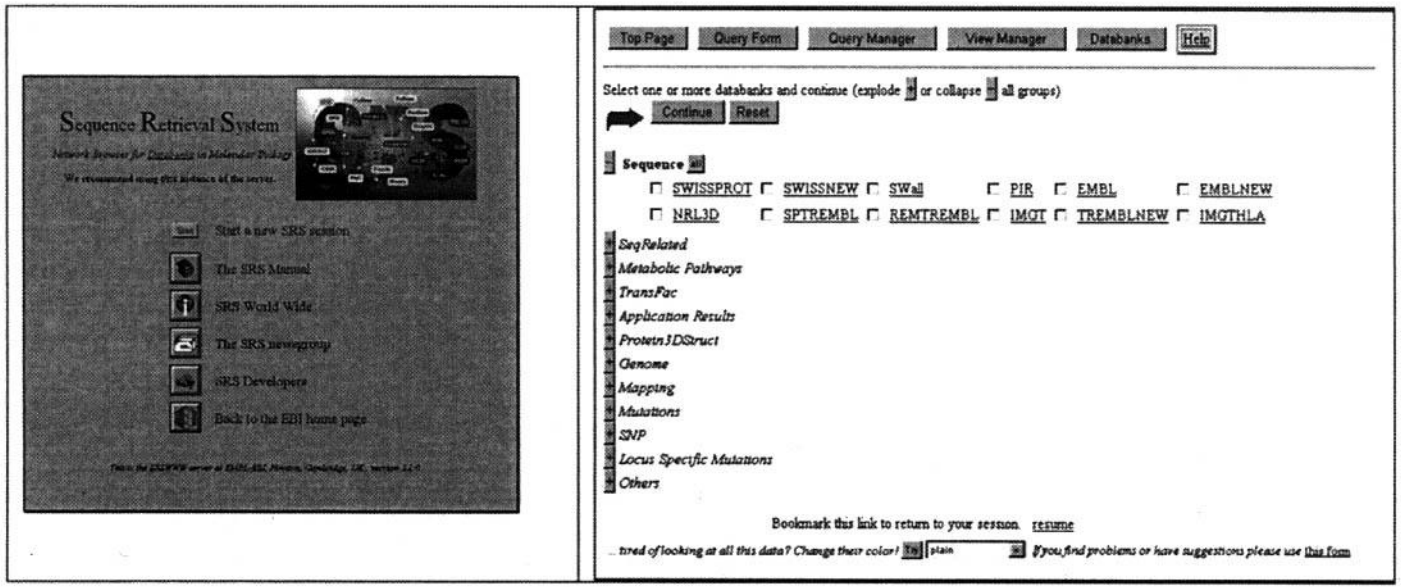



**Figure 1** (a) The SRS Home Page at EMBL-EBI, `http://srs.ebi.ac.uk/`. (b) The 'Top Page': Select the databases to query.

## 2.1 A simple query

SRS is a very powerful and complex system. Nevertheless, the basic features are easy to use, and its interface also caters for the not so experienced user. *Protocol 1* is a guide to performing a simple query and should be accessible for absolute beginners.

### Protocol 1

### Performing a simple SRS query

1. Connect to an SRS server, e.g. *http://srs.ebi.ac.uk/* at EMBL-EBI (Figure 1a).
2. Press the 'Start' button to begin a new SRS session.[a]
3. Select one or several[b] databases to query, e.g. `SWISSPROT`. Database names are displayed in groups. If necessary, expand and collapse groups by clicking on the '+' and '-' buttons respectively (*Figure 1b*).
4. Press the 'Continue' button to get the Query Form (*Figure 2*).
5. Select the field to query in the drop-down box, e.g. 'Description'.
6. Enter the term to search for, e.g. `Tetracycline` (without the quotes). By default, the wildcard '*' will be appended, thus also matching 'Tetracyclines' etc. Deselect this option for a verbatim search.[c]
7. Press the 'Do Query' button to submit your query.

[a] SRS will store your work. If you plan to come back to it, add the 'resume' link from this page to your bookmarks. At EMBL-EBI, an SRS work session is deleted when it has not been used for two consecutive days. Other public sites may have similar mechanisms in place.

[b] When several databases are selected, only fields present in all the selected databases are available in the Query Form.

[c] Avoid using wildcards when not needed since this considerably speeds up the search.

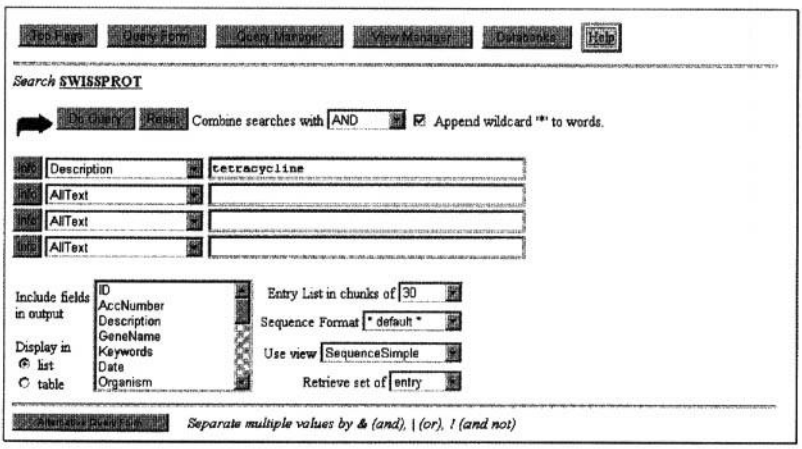



**Figure 2** An SRS query form showing an example of a simple query.

If the list of results is long, the user can page through the list using the links at the bottom of the screen ('go to entries in chunk |. . .|'). Every item in the list of results has an identifier composed of database name and entry-i.d., such as 'SWISSPROT:LPTR_BACST'. They are hyper-linked to the respective complete entries.

To submit another query of the same database(s), press the 'Query Form' button and continue with *Protocol 1*, step 5. For a new query, press the 'Top Page' button and go to Protocol 1, step 3. See *Figure 6* for more options.

## 2.2 Exploiting links between databases

In contrast to many other systems, SRS allows queries that search so-called *links*. The meaning of links between entries depends on the databases involved, yet it is usually evident. For example, an entry *E* in SWISS-PROT is linked to those EMBL entries which contain nucleotide sequences that belong to the protein described by the entry *E*. Similarly, a SWISS-PROT entry *E* is linked to those PDB entries which hold a 3D-structure for the protein described by the entry *E*. EMBL entries are linked to updates in EMBLNEW that supersede the original entries in EMBL. The example given in the next protocol assumes that the reader has followed *Protocol 1*.

### Protocol 2

### Applying a link query to selected entries

#### *Selecting a set of entries*

1 Start from a page showing the results of a query, e.g. after following *Protocol 1*. Performing a simple SRS query.

2 Mark the tick boxes next to the entries you wish to include or exclude. Use the radio button to choose to operate on the 'selected' or on 'all but selected' entries. For example, chose 'selected' and mark entry 'SWISSPROT:TCRB_BACSU'.

#### *Performing a link query*

1 Press the 'Link' button.

2 Select the database(s) to link to [cf. *Protocol 1*, step 3], e.g. EMBL.

3 Press the 'Continue' button to ask for the list of results.

*Protocol 2* shows how to apply a link query to a selected set of entries. The example query retrieves an EMBL entry that contains the nucleotide sequence of the gene that encodes the protein described by 'SWISSPROT:TCRB_BACSU'. In addition, entries that hold larger stretches of genomic DNA containing this gene are retrieved (such as the complete genome of *Bacillus subtilis*).

## 2.3 Using Views to explore query results

There are several pre-defined Views, which present various aspects of a list of results. Besides the Views that display just the entry-id's, or the complete entries, the selection of Views available is database dependent. Protocol 3 shows how to display a set of selected entries in a specific View. The example given in the next protocol assumes that the reader has followed *Protocol 1*.

## Protocol 3

## Displaying selected entries with one of the pre-defined views

### *Selecting a set of entries*

1 Start from a page showing the results of a query, e.g. after following *Protocol 1*.

2 Mark the tick boxes next to the entries you wish to include or exclude. Use the radio button to choose to operate on the 'selected' or on 'all but selected' entries. For example, chose 'selected' and mark the entries `'SWISSPROT:LPTR_BACST'`, `'SWISSPROT:LPTR_BACSU'`, `'SWISSPROT:TCR1_ECOLI'`, and `'SWISSPROT:TCR3_ECOLI'`.

### *Changing the View*

1 Select a View from the drop-down menu next to the 'view' button, e.g. 'Sequence Simple'.

2 Optionally change the number of entries to be shown per screen (*. . . entries in chunks of . . .*').

3 Press the 'view' button.

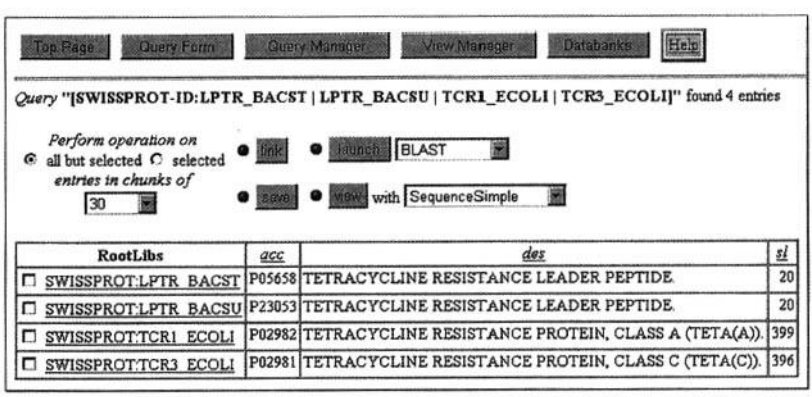



**Figure 3** Showing a selection of entries in the View 'SequenceSimple'.

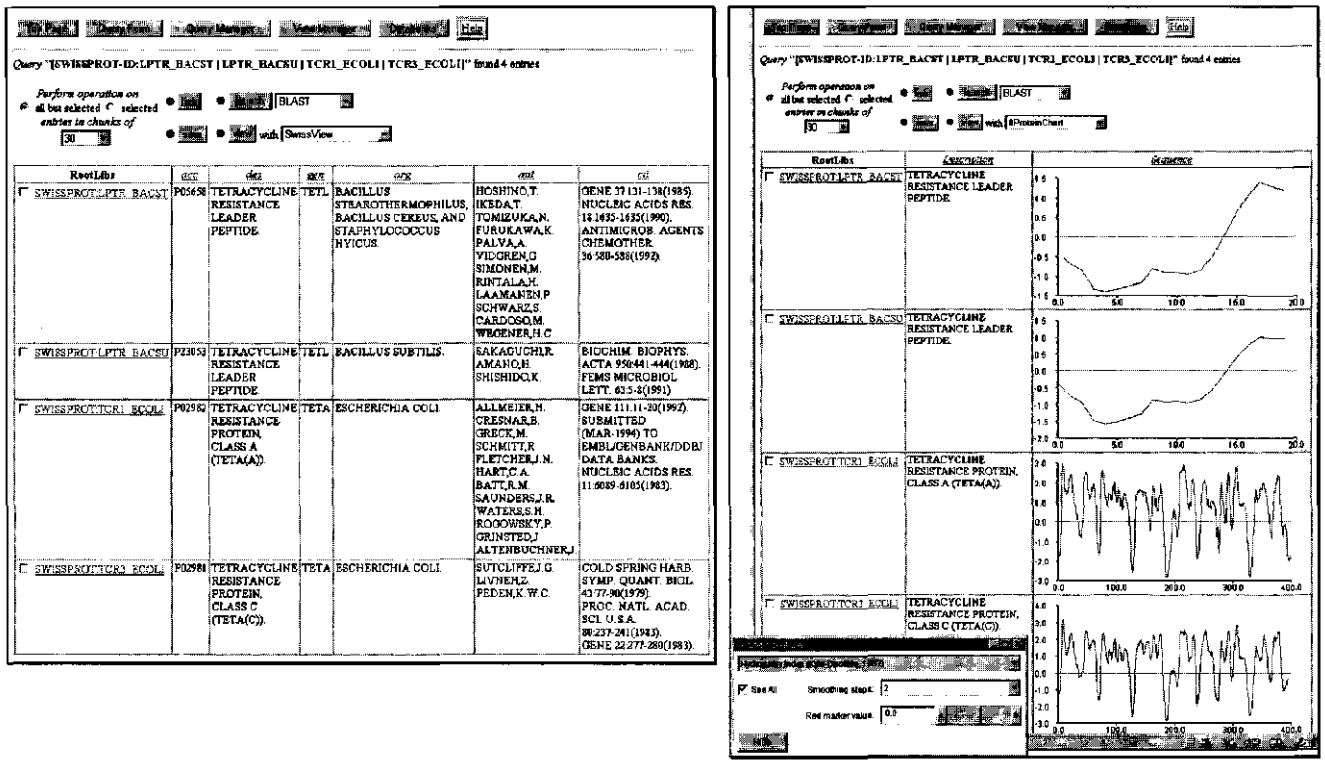



**Figure 4** More examples of Views showing the same selection of entries. The screen on the right (b) employs a Java Applet to display various local protein properties as controlled by the user.

Some pre-defined Views present connected fields from multiple databases by linking the displayed entries to other databases and integrating the results. See below for an example.

## 2.4 Launching analysis tools

Besides browsing results of database field and link queries, external application programs can be used to analyse data. Typical of such applications are database searches by sequence similarity, e.g. BLAST (10), construction of multiple sequence alignments, e.g. CLUSTALW (11), restriction map analysis, and tools predicting various sequence properties (trans-membrane regions, protein secondary structure, etc).

## Protocol 4

## Launching an external application program for selected entries

### *Selecting a set of entries*

1 Start from a page showing the results of a query, e.g. after a query for '`uroplakin`' in the 'Description' field of SWISS-PROT (cf. *Protocol 1*).

2 Mark the tick boxes next to the entries you wish to include or exclude. Use the radio button to choose to operate on the 'selected' or on 'all but selected' entries. For example, chose 'selected' and mark the entry '`SWISSPROT:UPKB_MUSVI`'.

### *Request an application launch*

1 Select a tool from the drop-down menu next to the 'launch' button, e.g. 'SW', the Smith & Waterman hardware accelerated search at EMBL-EBI.

2 Press the 'launch' button to get a page of application options.

### *Adjusting launch parameters and starting the analysis*

1 Check and/or amend your original selection of input to the application (for example, by selecting only a fragment of the sequence to be searched for).

2 Adjust the application parameters according to your data and your preferences. Hyper-links offer additional help that is displayed in a separate window to aid data entry (see *Figure 5a*).

3 Consider turning off the 'automatically display result' feature because some browsers do not correctly support it.

4 Press the 'Continue' button.

5 If the 'automatically display result' feature was disabled, wait until the next page has completely loaded; this may take a while. Then follow the 'Display Results' link at the top of the screen.

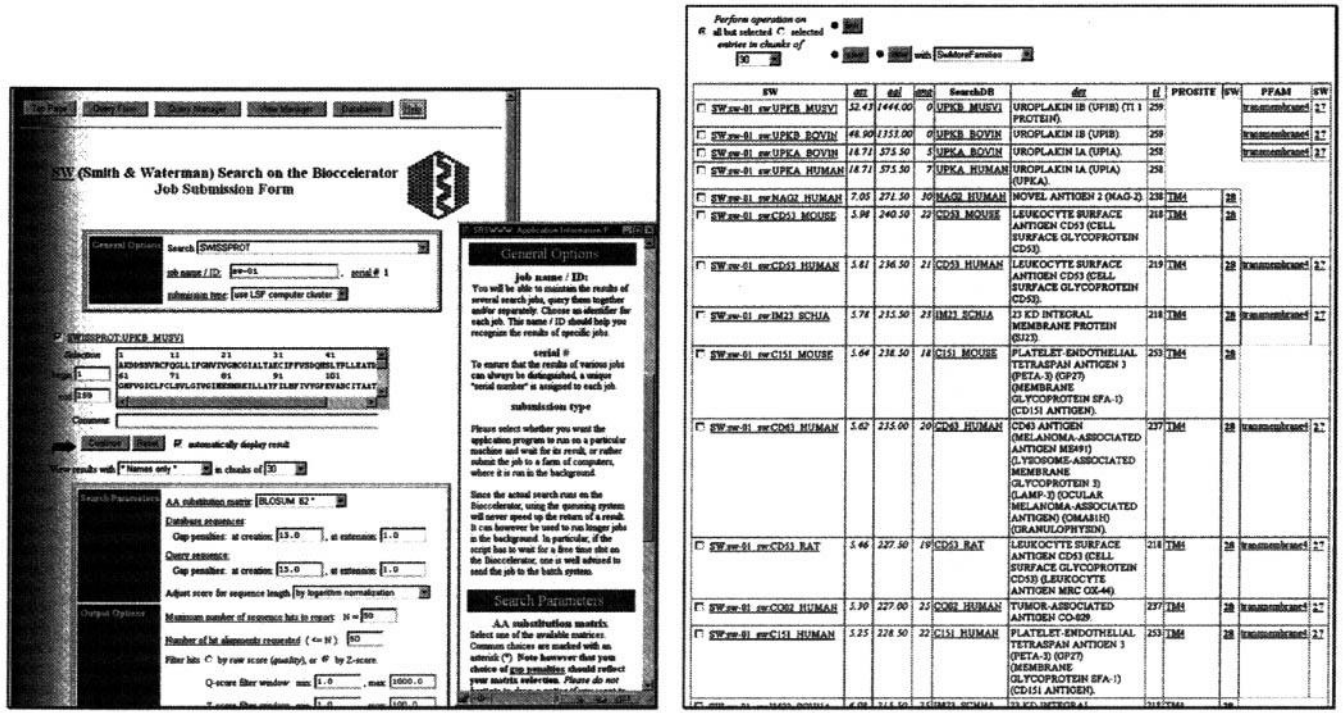



**Figure 5** The application launch page of the 'SW' (Smith–Waterman search) application program. Here users can set application parameters. Clicking on a hyper-linked parameter prompts will open a separate window with related help text. This window is re-used to reduce screen clutter. (b) Results of an 'SW' application launch as seen with the view 'SwMoreFamilies'.

Follow *Protocol 4* to launch an external application for selected entries. The results of the application launch as shown in *Figure5b* are an example of a more sophisticated pre-defined View. (See *Protocol 3* on how to choose a pre-defined View.) For each sequence hit by the search, a normalized score (*Z*-score), the raw score, and the number of gaps in the alignment of the query sequence with the hit sequence are displayed. Then each hit is linked to the database searched (SWISS-PROT, in this case) to display the full-length description and the sequence length from that record. Further links to PROSITE and PFAM give information on membership to well classified families. In our example, the search returned related Uroplakin 1A/B sequences as top hits. Examination of the match data together with the family classifications strongly suggests that the longer ranking sequences are not distantly related to the query, but rather scored because they all have a trans-membrane-4 domain. Also, note how the data in PROSITE and PFAM augment each other.

### 2.5 Overview

Here we give a flow-chart like overview of the interface screens covered by the protocols of this section (*Figure 6*). Users may abort operations and return to the Top Page or a new Query Form by pressing the respective buttons shown at the top of most screens.

## 3 Advanced tools and concepts

In this section we introduce advanced methods of working with SRS. Users are advised to become familiar with these as soon as they have mastered the basics, as the complexity of the molecular biological data and their storage demands fairly sophisticated data handling skills.

### 3.1 Refining queries

Due to the nature of molecular biological databases, even simple questions often require a complex query. Here we explain several techniques that help the user go beyond the first simple queries.

#### 3.1.1 How the system works

As in many cases, it helps to know a little about what is happening under the hood. In a nutshell, SRS parsers read text from a file into *entries*. These are then parsed into *fields*, which usually are further decomposed or processed. In particular, it is the parser's job to extract terms to be written into *index* files. These index files make fast queries of database fields possible. Obviously, they also determine what fields are available for queries, and what terms a query can be matched against.

#### 3.1.2 The index browser

Consider searching for human SWISS-PROT entries. SWISS-PROT has an 'organism' field which can be queried for 'human'—or should it be '*homo sapiens*', or

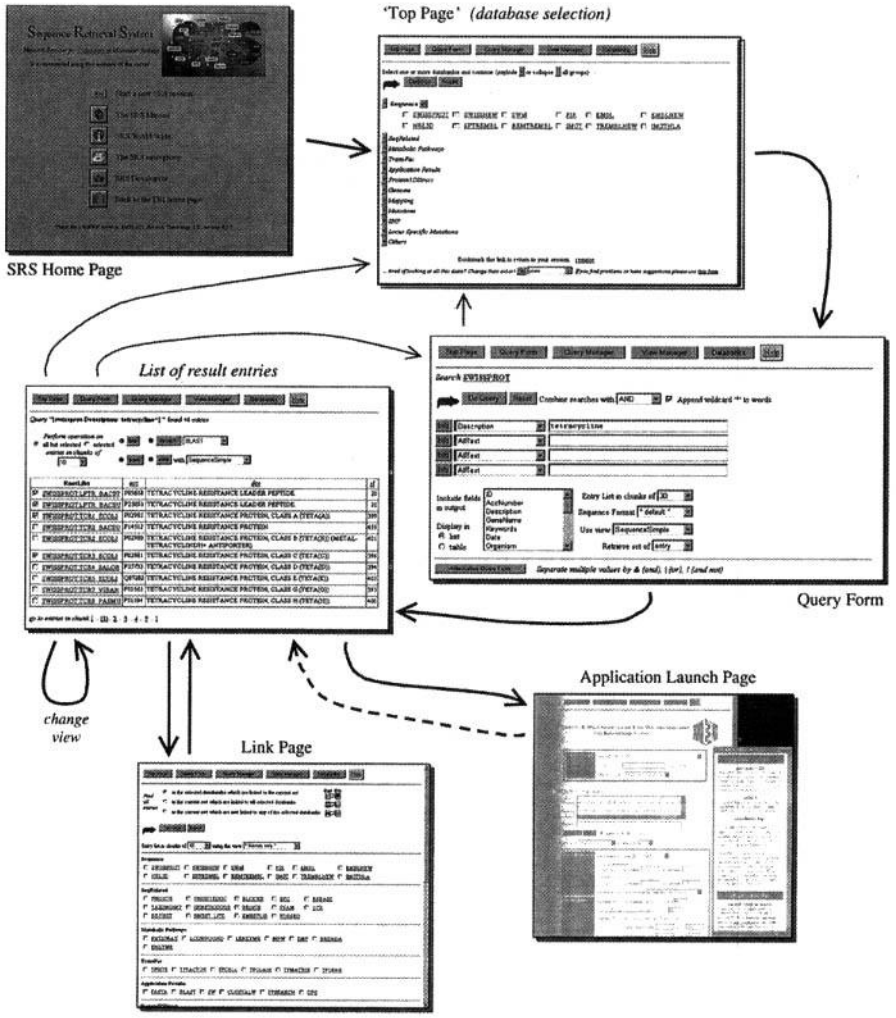



**Figure 6** An overview of the pages discussed so far. The thick *arrows* show the path of usual progression as described in the protocols. There may be an intermediate page before the list of results after an application launch.

'*homo sapiens sapiens*', or a combination of all these terms? The index browser is the easiest way to inspect the terms a query can be matched against. Always consult the index browser when working with an unfamiliar database or database field, and when in doubt. *Protocol* 5 shows how to do that.

Looking at the returned values, one sees the need to restrict the search to 'human' without a trailing wildcard '*' (to exclude, e.g. the human viruses). Checking the index for terms matching 'homo*', one respectively finds the same number of entries for '*homo*' and '*homo sapiens*' as for 'human'. Queries using logical operators (see below) can be used to check the suspicion that the three terms are equivalent and that it suffices to use only one of them.

## Protocol 5

## Browsing the index for a database field

1. In the query form, select a database field from the drop-down box, e.g. 'Organism'.
2. Press the 'Info' button next to the selected field name to get the respective Index Browser.
3. Modify the query mask, e.g. enter 'human*' (without the quotes).
4. Press the 'List values' button.

Typical use of the index browser includes:

(a) Checking whether single words or phrases are available from the index for searching. Most free-text fields (such as 'Description') are usually indexed using single words, while fields with a more standardized vocabulary often store longer phrases in the index (e.g. the 'Keywords' field of SWISS-PROT). When a field is indexed using single words, logical operators must be used to combine words. Instead of asking for 'tetracycline repressor', the query must request 'tetracycline' *and* 'repressor' (see below).

(b) Learning about the terms used in fields that have a controlled vocabulary (e.g. the SWISS-PROT Feature Key, 'FtKey').

(c) Learning about special standardized formats used to represent data in indices of particular fields (see the 'Citation' of SWISS-PROT, as an example).

(d) Looking for alternative spellings or words with typing errors using wildcards or a regular expression. SRS regular expressions are delimited by forward slashes, e.g. '`/p[0–9]+/`', matching 'p' followed by one or more digits.

The index browser also shows documentation for the database field, and status information on the respective index (*Figure 7*).

### 3.1.3 Complex queries with logical operators

Logical operators are a powerful extension to queries, whether one needs to refine or broaden a search, or it is necessary to combine words of a phrase for searching a field with a single word index. SRS supports the following logical operators:

- '*a* AND *b*', which requires *both* the terms *a* and *b* to be present—*in queries*: '`&`';
- '*a* OR *b*', which requires at least one of the terms *a* and *b* to be present—*in queries*: '`|`';
- '*a* BUTNOT *b*', which requires the term *a* to be present and *b* to be absent—*in queries*: '`!`'.

The short forms of the logical operators ('`&`' '`|`', and '`!`') can be used when typing in the query form. Please note that parentheses cannot be used for grouping,

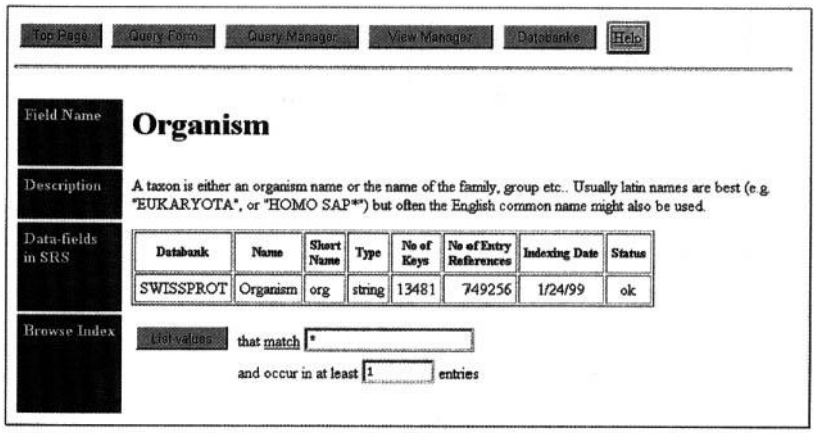



**Figure 7** The index browser page for the database field 'Organism'.

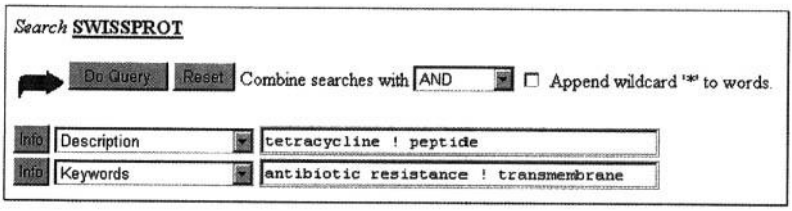



**Figure 8** This query retrieves all entries on tetracycline that are relevant to antibiotic resistance but are neither trans-membrane proteins nor peptides. Note that one can directly search for the phrase '`antibiotic resistance`' in the 'Keyword' index. To search for entries described by '`resistance protein`', one has to ask for each word separately (namely, 'resistance & protein') because the index for 'Description' only contains single words (cf. Index Browser, above).

and that expressions are evaluated strictly left to right (all operators have the same precedence). However, the drop-down menu 'Combine searches with' can be used to select a logical operator that will be applied to join the lines of the query form. *Figure 8* shows a query that retrieves all entries on tetracycline that are relevant to antibiotic resistance but are neither trans-membrane proteins nor peptides.

### 3.1.4 The Query Manager

Often it is helpful or even necessary to break complex operations into smaller, more manageable tasks. SRS stores user queries and makes them available for later reuse through the Query Manager. Selected query results may be annotated, saved to disk, deleted, displayed in particular Views, or combined using logical operators.

Given the previous example, consider the wish to exclude those entries with an uncertain sequence. Following the next few steps as outlined below will accomplish just that. To avoid editing the original query, which one might want

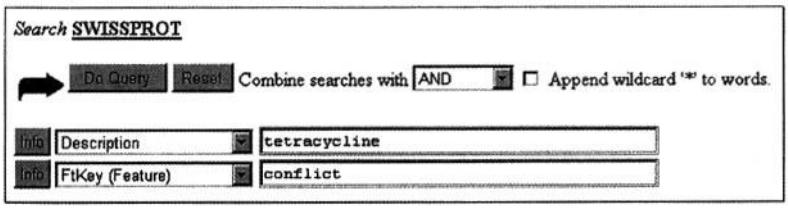



**Figure 9** A query to retrieve entries described as 'tetracycline' and with a sequence about which there is disagreement in the available literature.

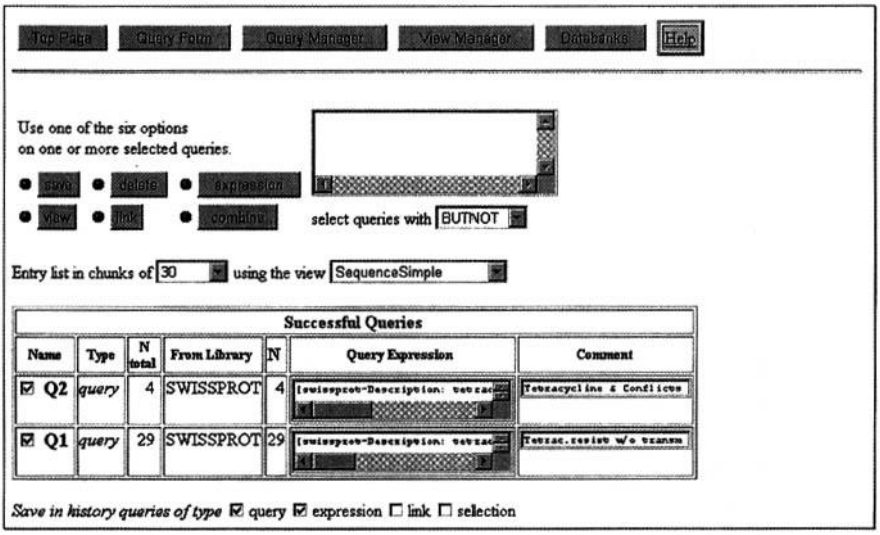



**Figure 10** The Query Manager page offers previous results for reuse and allows for more complex operations.

to keep for reference, run a new query as shown in *Figure 9*. Go to the Query Manager page, select the last two entries, and finally choose 'BUTNOT' as operator and click on the 'combine' button (*Figure 10*).

Advanced users will learn to appreciate the option to enter more complex query expressions directly. The previous result can also be obtained by entering '`[SWISSPROT-Description: tetracycline ! peptide] & [SWISSPROT-Keywords: antibiotic resistance ! transmembrane] ! ( [SWISSPROT-FtKey: conflict] > parent)`' (without the quotes) into the window next to the 'expression' button. Press the 'expression' button to request the evaluation of the query. This example demonstrates several features:

(a) Simple queries take the form '`[DatabaseName-FieldName:QueryString]`'. Logical operators may be used with the same restrictions as in the Query Form. For brevity, the abbreviation of the field name, as displayed in the Index Browser, can be used, e.g. '`[SWISSPROT-des:tetracyc*]`' to query the 'Description' field.

(b) Queries can be combined using logical operators. Here, parentheses *may* be used for grouping.

(c) The syntax for a link query is '$A <$ $B$' or '$A > B$', where $A$ and $B$ can each be a database name or the name of a set of entries (such as those automatically assigned, e.g. '`Q2`'). The first expression retrieves those entries of $A$ that are linked to $B$, while the second one retrieves the entries of $B$ that are linked to $A$ (as suggested by the points of the 'arrows'). Thus, '$A > B$' and '$B < A$' are equivalent expressions.

(d) *Special links*: Some entries can have sub-entries (such as the SWISS-PROT Features). These are indexed and can be queried just like the main entries. Linking with the special pre-defined '`parent`' provides a mechanism to access the entries that contain the retrieved sub-entries. There are other special links for moving up and down in hierarchical databases and to link hits of a search application to the originally searched database (. . .`searchdb`').

SRS Links can always be searched in both directions; they are bi-directional. If a specialized database explicitly refers to a common (e.g. repository) database, one can thus also request linked entries of the specialized database starting from an entry in the common database. The designers of the common database do not need to know even of the existence of the specialized database. SRS also finds entries that are linked via an intermediary. For example, a SWISS-PROT entry that has links to both EMBL and PDB creates (indirect) links between the respective EMBL and PDB entries. Consequentially, indirect links can involve any number of intermediate linking steps. SRS uses the shortest way through the net of databases when resolving a link. SRS server administrators can set up penalties reflecting the desirability of using a particular inter-database link. Users can chose a different path through the network by explicitly requesting a series of links (. . . $A > B > \ldots > C$').

## 3.2 Creating custom Views

The simplest way to have query results displayed in a custom View is specifying the fields of interest in the Query Form. There are two conceptually different View types that can be requested.

(a) **List Views** usually try to conserve as much as possible of the entry as formatted in the original database text file. List Views thus typically act as a filter, displaying only those lines of the entry that have information pertaining to the selected fields. Often, the text is pretty-printed and contains hyperlinks where appropriate. It is not, however, altered in content. List Views are best employed when information in a database entry needs to be read in its original context. This is typically the case for fields that implicitly or explicitly refer to information elsewhere in the entry. *Figure 11* shows an excerpt of an entry that would considerably lose in clarity when displayed out of context as fields in a table.

(b) **Table Views** give a better overview by collecting only requested data for

display. Values are extracted from the original entry format of the database text file, and the data are usually re-formatted for easy readability and concise display. Table Views are ideal for summaries, excerpts, and also when data from different databases needs to be combined. All the Views shown in *Figure 3*, *Figure 4*, and *Figure 5* are Table Views.

The View Manager allows the creation of more complicated Views, such as the one show in *Figure 5b*. Also, Views can be deleted. Named custom Views can be created from scratch, or derived by editing existing Views. Named custom Views are then available just like the pre-defined Views. Users can specify the type of the View (List View or Table View), and whether the abbreviated forms or the complete names of the displayed database fields are to be shown in Table Views. Views are defined for a set of databases (the *root*-libraries). Views also may include fields that are only present in some of the chosen root-libraries. Many elaborate Views link to additional databases that are to be specified (the *leaves* of the View). When all the involved databases have been selected, the user chooses the respective database fields that should be shown, and the formats in which to display data objects (e.g. sequences, alignments) of particular fields.

Advanced options allow the following:

(a) 'Display only number of linked entries': The query that links a displayed entry to a leaf database may result in a set of several entries. For summaries, the size of this set (instead of the individual entries) can be displayed; this includes a hyper-link to the set of entries for inspection of details.

(b) 'Use view to display entries': In List Views, the entries of each leaf-database can be shown in a specified View.

```
SWISSPROT:TER1_ECOLI

AC   P03038;
DE   TETRACYCLINE REPRESSOR PROTEIN CLASS A (TRANSPOSON 1721).
RN   [1]
RA   ALLMEIER H., CRESNAR B., GRECK M., SCHMITT R.;
RL   GENE 111:11-20(1992).
RN   [2]
RA   TRUEMAN P., SHARPE G.S., BARTH P.T.;
RL   SUBMITTED (NOV-1993) TO EMBL/GENBANK/DDBJ DATA BANKS.
RN   [3]
RA   WATERS S.H., ROGOWSKY P., GRINSTED J., ALTENBUCHNER J., SCHMITT R.;
RL   NUCLEIC ACIDS RES. 11:6089-6105(1983).
FT   DNA_BIND      26     45       H-T-H MOTIF.
FT   SITE          54     64       INVOLVED IN BINDING TO [MG-TC]+ (BY
FT                                 SIMILARITY).
FT   METAL        100    100       MAGNESIUM (OF [MG-TC]+ COMPLEX) (BY
FT                                 SIMILARITY).
FT   CONFLICT      65     66       TH -> ST (IN REF. 3).
FT   CONFLICT      80     80       I -> T (IN REF. 3).
FT   CONFLICT     154    155       DA -> ES (IN REF. 3).
SQ   Sequence   216 AA;
     RTKLQPNTVI RAALDLLNEV GVDGLTTREL AERLGVQQPA LYWHFRNKRA LLDALAEAML
     AENHTHSVPR ADDDWRSFLI GNARSFRQAL LAYRDGARIH AGTRPGAPQM ETADAQLRFL
     CEAGFSAGDA VNALNTISYF TVGAVLEEQA GDSDAGERGG TVEQAPLSPL LRAAIDAFDE
     AGPDAAFEQG LAVIVDGLAK RRLVVRNVEG PREKGDD
//
```

**Figure 11** For this List View, the fields 'AccNumber' (accession number), 'Description', 'CitationNo', 'Authors', 'Citation', 'Sequence', and 'FtDescription' (feature sub-entry description) have been selected for display. The 'swiss' (SWISS-PROT) format has been chosen for the sequence field.

(c) 'Use query instead of link': This allows complex operations, such as performing a link following a specified path through the net of databases, or excluding certain entries. The set '`entry`' holds the entry to be displayed and can be used in the query expression.

## 3.3 SRS world wide: using DATABANKS

The public network of SRS servers currently provides the scientific community with access to ~350 different databases. The 40 sites located in 26 countries offer a total of ~1300 databank copies.

To facilitate identification of databases of interest and choosing an appropriate server, a new component of SRS version 5.1 generates DATABANKS, a database of databanks, by traversing the public SRS servers around the world (4). Automated nightly compilation of the data guarantees that the catalogue is up to date.

SRS provides a framework for database documentation in form of the 'databank information page', which has a standardized layout for easy reference and includes a general description, references and internet links, as well as detailed documentation of database fields. Each entry in DATABANKS contains a copy of the SRS databank information page as provided by the server it was collected from, and it concludes with an overview of alternate sites. A typical entry is shown in *Figure 12*. If a stable connection to a particular site could not be established, the site is moved to the end of the list of alternatives. In these cases, data from previous runs are used as backup. A record of when the backup was originally retrieved indicates whether it might be out of date.

DATABANKS is typically searched by databank name or description. Protocol 6 is a guide to solving common questions using DATABANKS. More generally, any field present in the databank information pages, as well as site and server characteristics, can be queried.

### Protocol 6

### Search SRS world wide

#### *Query DATABANKS*

1 Connect to an SRS server that offers DATABANKS, for example *http://srs.ebi.ac.uk/* at EMBL-EBI.

2 Select 'SRS World Wide' from the SRS home page. This leads to a list of known public SRS servers.

3 Select 'Search' to get to the Query Form for DATABANKS.

#### *Identification of databases of interest*

1 Enter a query into the form. (See *Figure 13* for an example.)

2 Restrict the search to a subset of DATABANKS that includes only one site from each group of alternatives[a] as shown in *Figure 13*.[b]

**Protocol 6** continued

3 Press the 'Continue' button to submit your query.

***Choosing an appropriate server***

1 Go back to the DATABANKS Query Form.

2 Enter your query into the form, e.g. ask for databanks named 'PIRALN' (without the quotes).

3 Press the 'Continue' button to submit your query.

[a] Currently, this representative site is chosen as the site that has the most extensive databank information page.

[b] This requests the 'selection flag' by including '[DATABANKS-SelFlag:*]' in your query.

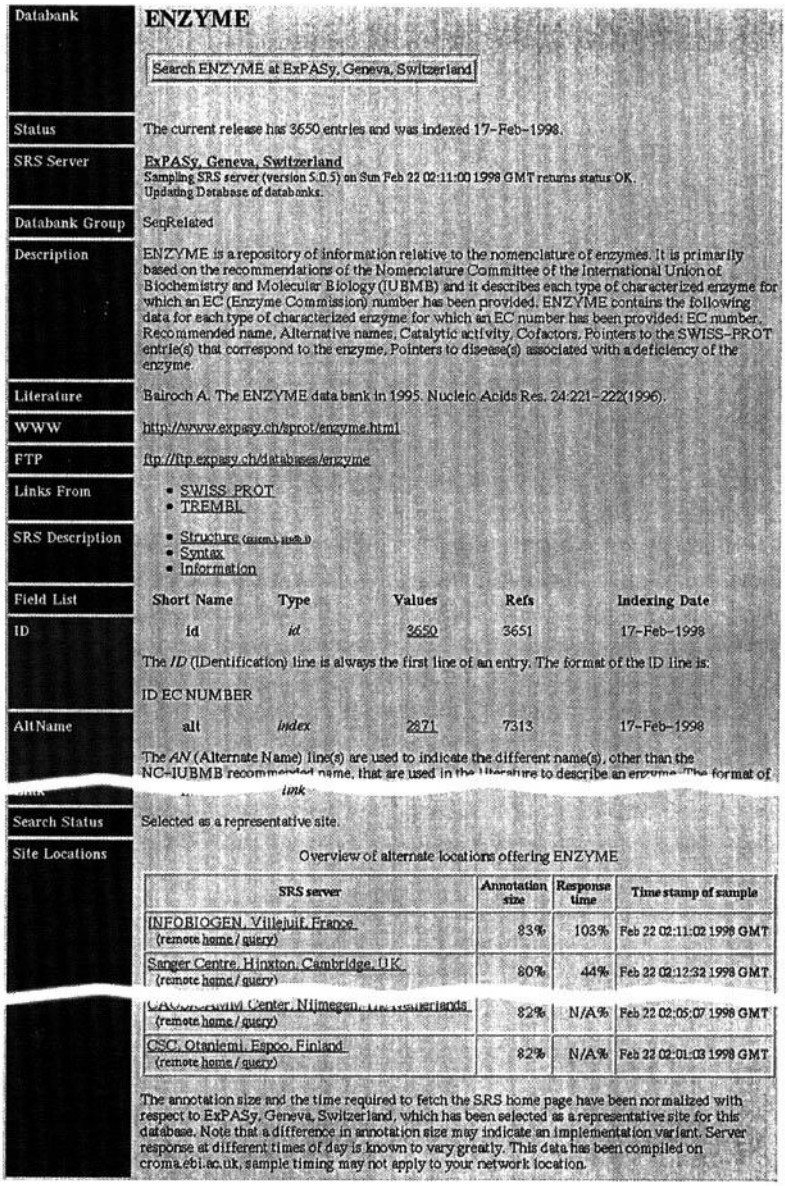
Consider a user who is interested in databases offering sequence alignments, which hold information on well-characterized protein domains or families and can be used for functional assignments or phylogenetic examinations. Selecting the field 'Description' in the query form and asking for 'sequence' and 'align', yields a list of approximately 60 databank copies. By restricting the search as instructed by Protocol 6, part B; however, the query only fetches a more manageable list of 15 representative databanks. In addition to general databases of protein domains or families such as PFAM (12), PRINTS (13), or PIRALN (14), a user will also find specialized databases, such as HOVERGEN (vertebrates, 15), AMmtDB (vertebrate mitochondria, 16), RDP (ribosomes, 17), FSSP and HSSP (protein structure, 18 and 19), or TRANSFAC (transcription factors, 20). The user can now browse the descriptions of the databases retrieved and refine or broaden the search.

Having decided on a particular database, e.g. PIRALN, one usually finds that more than one server maintains a copy of the database, and the list of results shows alternative sites (see *Figure* 14). The number of indexed entries and the release number (where assigned by the server maintainers) help users to choose a nearby site that has a current version of the database.

Both the list of results and the overview of alternate sites compiled for each entry of DATABANKS provide direct links for remote queries. These lead to the respective query forms of the databases at the remote sites as specified.

## 3.4 Interfacing with SRS over the network

This section demonstrates with a few examples how to access SRS directly over a network connection. Currently, that means using the Web to request specific parts of the SRS Web interface. The Web interface is rendered by the program 'wgetz', which sites may provide in different locations, e.g. 'http://srs.ebi.ac.uk/srs5bin/cgi-bin/wgetz' for EMBL-EBI. An easy way of getting the path to wgetz is looking at the 'Databanks' hyper-link on the SRS home page. To request the launch of wgetz, append the character '?' and any parameters that you wish to supply. If you specify more than one parameter, separate them



| | |
|---|---|
| Databank | **ENZYME**<br>Search ENZYME at ExPASy, Geneva, Switzerland |
| Status | The current release has 3650 entries and was indexed 17-Feb-1998. |
| SRS Server | ExPASy, Geneva, Switzerland<br>Sampling SRS server (version 5.0.5) on Sun Feb 22 02:11:00 1998 GMT returns status OK.<br>Updating Database of databanks. |
| Databank Group | SeqRelated |
| Description | ENZYME is a repository of information relative to the nomenclature of enzymes. It is primarily based on the recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (IUBMB) and it describes each type of characterized enzyme for which an EC (Enzyme Commission) number has been provided. ENZYME contains the following data for each type of characterized enzyme for which an EC number has been provided: EC number, Recommended name, Alternative names, Catalytic activity, Cofactors, Pointers to the SWISS-PROT entrie(s) that correspond to the enzyme, Pointers to disease(s) associated with a deficiency of the enzyme. |
| Literature | Bairoch A. The ENZYME data bank in 1995. Nucleic Acids Res. 24:221-222(1996). |
| WWW | http://www.expasy.ch/sprot/enzyme.html |
| FTP | ftp://ftp.expasy.ch/databases/enzyme |
| Links From | • SWISS-PROT<br>• TREMBL |
| SRS Description | • Structure (norm.) (refs.)<br>• Syntax<br>• Information |

| Field List | Short Name | Type | Values | Refs | Indexing Date |
|---|---|---|---|---|---|
| ID | id | *id* | 3650 | 3651 | 17-Feb-1998 |

The *ID* (IDentification) line is always the first line of an entry. The format of the ID line is:

ID EC NUMBER

| | | | | | |
|---|---|---|---|---|---|
| AltName | alt | *index* | 2871 | 7313 | 17-Feb-1998 |

The *AN* (Alternate Name) line(s) are used to indicate the different name(s), other than the NC-IUBMB recommended name, that are used in the literature to describe an enzyme. The format of

*link*

| | |
|---|---|
| Search Status | Selected as a representative site. |
| Site Locations | Overview of alternate locations offering ENZYME |

| SRS server | Annotation size | Response time | Time stamp of sample |
|---|---|---|---|
| INFOBIOGEN, Villejuif, France (remote home / query) | 83% | 103% | Feb 22 02:11:02 1998 GMT |
| Sanger Centre, Hinxton, Cambridge, UK (remote home / query) | 80% | 44% | Feb 22 02:12:32 1998 GMT |
| CAOS/CAMM Center, Nijmegen, The Netherlands (remote home / query) | 82% | N/A% | Feb 22 02:05:07 1998 GMT |
| CSC, Otaniemi, Espoo, Finland (remote home / query) | 82% | N/A% | Feb 22 02:01:03 1998 GMT |

The annotation size and the time required to fetch the SRS home page have been normalized with respect to ExPASy, Geneva, Switzerland, which has been selected as a representative site for this database. Note that a difference in annotation size may indicate an implementation variant. Server response at different times of day is known to vary greatly. This data has been compiled on croma.ebi.ac.uk, sample timing may not apply to your network location.

**Figure 12** A typical DATABANKS entry. The entry contains a copy of the respective remote SRS databank information page, which includes a description, references and links, as well as detailed documentation of database fields and indices. It concludes with a listing of alternative sites that offer ENZYME. Direct links to these sites and the remote query forms for ENZYME are provided. For uses in the network vicinity of a particular DATABANKS server, the relative response times compiled by that server give a clue to the net distances to other sites ('N/A' indicates problems connecting at the specified time).

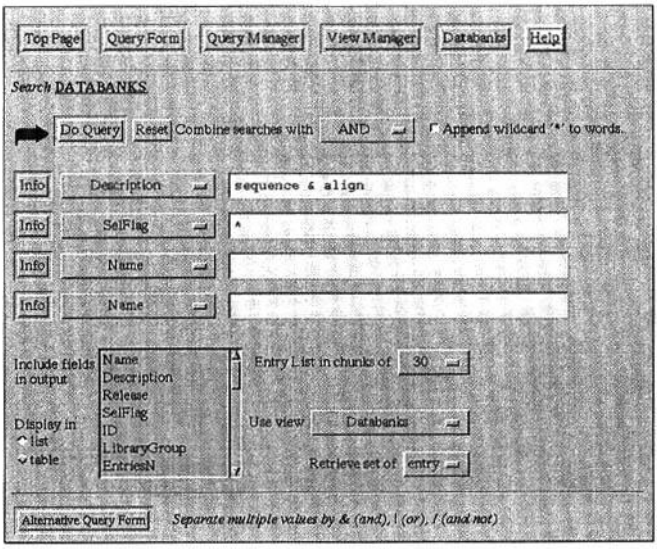
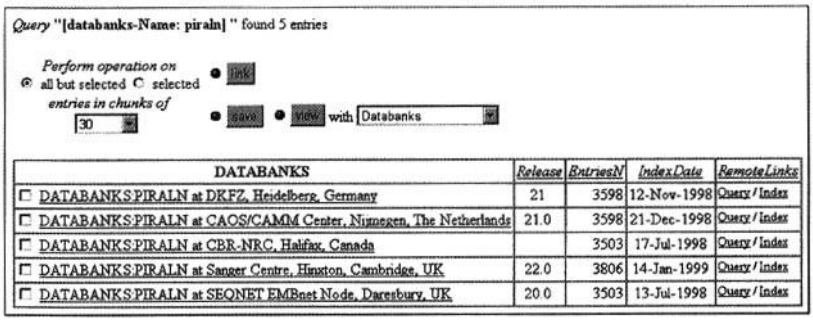



**Figure 13** Query for databanks that have a description containing the terms '`sequence`' and '`align`'. The second line of the query form requests that the results be restricted to one representative databank for each group of alternatives.




| DATABANKS | Release | EntriesN | IndexDate | RemoteLinks |
|---|---|---|---|---|
| DATABANKS:PIRALN at DKFZ, Heidelberg, Germany | 21 | 3598 | 12-Nov-1998 | Query / Index |
| DATABANKS:PIRALN at CAOS/CAMM Center, Nijmegen, The Netherlands | 21.0 | 3598 | 21-Dec-1998 | Query / Index |
| DATABANKS:PIRALN at CBR-NRC, Halifax, Canada | | 3503 | 17-Jul-1998 | Query / Index |
| DATABANKS:PIRALN at Sanger Centre, Hinxton, Cambridge, UK | 22.0 | 3806 | 14-Jan-1999 | Query / Index |
| DATABANKS:PIRALN at SEQNET EMBnet Node, Daresbury, UK | 20.0 | 3503 | 13-Jul-1998 | Query / Index |

**Figure 14** The results of a query for databanks named '`PIRALN`'. The number of indexed entries and the release number (where assigned by the server maintainers) help users to choose a nearby server that offers a current version of the appropriate database.

with '`+`' instead of with spaces. To include literal spaces in your parameters, replace them with '`%20`'.

There are various ways in which entries can be displayed, for example:

(a) **As plain lists of entries**: Specify your query as you would in the expression box of the Query Manager.

(b) **As individual entries**: Specify the '`-e`' (entry) switch together with an SRS query, e.g. `.../wgetz?-e+[SWISSPROT-ID:LPTR_BACST]`.

(c) **As lists of entries that can be browsed**: Specify the '`-sl`' (sequence list) switch with an SRS query. This yields a page that looks like the list of results in the standard SRS Web interface.

For each of the above, a particular View may be requested, e.g. '`-view + SequenceSimple`'.

Some SRS functions can be accessed directly as entry points into the system, e.g. requesting the query form for a specified set of databases. The switch '`-l`' (libraries) sets the databases to operate on, the '`-fun`' (function) switch selects the operation requested: e.g. '`.../wgetz?-fun+PageQueryForm+-l+SWISSPROT%20SWISSNEW`'.

Coming back to a previous session, or to maintain data across several `wgetz` calls, a user context needs to be specified with the '`-id`' switch. The id of a session is part of most of the links it displays; it is easily seen in the 'resume' link on the Top Page (cf. *Figure 6*). Chapter 3 of the on-line manual contains further information on linking to SRS servers using the Web.

Future versions of SRS will also support other mechanisms of remote access to SRS servers. We have already developed a prototype Corba server. Generally, SRS is able to serve structured data and methods that operate on them to any object oriented or procedural environment.

# 4 SRS server side

SRS may be installed locally on your site. This section outlines how to take advantage of a local SRS system.

## 4.1 User's point of view

In addition to accessing their SRS system over the Web, local users can run queries from the command line, use scripting to automate repetitive tasks or perform elaborate analyses, and modify parsers (if the administrators of the system permit that).

To activate a particular installation of SRS on your site, use '`. prep_srs.sh`' from Bourne, Korn or POSIX shells, or '`source prep_srs`' from C shells. If the sourced files are not in your path, they are provided in the '`etc`' sub-directory of the place where SRS was installed (the SRS *root* directory). After this initialization, one can use the '`getz`' program to query the SRS system; e.g. '`getz [SWISSPROT-Des:tetracycline] -f des`' will print those lines of the matching entries that hold the Description field. Use Chapter 4 of the on-line manual as a reference and for more examples.

While '`getz`' is quite popular among users and is frequently used within shell scripts or Perl programs, it is inefficient for repeated fine-grained access to data. The only way to avoid a huge number of '`getz`' invocations is to request many fields in one query, and users often end up parsing the resulting '`getz`' output—what a waste considering that SRS has already parsed the entry into fields! A better solution is integrating a programming language with SRS:

(a) The entire SRS functionality can be accessed through the C application programming interface (API). See Chapter 12 of the on-line manual for instructions on linking your C code with the SRS library, and several examples.

More examples can be found in the 'demo' sub-directory of the SRS installation. This is the interface to use for the most extensive access to SRS features. However, only the functions introduced in the examples are guaranteed to be supported in future versions—other functions may change as we improve and extend the system.

(b) More and more, Icarus allows access to SRS functionality. Already now, SRS queries and access to token tables is supported. Token tables hold the results of the parsing process. In future versions, application launching and all other SRS functions and data will be accessible through Icarus. The example shown in *Figure 15a* prints the entry-ids and the molecular weights of all matches to the query '`SWISSPROT-des:Tetracycline*`'. The example shown in *Figure 15b* takes an SRS query string on the command line and dumps the '`fields`' token table. See Chapters 5–7 of the on-line manual to learn more about Icarus.

(c) Future versions of SRS will offer native language interfaces to other general purpose languages like C++ or Java, and to popular scripting languages like Perl (for which a prototype has already been developed)

Sometimes users find they struggle to extract particular data through a query. Often a simple modification or extension of the parsers involved helps a lot. Clearly, many users do not want to deal with the parsers themselves, or they even may not have permission to do so. Still, they need to know when it pays to ask for help. Typical problems that can be solved include:

```
#!/bin/env icarus

$my_set=$Query:'SWISSPROT-des:Tetracycline*'
if:$my_set
{ foreach:[$entry in:$my_set]
  { $Print:| - Entry ($entry.name)
    foreach:[$token in:$entry.tokens:molweight]
      $Print:|   molecular weight = ($token.str)
    $entry.delete # free space allocated for $Entry object
  }
  $my_set.delete  # free space allocated for $Set object
}
$dp:''  # to force stdout flush
```

```
#!/bin/env icarus

$dp:''
if:$ArgN<2
{ $Print:|||  usage:  ($Arg:1) (SRS-query)
  $dp:''  # to force stdout flush
  $Exit
}

$query=$Arg:2

$my_set=$Query:$query
if:$my_set
{ $Print:|||  Set returned by query '$query'
  foreach:[$entry in:$my_set]
  { $Print:|||  Entry ($entry.name)
    foreach:[$token in:$entry.tokens:fields]
    { $Print:" - token code:  "    $Print:$token.code
      $Print:"   token string:\n" $Print:$token.str
    }
    $entry.delete # free space allocated for $Entry object
  }
  $my_set.delete  # free space allocated for $Set object
}

$dp:''  # to force stdout flush
```

**Figure 15** Example for using Icarus as a cripting language (for SRS version 5.1 or higher). The program displayed on the left prints the entry-ids and the molecular weights of all matches to the query 'SWISSPROT-des:Tetracycline*'. The program shown on the right takes an SRS query string as its argument on the command line and prints the parser's 'fields' token table: For each token, it shows both the *token code* (a label, which can be used to determine how a token is further processed), and the *token string*. The *token object* (which is used, e.g. to hold the sequence object) could have been accessed using $token.obj. $Query returns a $Set object. See Chapter 7 of the on-line manual for a reference of Icarus classes and chapters 5 and 6 to learn more about Icarus syntax and functions.

(a) **Extracting data from a field**: If you find yourself trying to query certain *parts* of a field regularly, that extraction process should be delegated to the parser. In a parser, this extraction can also be much more sophisticated.

(b) **Querying phrases**: The parser controls which terms can be matched by a query (see Sections 3.1.1 and 3.1.2). Sometimes the way a parser breaks a field into indexed terms is not well suited for answering a particular question. Changing the parser or adding a new field that takes particular requirements into account solves the problem. Consider a comment field, for which the parser extracts single words for indexing; the query '`not & known`' will not only find the phrase 'not known', but also retrieve instances that contain the two words out of context. To address this problem, the parser might be changed to watch, e. g., for words preceded by 'not', and write the two word phrase to the index.

(c) Similarly, queries that use logical operators to combine data from several fields are sometimes not sufficient if the necessary context has been lost in the indexing process. Changing the parser is again the only satisfactory solution.

## 4.2 Administrator's point of view

Within the scope of this chapter, we can only give some help in getting an SRS system up and running. Each of the following topics, however, should be of further interest to SRS server administrators:

(a) The automation of tasks in server, database, and application maintenance.

(b) The design and implementation of new modules for the integration of databases and application programs. This must include a discussion of lazy, forced, and explicit parsing, and how to best make use of this layered approach.

(c) A manual of Icarus that introduces new users, and serves as a reference to experienced programmers. Icarus is an efficient, expressive scripting language that has been particularly designed for the purpose of describing data-structures and the rapid development of highly flexible parsers. Yet Icarus has evolved towards a general-purpose object-oriented programming language, which we expect to play a central role in scripting and large-scale data analysis in the future of SRS.

Clearly, these issues are all rather complex and deserve a dedicated discussion in themselves.

### 4.2.1 How to get your own SRS system

Download SRS from `ftp://ftp.ebi.ac.uk/pub/software/unix/srs/`, the current version is in file '`srs5.1.tar.gz`' and is freely available to the public. Please note that future versions of SRS will continue to be freely available for academic non-commercial use. Future versions of SRS and services are provided by LION Bioscience Ltd (`http://www.lionbio.co.uk/`).

To install SRS in a place of your choice, unpack the distribution file (e.g. '`gzip -cd srs5.1.tar.gz | tar -xf -`'). This creates a directory for SRS,

which we call the SRS *root directory*, and which we will print as '. . .' for the rest of this chapter. From this directory, first run '`./srsinstall all`'. To also install the SRS web interface, run '`./srsinstall www`' next. At completion, this prints two lines that have to be inserted into the '`srm.conf`' file of your web server. Ask your system administrator to do this if necessary. If there is no web server installed on your site, save the lines for later reference. You then need to have a Web server installed. We now normally use Apache (see `http://www.apache.org/`).

### 4.2.2 Basic configuration steps

SRS obviously comes without any databases or applications. However, a large number of *database modules* come with the distribution. These modules allow SRS to index and access the respective databases. The files that constitute a module are stored in '`.../icarus/db/`'. For each database, respectively, there is:

(a) A '*module-name*`.is`'-file that holds the parser.

(b) A '*module-name*`.i`'-file defines the database fields and links that can be queried.

(c) A '*module-name*`.it`'-file that contains optional database documentation, which is used to construct the databank information page.

The databank information pages usually contain fields that report database sources, indicating where the required files can be downloaded. This can directly be read in the '*module-name*`.it`'-files. More conveniently, this information can be queried for in the Database of Databanks as introduced in Section 3.3. This also gives access to the hundreds of modules developed worldwide, of which only some are included with the standard distribution. If a database module is offered by remote servers only, download the files that constitute the respective database module by following the hyper-links provided in the 'SRS Description' field of the DATABANKS entry.

Edit '`.../icarus/db/srsdb.i`' to reflect the databases available at your installation:

(a) There needs to be a '`file:`*module-name*`.i`' command to include the appropriate field definitions.

(b) Each database needs a unique id number: Edit the list '`libIds`' accordingly. When adding a new database, specify the database as defined by '`$Library`' in the '*module-name*`.i`'-file.

(c) The locations of the database files are declared in the '`libs`' list of '`$LibLoc`' statements. SRS uses this list to determine which databases are available. Therefore, remove statements for databases not kept at your site, or turn them into comment lines, which start with a hash ('`#`'). Edit the other directories as appropriate. Again, when adding a new database, specify the database as defined by `$Library` in the '*module-name*`.i`'-file.

To commit your changes, run '`srssection`'. In preparation for this or any other SRS program, initialize your environment: Users of Bourne, Korn or POSIX shells

execute '`..../etc/prep_srs.sh`', while C shell users '`source .../etc/prep_srs`'. (This sets the path for executables, the environment variable '`$SRSROOT`' and several others.)

All of the above also applies to modules for application programs, only they are not yet included in DATABANKS.

### 4.2.3 Prepare your databases for access through SRS

After preparing the appropriate database text files and configuring SRS accordingly, a set of indices has to be created for each database. These indices allow fast queries of database fields and links.

The program '`srscheck`' generates the script '`srsupdate`' and shows which indices need to be created or updated. Use the '`-l`' (library) switch to check the indices of a particular database only. Run `srsupdate` to actually build the required indices. For small databases, this is a matter of minutes. Indexing large databases like EMBL may take several hours, though. At successful completion, your databases are ready for SRS queries.

## 5 Where to turn to for help

If you experience difficulties, do not despair!

(a) If you have installation problems, please read carefully through the `README` file that comes with the distribution.

(b) Try to make good use of the on-line manuals. While it is sometimes difficult to find particular bits of information there, they are quite extensive, and should certainly be consulted first. As the individual sections can be quite large, using the text search feature of your browser can be quite helpful. Users will want to focus on Chapters 1 and 2, which deal with the Web interface and the Query language, respectively. Administrators should start with Chapters 4, 10, and 11. Chapters 10 and 11 contain a lot of information on installation and set-up, and Chapter 4 is a reference of SRS programs available from the command line. Use the to search within chapters, and do not be fooled by the section headings. Please note that the chapter numbers refer to the *on-line manual* of *SRS-5.1*.

(c) Try asking colleagues for help. There is a strong community of SRS aficionados out there that will be happy to help (that is, *if* you have looked through the on-line manual first).

(d) Send your question to *`news://localnews/bionet.software.srs`*, a newsgroup dedicated to SRS related questions.

(e) If you discover a bug, or have a problem of very technical nature, report it by e-mail to *`srsdev@ebi.ac.uk`*.

If you need more help configuring and customising your system, you may want to learn about SRS workshops and consultancy available from LION Bioscience Ltd (*`srs-info@lionbio.co.uk`*).

## Acknowledgements

Many people have contributed to the wealth of SRS database and application modules that is publicly available now, and we are indebted to them all!

We wish to warmly thank Rob Falla for his help and the fruitful late-night discussions. Also, we gratefully thank Mark Wooding who thoroughly examined the final draft of this chapter. Any errors were certainly introduced afterwards!

## References

1. Discala, C., *et al.* (1998). DBCAT, the public catalog of databases. INFOBIOGEN, Villejuif, France;contact `discala@infobiogen.fr`.
2. Frishman, D., Heumann, K., Lesk, A., and Mewes, H.-W. (1998). *Bioinformatics*, **14**, 551.
3. Brenner, S. E. (1995). *Science*, **268**, 622.
4. Kreil, D. P. and Etzold, T. (1998). *Trends Biochem. Sci.*, **24**, 155.
5. Etzold, T., Ulyanov, A., and Argos, P. (1996). In *Methods in enzymology* (ed. R. F. Doolittle). Vol. 266, p. 114. Academic Press.
6. Etzold, T. and Argos, P. (1993). *Comput. Appl. Biosci.*, **9**, 49.
7. Etzold, T. and Argos, P. (1993). *Comput. Appl. Biosci.*, **9**, 59.
8. Markowitz, V. M. and Ritter, O. (1995). *J. Comput. Biol.*, **2**, 537.
9. Davidson, S. B., Overton, C., Tannen, V., and Wong, L. (1997). *Int. J. Digit. Libr.*, **1**, 36.
10. Altschul, S. F., *et al.* (1990). *J. Mol. Biol.*, **215**, 403.
11. Thompson, J. D., Higgins, D. G., and Gibson, T .J. (1994). *Nucleic Acids Res.*, **22**, 4673.
12. Sonnhammer, E. L., *et al.* (1998). *Nucleic Acids Res.*, **26**, 320.
13. Attwood, T. K., Beck, M. E., Bleasby, A. J., and Parry-Smith, D. J. (1994). *Nucleic Acids Res.*, **24**, 182.
14. Barker, W. C., *et al.* (1998). *Nucleic Acids Res.*, **26**, 27.
15. Duret, L., Mouchiroud, D., and Gouy, M. (1994). *Nucleic Acids Res.*, **22**, 2360.
16. Lanave, C., *et al.* (1999). *Nucleic Acids Res.*, **27**, 134.
17. Maidak, B. L., *et al.* (1997). *Nucleic Acids Res.*, **25**, 109.
18. Holm, L., *et al.* (1992). *Protein Sci.*, **1**, 1691.
19. Sander, C. and Schneider, R. (1991). *Proteins*, **9**, 56.
20. Heinemeyer, T., *et al.* (1998). *Nucleic Acids Res.*, **26**, 362.

*This page intentionally left blank*