Version: V1.0

Date: 07.05.19

Author: Dr Georgina R Toye

Address: Department of Biological Sciences
School of Science
Birkbeck College
University of London

DATABASE API DOCUMENTATION

Description:

The function, db_request, which represents the functionality of the dummy_db_api, takes all database query requests from the BL, and is the only function which is directly available to it. The program has dual search/browse functionality. The function has three arguments.

String searches: The program takes a 'search' data field descriptor from 1-4, a search string, and an 'output' data field/s descriptor from 1-8. Data field descriptors are indicated on the table below. It returns data from requested output field/s for any matching record/s.

Browse requests: The program takes a 'browse' data field descriptor from 1-8, a 'browse data field' request string ‘B’, and an 'additional output' data field/s descriptor. It returns data for all records from stipulated browse fields and, where different, stipulated output field/s.

A variety of data output options is available, depending on search type and string entered.

Some options given in this dummy program will not be available in the final db_api due to excessive size of the output records e.g it will not be possible to browse all DNA sequence records. These are labelled as 'Not available' in the mysql_query document.

Three arguments, db_input, db_query and db_output, should be entered by the BL, such that:

db_result = db_request(db_input, db_query, db_output)

For the purposes of db_input and db_output, which are both integer arguments, the following protocol is used for both string and browse functions:

1 = Genbank accession code

2 = chromosomal location

3 = gene identifier code

4 = protein product name

5 = locus DNA sequence

6 = CDS element positions in the sequence, start codon position & gene span.

7 = 'Summary list' consisting of data types 1-4

8 = 'Full detail list' consisting of data types 1-6

In all cases, and for each record, the Genbank accession code is included separately as a unique identifying tag when data types are returned, to identify all results unambiguously. Note that, in data selected from raw Genbank records, only category 1 (Genbank accession code) is guaranteed to be unique to a single record.

1. Database input

The db_input argument indicates which data category should be searched by a given search term, "db_query". When used with the search term query, db_input can only take values 1-4.

Alternatively, db_input can be used to indicate particular categories of data selected for browsing, where db_input can take values of 1-8.

2. Database query

The db_query is a string argument which may be used for the following purposes:

(i) The db_query string can be used to search the data category specified by db_input for all records containing the string term entered;

(ii) The db_query string can alternatively signal for browsing of the data category indicated by the db_input. In this case the BL submits 'B' as the string argument, which enables browsing.

3. Database output

The db_output is a more flexible argument which selects one or more data types to be returned to the BL for one or more records, as appropriate.

In particular, it allows expanded functionality for string searches, beyond the single data type stipulated by the search_by argument. The BL submits a second integer argument which may be selected from the full range of values shown above i.e. 1-8. This allows multiple data types to be selected for return during a string-based search. It also allows the data category which is returned to be different from the data category used to carry out the search itself.

The browsing feature described earlier may also be expanded upon by appropriate use of the db_output argument. The browsing feature automatically returns the data categories entered for db_input, but, by entering a different data category into db_output, the selection of any two of data categories 1-6 is enabled, which may be useful for delivering results to the user directly, or for processing sets of data in the BL before passing to the user.

Data return and examples

Results are returned as nested lists of strings, as illustrated by the examples below. Where a field is not found for a particular data type, 'NF' will be reported:

Example 1: Unique string search; single data type:

Obtaining a protein name using its unique accession number.

db_request(1,'AY177663',4)

db_result = [['desmoglein 4 preproprotein'], ['AY177663']]

Example 2: Non-unique string search; single data type:

Obtaining gene identification numbers, along with their unique accession numbers, using a gene identifier fragment.

db_request(3,'SSCA',3)

db_result = [['SSCA2', 'SSCA', 'pseudoSSCA'], ['AF485254', 'U49845','AF165912']]

Example 3: Browse; single data type:

Browsing a single data type, chromosome location, which is returned as a string list along with the appropriate accession number string list.

db_request(2,'B',2)

db_result = [[['18q24.3', '18q24', 'NF', '18q11.2', 'q12.1', '18q11'],

['AF485254', 'U49845', 'AF165912', 'U55184', 'U19576', 'U55184']]]

Example 4: Unique string search; multiple data types:

The standard short list of data types is returned for a unique search string.

db_request(4,'desmoglein 4 preproprotein',7)

db_result = [['AY177663'], ['18q24'], ['DSG4'], ['desmoglein 4 preproprotein']]

Example 5: Non-unique string; multiple data types:

A non-unique chromosomal location string returns two candidates, providing the full list of data for both of them.

db_request(2,'18q24.1',8)

db_result = [['D00596', 'U11424'], ['18q24.1', '18q24.1'], ['TPMT', 'pseudoTPMT'],['thiopurine methyltransferase processed', 'pseudo-thiopurine methyltransferase'], ['cacccacat...ataacccaatca', 'tcttagaatat...atatatgacaac'],['99..987,6619..7050','20..125,300..490'],['start=1','start=3'],['234..456','NF']]

Example 6: Browse request; multiple data types:

db_request(5,"B",6)

db_result=[[['cacccacat...tctcacacc','tcttagaag...ttgactaaac','aggactcat...cacatataac','tatatccagg...tgaaccttt'],['AF485254','U49845','AF165912','AJ001716']],[['1799..5187,6619..7050','20..125,300..490','NF', '111..183,8360..8395,9912..10043'], ['start=2', 'start=1','start=3', 'start=1'], ['AF485254', 'U49845', 'AF165912', 'AJ001716']]]

<u>Notes on usage and further examples</u>

When submitting a multiple data category query i.e. category 7 or 8, the use of a non-unique db_query might deliver unexpectedly large quantities of data. Therefore, unless a unique search term is already known - such as a Genbank accession code - it is recommended that a two step search procedure is used. This would consist of two separate calls to the db_query function as follows:

<u>Example 7: Preliminary, non-unique string search; single data type:</u>

(i) A preliminary search where the db_return argument is confined to options 1-4, 6 or 7:

db_request(3,'DSG',4)

db_result = [['desmoglein type 1', 'desmoglein 4 preproprotein'],

['AF485254', 'U49845']]

<u>Example 8: Follow-up, unique string search; multiple data types:</u>

(ii) A follow-up search where a unique db_query (obtained during the initial search) can be submitted, together with a multiple data type request;

db_request(1,'AF485254',8)

db_result = [['AF485254'], ['18q11.2'], ['DSG1'], ['desmoglein type 1'],

['cacccacatatgtcgtgtcgatcacccctactacataggatggatacgataacgatccaccac'],

['99..987,6619..7050', '20..125, 300..490'], ['start=3'], ['234..678']]