SWI-Prolog External Database

Jan Wielemaker
SWI,
University of Amsterdam
The Netherlands
E-mail: jan@swi.psy.uva.nl

June 9, 2000

Abstract

This package realised external storage of Prolog terms based on the *Berkeley DB* library from www.sleepycat.com. The DB library implements modular support for the bottom layers of a database. The database itself maps unconstrained keys onto values. Both key and value are *binary blobs*.

The SWI-Prolog interface for DB allows for fast storage of general Prolog terms in the database.

Contents

1 Introduction

The native Prolog database is not very well suited for either *very* large data-sets or dynamically changing large data-sets that need to be communicated between Prolog instances or need to be safely guarded against system failure. These cases ask for an external database that can be attached quickly and provides protection against system failure.

The Berkeley DB package by SleepyCat software is a GPL'ed library realising the bottom-layers of a database. It is a modular system, which in it's simplest deals with resource management on a mapped file and in its most complex form deals with network transparency, transaction management, locking, recovery, life-backup, etc.

The DB library maps keys to values. Optionally multiple values can be associated with a key. Both key and value are arbitrary-length binary objects.

This package stores arbitrary Prolog terms, using PL_record_external() introduced in SWI-Prolog 3.3.7, in the database. It provides an interface similar to the recorded-database (`recorda/3`). In the future we plan to link this interface transparently to a predicate.

2 The DB interface

2.1 Opening a database

db_open(*+File, +Mode, -DB, +Options***)**
Open a file holding a database. *Mode* is one of `read`, providing read-only access or `update`, providing read/write access. *Options* is a list of options. Currently supported options are:

duplicates(*bool***)**
Do/do not allow for duplicate values on the same key. Default is not to allow for duplicates.

2.2 Manipulating and querying a database

db_put(*+DB, +Key, +Value***)**
Add a new key-value pair to the database. If the database allows for duplicates this will always succeed, unless a system error occurs.

db_del(*+DB, +Key***)**
Delete all key-value pairs with the named *Key* from the database. Note that this is equivalent to `db_delall`(*DB, Key,* _).

db_del(*+DB, ?Key, ?Value***)**
Delete the first matching key-value pair from the database. If the database allows for duplicates, this predicate is non-deterministic. The enumeration performed by this predicate is the same as for `db_get/3`.

db_delall(*+DB, +Key, ?Value***)**
Delete all matching key-value pairs from the database. With unbound key this calls `db_del/2`.

db_get(*+DB, ?Key, -Value*)
Query the database. If the database allows for duplicates this predicate is non-deterministic. If *Key* is unbound, all keys are enumerated. This mode is especially useful to enumerating the entire content of a database.

db_getall(*+DB, +Key, -Value*)
Get all values associated with *Key*. Fails if the key does not exist (as `bagof/3`).

2.3 Initialisation

db_init(*+Options*)
Initialise the DB package. This must be done before the first call to `db_open/4` and at maximum once. If `db_open/4` is called without calling `db_init/1`, default initialisation is used, which is generally suitable for handling small databases without support for advanced features.

Options is a list of options. The currently supported are listed below. For details, please refer to the DB manual.

home(*Home*)
Specify the DB home directory, the directory holding the database files.

config(*+ListOfConfig*)
Specify a list of configuration options, each option is of the form *Name*(*Value*).

mp_size(*+Integer*)
Size of the memory-pool used for caching.

mp_mmapsize(*+Integer*)
Maximum size of a DB file mapped entirely into memory.

3 Installation

3.1 Unix systems

Installation on Unix system uses the commonly found *configure*, *make* and *make install* sequence. SWI-Prolog should be installed before building this package. If SWI-Prolog is not installed as `pl`, the environment variable `PL` must be set to the name of the SWI-Prolog executable. Installation is now accomplished using:

```
% ./configure
% make
% make install
```

This installs the foreign libraries in `$PLBASE/lib/$PLARCH` and the Prolog library files in `$PLBASE/library`, where `$PLBASE` refers to the SWI-Prolog 'home-directory'.

3.2 Acknowledgements