



Adding Transactions to DBM::Deep

Rob Kinyon
rob.kinyon@gmail.com

DBM::Deep is a module written completely in Perl that provides a way to store Perl data structures (scalars, hashes, and arrays) on disk instead of in memory. The data file produced is able to be moved from one machine to another and re-loaded, regardless of the operating system or Perl version. I recently added the ability for DBM::Deep to handle transactions, just database servers can do.

DBM::Deep, originally written by Joseph Huckaby and now maintained by me, is a replacement for Perl's built-in dbmopen (as well as many of the other DBM implementations). It doesn't have the limitations on key size or the number of keys that some other formats have, and it can handle both hashes and arrays as well as complex data structures. Here's the example from its documentation:

```

use DBM::Deep;
my $db = DBM::Deep->new( "foo.db" );

$db->{key} = 'value'; # tie() style
print $db->{key};

# OO style
$db->put('key' => 'value'); print
$db->get('key');

# true multi-level support
$db->{my_complex} = [
    'hello',
    { perl => 'rules' },
    42,
    99,
];

```

The latest version of DBM::Deep, 1.00, I add the ability to rollback operations. I create a DBM::Deep object, call the begin_work method to set up a transaction, and do some work (perhaps inside an eval). Once I've done my work I decide if I want to keep it. If I encountered an error, I call rollback to undo everything I've done. Otherwise, I call commit to make my work permanent.

```

use DBM::Deep;
my $db = DBM::Deep->new( "foo.db" );

$db->begin_work;

eval { ... Do stuff here }

if( $@ ) { $db->rollback; }
else { $db->commit; }

```

Why DBM::Deep?

There are three major reasons to use some sort of disk-based persistence mechanism, whether that's DBM::Deep or something else.

Transparent Persistence

This is the ability to save a set of data structures to disk and retrieve them later without the of the rest program even knowing that the data is persistent. Furthermore, the data structure is stored immediately and not at set marshaling periods.

Huge data structures

Normally, data structures are limited by the size of RAM the server has. DBM::Deep allows for the size a given data structure to be limited by disk instead. Since the data live on the disk, they don't have to be in memory.

Interprocess communication

Storing the data on disk means that other programs can use it as well.

Now, with the release of DBM::Deep 1.00, there is now a fourth reason: software-transactional memory, or STM.

This work was sponsored by a Stonehenge Rock Star Grant. http://www.stonehenge.com/rock_stars/