Perl Virtual Database Module

Bastian Friedrich
Collax GmbH

Edited by
Bastian Friedrich

Perl Virtual Database Module

Edited by Bastian Friedrich and Bastian Friedrich

Revision History

Revision $Revision: 771 $ $Date: 2007-03-21 09:53:33 +0100 (Mi, 21 Mar 2007) $

Table of Contents

Chapter 1. User's Guide

1.1. Overview

The Perl Virtual Database (VDB) provides a virtualization framework for OpenSER's database access. It does not handle a particular database engine itself but lets the user relay database requests to arbitrary Perl functions.

This module cannot be used "out of the box". The user has to supply functionality dedicated to the client module. See below for options.

The module can be used in all current OpenSER modules that need database access. Relaying of insert, update, query and delete operations is supported.

Modules can be configured to use the perlvdb module as database backend using the db_url_parameter:

```
modparam("acc", "db_url", "perlvdb:OpenSER::VDB::Adapter::AccountingSIPtrace")
```

This configuration options tells acc module that it should use the perlvdb module which will in turn use the Perl class OpenSER::VDB::Adapter::AccountingSIPtrace to relay the database requests.

1.2. Dependencies

1.2.1. OpenSER Modules

The following modules must be loaded before this module:

- *perl* -- Perl module

1.2.2. External Libraries or Applications

The following libraries or applications must be installed before running OpenSER with this module loaded:

- *None* (Besides the ones mentioned in the perl module documentation).

1.3. Exported Parameters

None.

1.4. Exported Functions

None.

Chapter 2. Developer's Guide

2.1. Introduction

OpenSER uses a database API for requests of numerous different types of data. Four primary operations are supported:

- query
- insert
- update
- delete

This module relays these database requests to user implemented Perl functions.

2.2. Base class OpenSER::VDB

A client module has to be configured to use the perlvdb module in conjunction with a Perl class to provide the functions. The configured class needs to inherit from the base class `OpenSER::VDB`.

Derived classes have to implement the necessary functions "query", "insert", "update" and/or "delete". The client module specifies the necessary functions. To find out which functions are called from a module, its processes may be evaluated with the `OpenSER::VDB::Adapter::Describe` class which will log incoming requests (without actually providing any real functionality).

While users can directly implement their desired functionality in a class derived from OpenSER::VDB, it is advisable to split the implementation into an Adapter that transforms the relational structured parameters into pure Perl function arguments, and add a virtual table (VTab) to provide the relaying to an underlying technology.

2.3. Data types

Before introducing the higher level concepts of this module, the used datatypes will briefly be explained. The OpenSER Perl library includes some data types that have to be used in this module:

2.3.1. OpenSER::VDB::Value

A value includes a data type flag and a value. Valid data types are DB_INT, DB_DOUBLE, DB_STRING, DB_STR, DB_DATETIME, DB_BLOB, DB_BITMAP. A new variable may be created with

```
my $val = new OpenSER::VDB::Value(DB_STRING, "foobar");
```

Value objects contain the type() and data() methods to get or set the type and data attributes.

2.3.2. OpenSER::VDB::Pair

The Pair class is derived from the Value class and additionally contains a column name (key). A new variable may be created with

```
my $pair = new OpenSER::VDB::Pair("foo", DB_STRING, "bar");
```

where foo is the key and bar is the value. Additonally to the methods of the Value class, it contains a key() method to get or set the key attribute.

2.3.3. OpenSER::VDB::ReqCond

The ReqCond class is used for select condition and is derived from the Pair class. It contains an addtional operator attribute. A new variable may be created with

```
my $cond = new OpenSER::VDB::ReqCond("foo", ">", DB_INT, 5);
```

where foo is the key, "greater" is the operator and 5 is the value to compare. Additonally to the methods of the Pair class, it contains an op() method to get or set the operator attribute.

2.3.4. OpenSER::VDB::Column

This class represents a column definition or database schema. It contains an array for the column names and an array for the column types. Both arrays need to have the same length. A new variable may be created with

```
my @types = { DB_INT, DB_STRING };
my @names = { "id", "vals" };
my $cols = new OpenSER::VDB::Column(\@types, \@names);
```

The class contains the methods type() and name() to get or set the type and name arrays.

2.3.5. OpenSER::VDB::Result

The Result class represents a query result. It contains a schema (class Column) and an array of rows, where each row is an array of Values. The object methods coldefs() and rows() may be used to get and set the object attributes.

2.4. Adapters

Adapters should be used to turn the relational structured database request into pure Perl function arguments. The alias_db function alias_db_lookup for example takes a user/host pair, and turns it into another user/host pair. The Alias adapter turns the ReqCond array into two separate scalars that are used as parameters for a VTab call.

Adapter classes have to inherit from the OpenSER::VDB base class and may provide one or more functions with the names insert, update, replace, query and/or delete, depending on the module which is to be used with the adapter. While modules such as alias_db only require a query function, others -- such as siptrace -- depend on inserts only.

2.4.1. Function parameters

The implemented functions need to deal with the correct data types. The parameter and return types are listed in this section.

insert() is passed an array of OpenSER::VDB::Pair objects. It should return an integer value.

replace() is passed an array of OpenSER::VDB::Pair objects. This function is currently not used by any publicly available modules. It should return an integer value.

delete() is passed an array of OpenSER::VDB::ReqCond objects. It should return an integer value.

update() is passed an array of OpenSER::VDB::ReqCond objects (which rows to update) and an array of OpenSER::VDB::Pair objects (new data). It should return an integer value.

query() is passed an array of OpenSER::VDB::ReqCond objects (which rows to select), an array of strings (which column names to return) and a single string by which column to sort. It should return an object of type OpenSER::VDB::Result.

2.5. VTabs

VTabs (virtual tables) provide a particular implementation for an adapter. The Alias adapter e.g. calls a function with two parameters (user, host) and expects a hash to be returned with the two elements username and domain, or undef (when no result is found). A sample VTab implementation for the Alias adapter demonstrates this technique with a Perl hash that contains the alias data.

The standard Adapter/VTab pattern lets the user choose between three options on how to implement VTabs:

- *Single function*. When a function is used as a virtual table, it is passed the operation name (insert, replace, update, query, delete) as its first parameter. The function may be implemented in the main namespace.

- *Package/class*. The defined class needs to have an init() function. It will be called during the first call of that VTab. Addtionally, the package has to define the necessary functions insert, replace, update, delete and/or query. These functions will be called in a function context (first parameter is the class name).

- *Object*. The defined class needs to have a new() function which will return a reference to the newly created object. This object needs to define the necessary functions insert, replace, update, delete and/or query. These functions will be called in a method context (first parameter is a reference to the object).

Chapter 3. Frequently Asked Questions

1. Where can I find more about OpenSER?

Take a look at http://openser.org/.

2. Where can I post a question about this module?

First at all check if your question was already answered on one of our mailing lists:

- User Mailing List - http://openser.org/cgi-bin/mailman/listinfo/users
- Developer Mailing List - http://openser.org/cgi-bin/mailman/listinfo/devel

E-mails regarding any stable OpenSER release should be sent to <`users@openser.org`> and e-mails regarding development versions should be sent to <`devel@openser.org`>.

If you want to keep the mail private, send it to <`team@openser.org`>.

3. How can I report a bug?

Please follow the guidelines provided at: http://sourceforge.net/tracker/?group_id=139143.