

Package ‘RObjectTables’

December 4, 2001

Version 0.1

Date 2001/12/04

Title User-level attach()’able table support

Author Duncan Temple Lang <duncan@research.bell-labs.com>

Depends R (>= 1.4.0)

Maintainer Duncan Temple Lang <duncan@research.bell-labs.com>

Description The C and S code allows one to define R objects to be used as elements of the search path with their own semantics and facilities for reading and writing variables. The objects implement a simple interface via R functions (either methods or closures) and can access external data, e.g. in other applications, languages, formats, ...

License GPL

URL <http://www.omegahat.org/RObjectTables>, <http://www.omegahat.org>
<http://www.omegahat.org/bugs>

R topics documented:

UserDatabase	1
dbmethods	2

Index	4
--------------	----------

UserDatabase	<i>Create user-defined attach()’able table</i>
---------------------	--

Description

These functions convert a user-level object into an R object that can be attached as an element of the R search path. The `newRFunctionTable` works on objects that have methods for the `dbexists`, `dbread`, ... functions.

`newRClosureTable` works on a collection of functions (usually sharing state with a common environment) that are called directly by the C-level interface between the R engine and the user-level table.

Usage

```
newRFunctionTable(db)
newRClosureTable(db)
```

Details

Value

An object of class `UserDefinedTable` that is an external pointer to a C-level object that represents the R table. This object can then be used in a call to [attach](#).

Note

This interface is experimental. Please ensure that important data is saved before using this.

See Also

[attach](http://developer.r-project.org/RTableObjects.pdf) <http://developer.r-project.org/RTableObjects.pdf>

dbmethods

Methods for user-defined tables

Description

These are generic functions that are extended by different classes of user-level tables that can be attached to the search path. They are called when the corresponding user-level functions are called for that ‘database’. A classes implementations of these methods must be globally accessible so that they can be called when needed. This differs from closure tables which pass functions, (rather than function names) to the C-level interface that implements the table’s connection to the R engine.

Usage

```
dbobjects(database)
dbexists(database, name)
dbread(database, name)
dbwrite(database, name, object)
dbremove(database, name)
dbattach(database)
dbdetach(database)
dbcache(database, name)

dbobjects.default(database)
dbexists.default(database, name)
dbread.default(database, name)
dbremove.default(database, name)
dbwrite.default(database, name, object)
dbattach.default(database)
dbdetach.default(database)
dbcache.default(database, name)
```

Details

These methods are the S4-compatible accessors for user-level tables that can be attached to the search path. They correspond to the `exists`, `get`, `remove`, `assign` and `objects` that are used to access and operate on variables within elements of the search path. These are not typically called directly but by the R engine when accessing user-level tables that are implemented by particular methods for these generic functions.

These functions are compatible with the equivalent S4 functions.

Value

`dbexists` returns a logical value indicating whether the database has a variable by that name.

`dbread` is equivalent to `get` and returns the value in the database assigned to the specified name.

`dbwrite` is equivalent to `assign` and returns the value being assigned, i.e. `object`. This allows one to do chained assignments of the form `x <- y <- 10`.

`dbremove` is equivalent to `remove` and removes the binding for the specified name in the database, discarding the value.

`dbobjects` is equivalent to `objects` and returns a character vector containing the names of all the name-value bindings in the database.

`dbcannotCache` returns a logical value indicating whether the value of the specified variable (given by `name`) cannot be changed except for in R (`TRUE`) or whether it might be changed externally (`FALSE`). This is used by the R engine to determine if it is entitled to cache the value associated with `name`. It does this to avoid searching through the list of elements in the search path each time it wants the value of a variable that it has already seen. This is useful when the data source can be modified externally by other applications such as another thread in a Java application, the CORBA naming service, etc.

`dbattach` and `dbdetach` have no (useful) return values and are invoked each time the user-level table is added and removed from the search path, respectively. These can be used to perform initialization and cleanup of values that the database uses to implement the other methods. For example, it might create a directory for caching values when it is attached and remove it on detach. Alternatively, it might open a connection to a database and close it when it is no longer needed.

Note

This is experimental. Make certain that important data is backed up before using this user-level table interface.

See Also

`newRFunctionTable` `newRClosureTable` `attach` `detach` <http://developer.r-project.org/RObjectTables.pdf>

Index

*Topic **data**

dbmethods, [2](#)

 UserDatabase, [1](#)

assign, [3](#)

attach, [2](#), [3](#)

dbattach, [3](#)

dbattach (*dbmethods*), [2](#)

dbdetach, [3](#)

dbdetach (*dbmethods*), [2](#)

dbexists, [1](#)

dbexists (*dbmethods*), [2](#)

dbmethods, [2](#)

dbobjects (*dbmethods*), [2](#)

dbread, [1](#)

dbread (*dbmethods*), [2](#)

dbwrite (*dbmethods*), [2](#)

detach, [3](#)

exists, [3](#)

get, [3](#)

newRClosureTable, [3](#)

newRClosureTable (*UserDatabase*), [1](#)

newRFunctionTable, [3](#)

newRFunctionTable (*UserDatabase*), [1](#)

objects, [3](#)

remove, [3](#)

UserDatabase, [1](#)