

Connecting and Managing Databases with **aRT**

Pedro Ribeiro de Andrade Neto
Paulo Justiniano Ribeiro Junior

December 07, 2007

Contents

1	Introduction	1
2	Connections	1
3	Managing users from aRT	4

1 Introduction

aRT is a package for manipulating spatial data using the library **TerraLib**. **TerraLib** manipulates data stored in DataBase Management System (DBMS), to which it is necessary establish connection. Connecting to a DBMS is always the first action when using **aRT**, and this short document introduces the primary class of the package, **aRTconn**, that implements a virtual connection to a DBMS. Initially in this document we consider the user already has permissions in the DBMS. Later in Section 3 we explain how **aRT** functions can help adding users to the DBMS.

```
> require(aRT)
```

```
-----  
R-TERRALIB API  
http://www.leg.ufpr.br/aRT  
TerraLib version 3.2.0 is now loaded  
aRT version 1.5-1 (2008-04-09) is now loaded  
-----
```

```
> aRTsilent(FALSE)
```

```
[1] FALSE
```

2 Connections

After loading the package, it is necessary to establish a DBMS connection and the function `openConn()` is designed for such task. Currently, this function takes four arguments, with the defaults indicated within the parenthesis: `user` (the current user as given by `$USER`), `password` (no password), `host` ('localhost') and `port` (3306). Future versions should include a fifth argument, `DBMS`, with options to supported BDMS such as "MySQL", "Postgres", "PostGIS".

```
> con = openConn(user = "root", host = "localhost", pass = "")
> con
```

If a connection is successfully established this function returns an object of the class `aRTconn`. It is important to notice the elements in the object `con` cannot be changed. The only possible way to set/change the connection parameters is creating another object again, calling `openConn()` with the new options. This is due to the fact data is stored in an external pointer. If the connection cannot be established, the function stops with an error, as in the following example.

```
> err = try(con2 <- openConn(user = "root", pass = "abc321"))

Trying to connect ... no

> strsplit(err[1], " : ")

[[1]]
[1] "Error in .local(.Object, ...)"
[2] "\n Access denied for user 'root'@'localhost' (using password: YES)\n"
```

An `aRTconn` object stores a *virtual* connection, i.e., every time a database access is required the object connects with the DBMS, performs the task, and then disconnects.

An `aRTconn` object allows for some basic queries and operations in the DBMS. The function `showDbs()` lists the databases available (the ones which the user has some permission), returning a character vector.

```
> showDbs(con)

[1] "information_schema" "auckland"      "bodmin"
[4] "ca20"               "catarina"      "dynatt"
[7] "image"              "meso"          "mysql"
[10] "parana"             "pol3"          "recife"
[13] "saudavel"           "sp"            "tabletest"
```

New databases can be created with `createDb()`:

```
> dbintro = createDb(con, "intro")
```

```

Creating database 'intro' ... yes
Creating conceptual model of database 'intro' ... yes
yes
Loading layer set of database 'intro' ... yes
Loading 'root' view set of database 'intro' ... yes

```

```
> dbintro
```

```
Object of class aRTdb
```

```

Database: "intro"
Layers: (none)
Themes: (none)
External tables: (none)

```

and existing databases can be opened with `openDb()`,

```
> db = openDb(con, "intro", update = TRUE)
```

```

Connecting with database 'intro' ... yes
Loading layer set of database 'intro' ... yes
Loading 'root' view set of database 'intro' ... yes

```

```
> db
```

```
Object of class aRTdb
```

```

Database: "intro"
Layers: (none)
Themes: (none)
External tables: (none)

```

In these examples, both objects `dbintro` and `db` belong to the class `aRTdb`, storing a real connection to a particular database. As a consequence, at this point the virtual connection is no longer needed and these objects are independent from `conn`.

Note that an `aRTdb` object can turn inconsistent if the database is removed from the DBMS, possibly generating a *core dump*. This cannot be avoided because the database can be removed from any other connection to the DBMS, which can be from another `aRTconn` object, directly from MySQL, or by another TerraLib-based application. Before removing a database, if there are any connection to it, we recommended to remove the database connection from memory using `gc()`.

```

> rm(db)
> rm(dbintro)
> invisible(gc())

```

Databases can be removed provided the user has necessary permissions. Notice the argument `force=TRUE` is used to avoid a keyboard confirmation, because this a dangerous operation which cannot be undone.

```
> if (any(showDbs(con) == "intro")) deleteDb(con, "intro",  
+      force = TRUE)
```

```
Checking for database 'intro' ... yes
```

```
Deleting database 'intro' ... yes
```

3 Managing users from aRT

The function `addPermission()` provides a way to add users to the DBMS, with corresponding permissions. The first step is to start a session with *root* permissions. Using this connection, some types of permissions can be set¹.

```
> con = openConn(u = "root")
```

- To create an user with access from localhost, without or with password use, respectively:

```
> addPermission(con, "elias")  
> addPermission(con, "elias", pass = "password")
```

- To create an user with access from a specific host, specified either by an IP number or hostname, without or with password use, respectively:

```
> addPermission(con, "elias", host = "est.ufmg.br")  
> addPermission(con, "elias", host = "est.ufmg.br", pass = "senha")
```

- Finally, to create an user from any host the password is compulsory:

```
> addPermission(con, "elias", remote = TRUE, pass = "senha")
```

This function, when called as before, provides **full access** to all the DBMS databases. It is also possible to restrict the permissions to some privileges or some specific databases. For instance:

```
> addPermission(con, "elias", db = "citrus")  
> addPermission(con, "elias", privilege = "select", db = "saudavel")  
> addPermission(con, "elias", privilege = "update", db = "parana")
```

In order to see the permissions of the DBMS, use `getPermissions`:

```
> getPermissions(con)[1:5, ]
```

¹Henceforth, the R code will not be executed for security reasons.

	host	user	password	select	insert	update
1	localhost	root	Yes	Yes	Yes	Yes
2	guaja	root	No	Yes	Yes	Yes
3	localhost	debian-sys-maint	Yes	Yes	Yes	Yes
4	localhost	pedro	No	Yes	Yes	Yes
5	pataxo.est.ufpr.br	saudavel	No	Yes	Yes	Yes
	delete	create				
1	Yes	Yes				
2	Yes	Yes				
3	Yes	Yes				
4	Yes	Yes				
5	Yes	Yes				

Finally, for removing the permissions of a given user, use `dropUser`:

```
> dropUser(con, user = "elias")
> dropUser(con, user = "elias", remote = TRUE)
> dropUser(con, user = "elias", host = "pataxo.est.ufpr.br")
```

References

- [1] Chambers, J.M., 1998, Programming with data, a guide to the S language. Springer, New York.
- [2] MySQL Documentation. <http://dev.mysql.com/doc/>. Last access: 08/14/2006