Connecting and Managing Databases with aRT

Pedro Ribeiro de Andrade Neto
Paulo Justiniano Ribeiro Júnior

November 28, 2005

Contents

1 Introduction

aRT is a package for manipulating spatial data using TerraLib. As TerraLib manipulates data stored in databases, we need to establish connections to them. This short document introduces the primary class of the package, aRTconn, that implements a virtual connection to a DBMS.

```
> library(aRT)

Loading required package: sp

------------------------------------------------
R-TERRALIB API
aRT version 0.4-15 (2005-12-20) is now loaded
------------------------------------------------
```

2 Connections

After load the package, we need to establish a DBMS connection, and we can do it calling `openConn()`. This function takes four arguments: `user`, `password`, `host` and `port`, and their default values are variable USER in the environment, empty string, "localhost" and 3306, respectively[1]. For example:

```
> conn <- openConn(user = "root", host = "localhost",
+     pass = "")
```

[1] In the future, there will be a fifth argument, `DBMS`, with values "MySQL", "Postgres", "PostGIS", etc.

```
Trying to connect ... yes
```

This function returns an `aRTconn` object if successful. Once the object `conn` is created, the variables it contains cannot be changed. If you need to set them, the only way is to create another object. It happens because data is stored in an external pointer.

```
> conn
```

```
Object of class aRTconn

DBMS: "MySQL"
User: "root"
Password: ""
Port: 3306
Host: "localhost"
```

If the connection cannot be established, the function stops with an error:

```
> err = try(conn2 <- openConn(user = "root", pass = "abc321"))
```

```
Trying to connect ... no
```

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

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

One `aRTconn` object stores a *virtual* connection, i.e., all time that a database access is required, it connects, does some stuff, and then disconnects. For example, if it is the first time you are using `aRT`, perhaps you will need to add permissions to some users. To do this, use `addUser()`:

```
> addUser(conn, "pedro")
```

```
Adding user 'pedro' ... yes
```

This function gives *all* permissions in *all* databases to a user, and only root can do that. If you want to do something different, you will need to run mysql for yourself, and use the *grant* command [2].

With an `aRTconn` object, you can also see the available databases (the ones which the user has permission):

```
> showDbs(conn)
```

```
 [1] "Parana"        "auck"          "bodmin"        "ca20"
 [5] "catarina"      "citrusTeste10" "dbsjc"         "dummy"
 [9] "dynatt"        "event"         "intro"         "leg"
[13] "meso"          "mysql"         "parana"        "pol3"
[17] "pr"            "recife"        "saudavel"      "sp"
[21] "t4"            "tabletest"     "test"          "testeparana"
[25] "tyr"
```

and then you can remove databases using this object:

```
> if (any(showDbs(conn) == "intro")) deleteDb(conn, "intro",
+     force = T)
```

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

The argument `force=TRUE` is used to avoid a keyboard confirmation, because it is a dangerous operation and cannot be undone.

Databases can be opened with `openDb()`:

```
> db = openDb(conn, showDbs(conn)[1])
```

```
Connecting with database 'Parana' ... yes
Loading layer set of database 'Parana' ... yes
Loading view set of database 'Parana' ... yes
```

or created, using `createDb()`:

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

```
Connecting with database 'intro' ... no
Creating database 'intro' ... yes
Creating conceptual model of database 'intro' ... yes
Loading layer set of database 'intro' ... yes
Loading view set of database 'intro' ... yes
```

Both objects belong to class `aRTdb`, and store a real connection to a database. Therefore these objects do not need the virtual connection anymore.

Note that an `aRTdb` object can become inconsistent if the database is removed from the DBMS, and it can generate a core dump. It cannot be avoided, because the database can be removed from anywhere, since another `aRTconn` object, directly from MySQL, or by another TerraLib-based program. The better solution is to remove the database connection from memory before delete it.

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

```
Removing aRTdb 'intro' from memory ... yes
Removing aRTdb 'Parana' from memory ... yes
```

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: 08/14/2005