R to Oracle Database Connectivity: Use ROracle for both Performance and Scalability

By Mark Hornick-Oracle on [Jun 12, 2013](https://blogs.oracle.com/R/entry/r_to_oracle_database_connectivity)

https://blogs.oracle.com/R/entry/r\_to\_oracle\_database\_connectivity

R users have a few choices of how to connect to their Oracle Database. The most commonly seen include: RODBC, RJDBC, and ROracle. However, these three packages have significantly different performance and scalability characteristics which can greatly impact your application development. In this blog, we’ll discuss these options and highlight performance benchmark results on a wide range of data sets.

*If you use ROracle, we'd like to hear about your experience. Please take this*[*brief survey*](https://www.surveymonkey.com/s/YKLLDHZ)*.*

By way of introduction, RODBC is an R package that implements ODBC database connectivity. There are two groups of functions: the largely internal odbc\* functions implement low-level access to the corresponding ODBC functions having a similar name, and the higher level sql\* functions that support read, save, copy, and manipulation of data between R data.frame objects and database tables. Here is an example using RODBC:

library(RODBC)

con <- odbcConnect("DD1", uid="rquser", pwd="rquser", rows\_at\_time = 500)

sqlSave(con, test\_table, "TEST\_TABLE")

sqlQuery(con, "select count(\*) from TEST\_TABLE")

d <- sqlQuery(con, "select \* from TEST\_TABLE")

close(con)

The R package RJDBC is an implementation of the R DBI package – database interface – that uses JDBC as the back-end connection to the database. Any database that supports a JDBC driver can be used in connection with RJDBC. Here is an example using RJDBC:

library(RJDBC)

drv <- JDBC("oracle.jdbc.OracleDriver",

classPath="…tklocal/instantclient\_11\_2/ojdbc5.jar", " ")  
con <- dbConnect(drv, " jdbc:oracle:thin:@myHost:1521:db", "rquser", "rquser")

dbWriteTable(con, "TEST\_TABLE", test\_table)

dbGetQuery(con, "select count(\*) from TEST\_TABLE")

d <- dbReadTable(con, "TEST\_TABLE")  
dbDisconnect(con)

The ROracle package is an implementation of the R DBI package that uses Oracle OCI for high performance and scalability with Oracle Databases. It requires Oracle Instant Client or Oracle Database Client to be installed on the client machine. Here is an example using ROracle:

library(ROracle)

drv <- dbDriver("Oracle")

con <- dbConnect(drv, "rquser", "rquser")

dbWriteTable(con,”TEST\_TABLE”, test\_table)

dbGetQuery(con, "select count(\*) from TEST\_TABLE")

d <- dbReadTable(con, "TEST\_TABLE")

dbDisconnect(con)

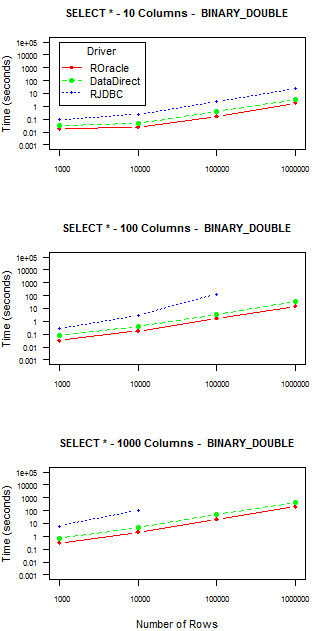
Notice that since both RJDBC and ROracle implement the DBI interface, their code is the same except for the driver and connection details.

To compare these interfaces, we prepared tests along several dimensions:

* Number of rows – 1K, 10K, 100K, and 1M
* Number of columns – 10, 100, 1000
* Data types – NUMBER, BINARY\_DOUBLE, TIMESTAMP, and VARCHAR; Numeric data is randomly generated, all character data is 10 characters long.
* Interface: RODBC 1.3-6 (with Data Direct 7.0 driver), RJDBC 0.2-1 (with rJava 0.9-4 with increased memory limit in JRIBootstrap.java),   
  and ROracle 1.1-10 (with Oracle Database Client 11.2.0.4)
* Types of operations: select \*, create table, connect

**Loading database data to an R data.frame**

Where an in-database capability as provided by Oracle R Enterprise is not available, typical usage is to pull data to the R client for subsequent processing. In Figure 1, we compare the execution time to *pull* 10, 100, and 1000 columns of data from 1K, 10, 100K, and 1M rows for BINARY\_DOUBLE data on a log-log scale. Notice that RJDBC does not scale to 100 columns x 1M rows, or above 1000 cols x 100K records. While RODBC and ROracle both scale to these volumes, ROracle is consistently faster than RODBC: up to 2.5X faster. For RJDBC, ROracle is up to 79X faster.

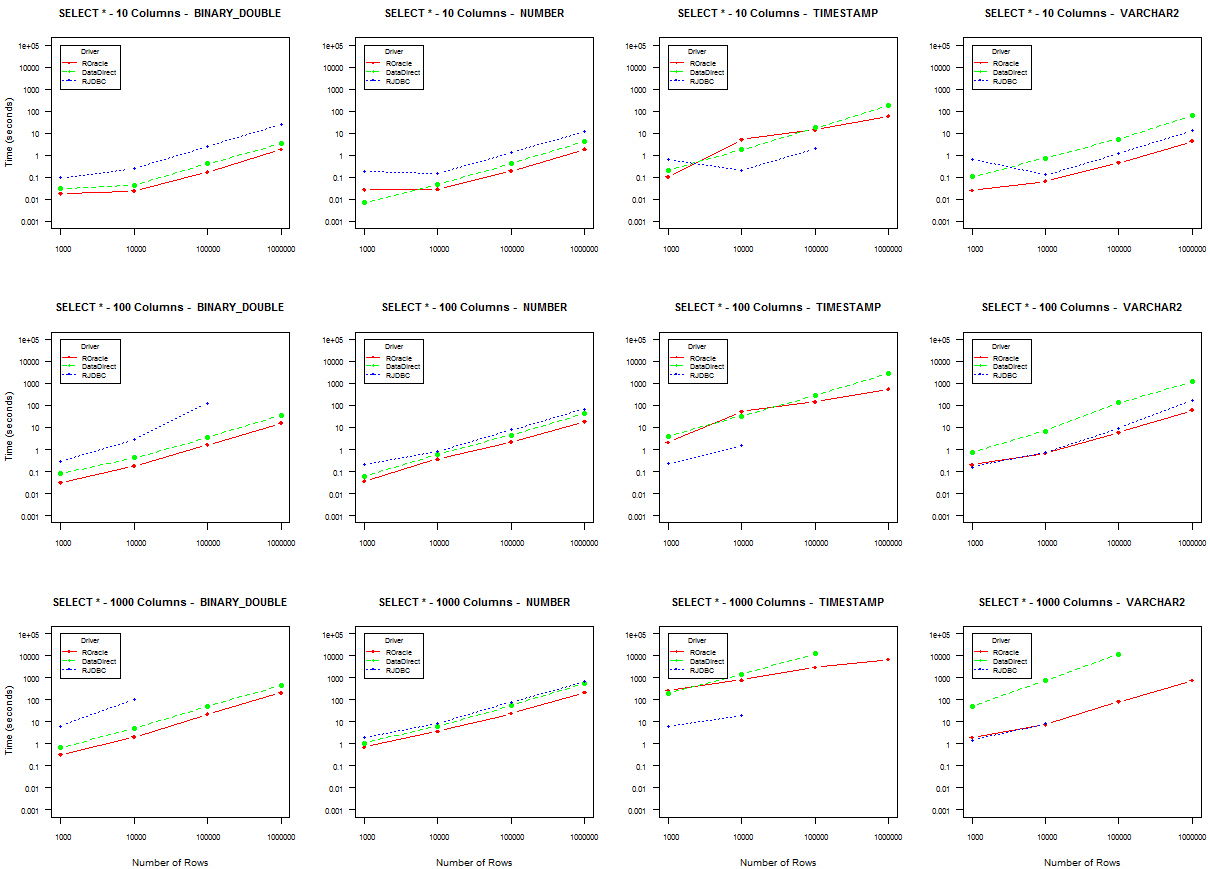


**Figure 1: Comparison of RJDBC, RODBC, and ROracle for BINARY\_DOUBLE for Select \***

In Figure 2, we provide the range of results for RODBC, ROracle, and RJDBC across all data types. Notice that only ROracle provides the full range of scalability while providing superior performance in general.

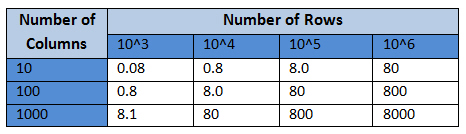
ROracle is virtually always faster than RODBC: NUMBER data up to 2.5X faster, VARCHAR2 data up to 142X faster, and time stamp data up to 214X faster. RODBC fails to process 1000 columns at 1M rows.

For RJDBC, ROracle is up to 13X faster on NUMBER data, 79X faster on binary double data, 3X for VARCHAR2 data (excluding the 25X over the smallest data set). Note that RODBC and RJDBC have a limit of 255 characters on the length the VARCHAR2 columns. TIMESTAMP data is the one area where RJDBC initially shines, but then fails to scale to larger data sets.



**Figure 2: Comparing the three interfaces for select \* from <table>**

Data set sizes represented in megabytes are captured in Table 1 for all data types. With only minor variation, the data sizes are the same across data types.

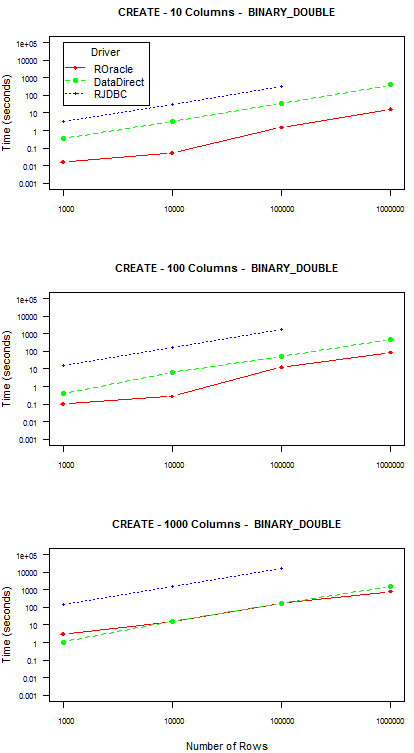


**Table 1: Dataset sizes in megabytes**

**Creating database tables from an R data.frame**

Data or results created in R may need to be written to a database table. In Figure 3, we compare the execution time to create tables with 10, 100, and 1000 columns of data with 1K, 10, 100K, and 1M rows for BINARY\_DOUBLE. Notice that in all three cases, RJDBC is slowest and does not scale. RJDBC does not support the NUMBER or BINARY\_DOUBLE data types, but uses FLOAT(126) instead. ROracle scaled across the remaining data types, while RODBC and RJDBC were not tested.

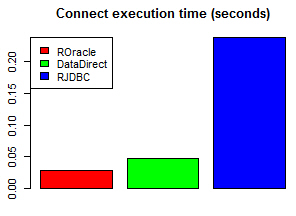
ROracle is 61faster than RODBC for 10 columns x 10K rows, with a median of 5X faster across all data sets. ROracle is 630X faster on 10 columns x 10K rows, with a median of 135X faster across all data sets. RJDBC did not scale to the 1M row data sets.



**Figure 3:** **Comparison of RJDBC, RODBC, and ROracle for BINARY\_DOUBLE create table**

**Connecting to Oracle Database**

Depending on the application any sub-second response time may be sufficient. However, as depicted in Figure 4, ROracle introduces minimal time to establish a database connection. ROracle is nearly 10X faster than RJDBC and 1.6X faster than RODBC.



**Figure 4: Database connection times for ROracle, RODBC, and RJDBC**

In summary, for maximal performance and scalability, ROracle can support a wide range of application needs. RJDBC has significant limitations in both performance and scalability. RODBC can be more difficult to configure on various platforms and while it largely scales to the datasets tested here, its performance lags behind ROracle.

***If you use ROracle, we'd like to hear about your experience. Please take this***[***brief survey***](https://www.surveymonkey.com/s/YKLLDHZ)***.***

*All tests were performed on a 16 processor machine with 4 core Intel Xeon E5540 CPUs @ 2.53 GHz and 74 GB RAM. Oracle Database was version 11.2.0.4. For JDBC, the following was modified before installing rJava.*

*rJava\_0.9-4.tar.gz\rJava\_0.9-4.tar\rJava\jri\bootstrap\JRIBootstrap.java was modified to use 2GB :*

try {

System.out.println(jl.toString()+" -cp "+System.getProperty("java.class.path")+" -Xmx2g -Dstage=2 Boot");

Process p = Runtime.getRuntime().exec(new String[] {

jl.toString(), "-cp", System.getProperty("java.class.path"),"-Xmx2g", "-Dstage=2", "Boot" });

System.out.println("Started stage 2 ("+p+"), waiting for it to finish...");

System.exit(p.waitFor());

} catch (Exception re) {}