**The installation of the ROracle driver/package on Windows 10**

By Joseph DeArce

This document describes the necessary steps for the installation of **ROracle** driver/package on Windows 10 using several software packages. You might ask why I should go through the trouble of installing these complicated software products, my answer is simple **SPEED** this connection is faster than ODBC, JDBC and anything else because it’s a native connection. If you’re interested in processing large amounts of data and they reside in RDBM databases, then ROracle is for you.

In benchmark comparisons, **ROracle** performed up to 79 times faster than **RJDBC** and 2.5 times faster than **RODBC** for reading data across a range of 1000 to 1 million rows, and 10 to 1000 columns. **ROracle** shows scalability across **NUMBER**, **VARCHAR2**, **TIMESTAMP**, and **BINARY\_DOUBLE** data types.

**Hardware**

Before we begin our installation the test system has the following hardware, an Acer Laptop running Windows 10 with a Core7i quad processor with 16GB of memory and a 1TB hard drive. Installed is Oracle 12c Enterprise database with a SGA of 4GB and with a custom configuration. In addition, there are six PDB’s on the Oracle 12c R1 instance.

**Prerequisite**

There are a number of prerequisites for the installation of ROracle, see the list below:

1. First step is to install the R language software from the main R software website called CRANS, see the URL below. This site also has tutorials documentation, books, FAQs, and other resources

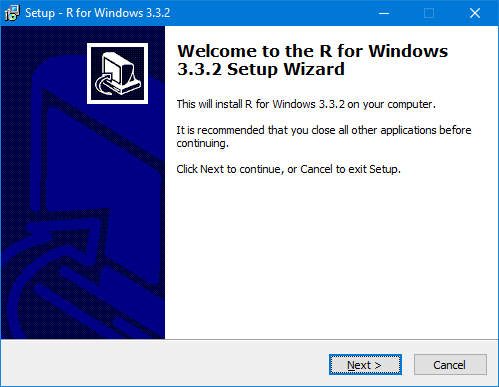
<https://www.r-project.org/>

Download page

<https://cran.r-project.org/bin/windows/base/>

On the main download page there are many resources for novices, FAQs, installation instructions, a compatibility list, instructions on how to update packages and New Features. On other pages there are manuals and tutorials. The current version is 3.2.2 when you have downloaded the file double-click on the file.

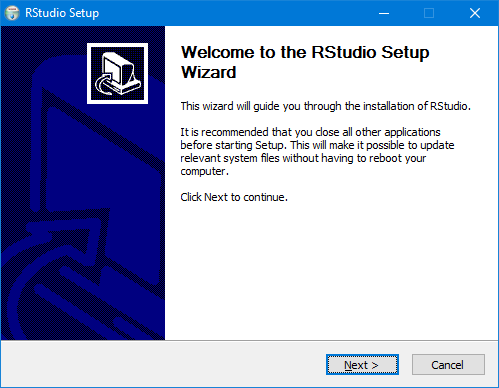
Follow the installation instruction, that you will see on the screen below.



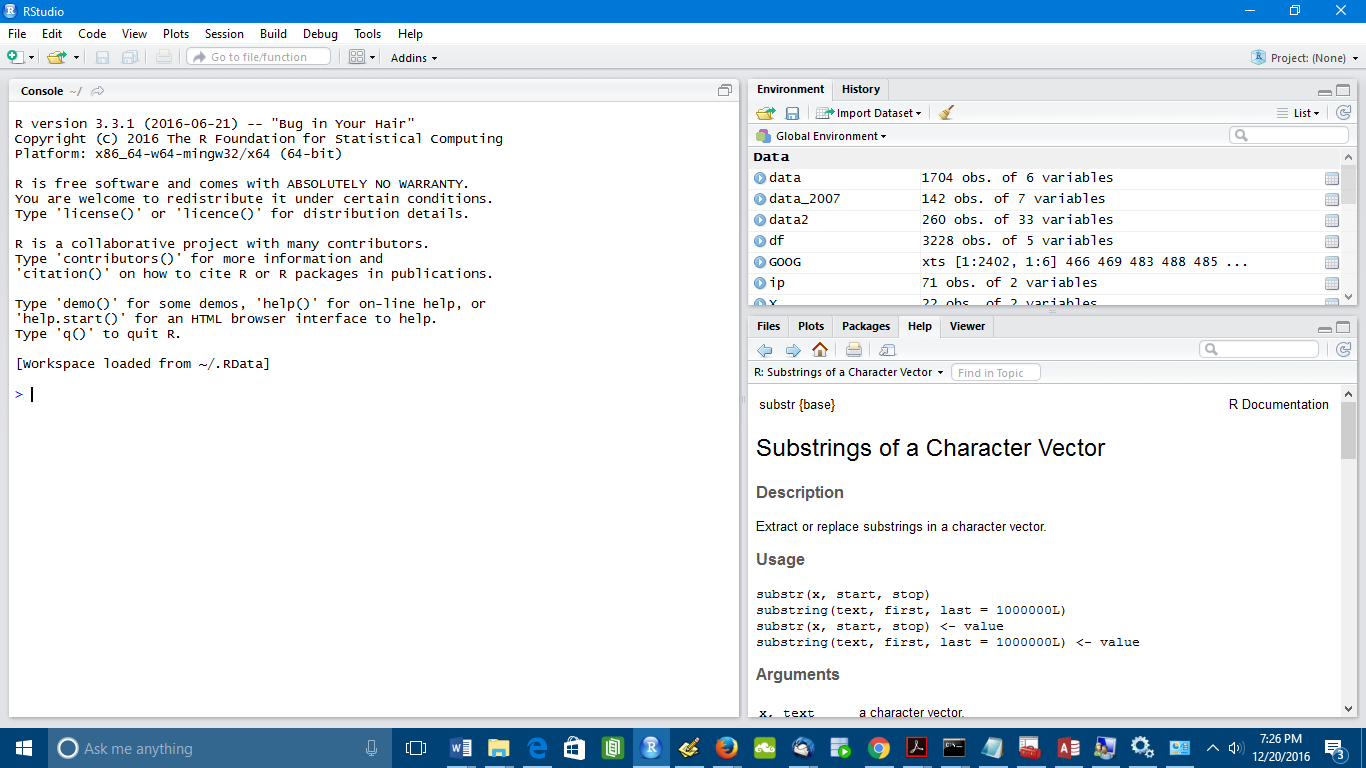
1. We will have to install an IDE for R we will do this to make programing in R easier and more intuitive. There are several IDEs to choose from, and a simple Google search will locate them for you. I picked RStudio which is one of the better known one’s. The RStudio can be installed from the URL below.

<https://www.rstudio.com/products/rstudio/download3/>

Click on the download link and wait until the file is downloaded to your machine then double-click on it, you will see the screen below then follow the instructions of the installer. The most current version I found was “**RStudio 1.0.44 - Windows Vista/7/8/10**”.



When the installer has finished and RStudio is installed, select it and bring it up. When RStudio comes up you will see the main screen for the program, see below.



The programming environment is divided into three panels the main programming panel takes up half the window the two additional panels show the environment and history. The other has several tabs but opens up to the documentation page, but also plots and files will be displayed here.

1. **Installing Oracle Client 12c**

In addition, you will need to install the Oracle client and create several environmental variables and add several configuration file to the install directory. The client software you can get from the main Oracle database page. Here you will be asked to agree to the Oracle OTN license which you must agree to, to be able to download the software.

You should also sign up for an OTN account since it is free. This site can be very helpful for getting timely information on Oracle developments for all of their products. Also Oracle has a large number of blogs by their development teams. These blogs have a greate deal of information and insights on trends in development.

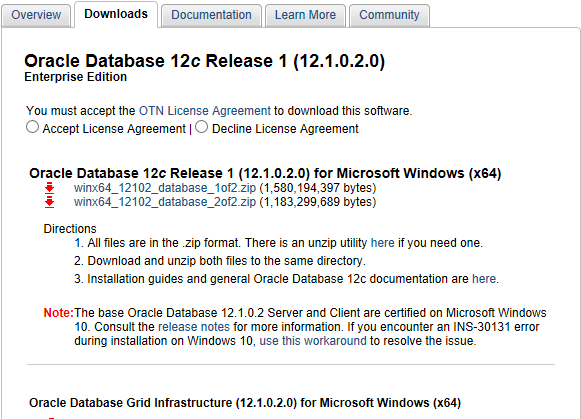
First step is to download the Oracle client from the main database website, use the URL below.

<http://www.oracle.com/technetwork/database/enterprise-edition/overview/index.html>

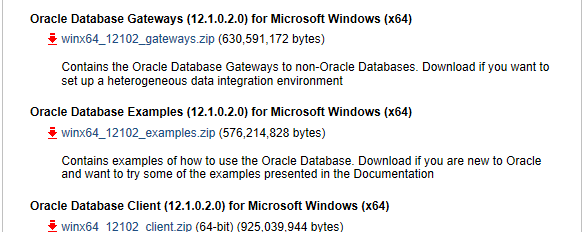


Click on the downloads tab and agree to the OTN license, then select the file you wish to download, in our case it’s the Oracle client for 12c

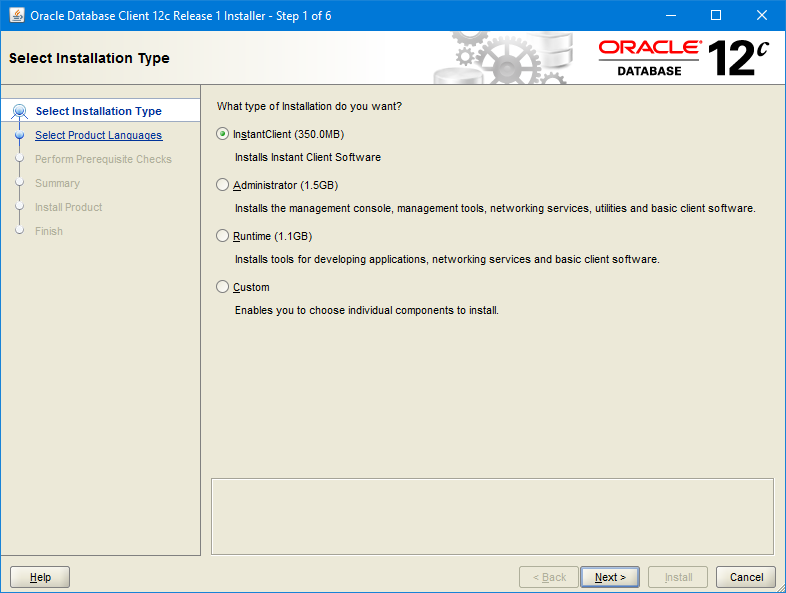
Select the’**Oracle Database 12c Release 1 (12.1.0.2)**‘ and then click the ‘See All’ hyperlink.



Click on the ‘**Oracle Database Client (12.1.0.2) for Windows (64x)**‘ and wait until the file is downloaded.

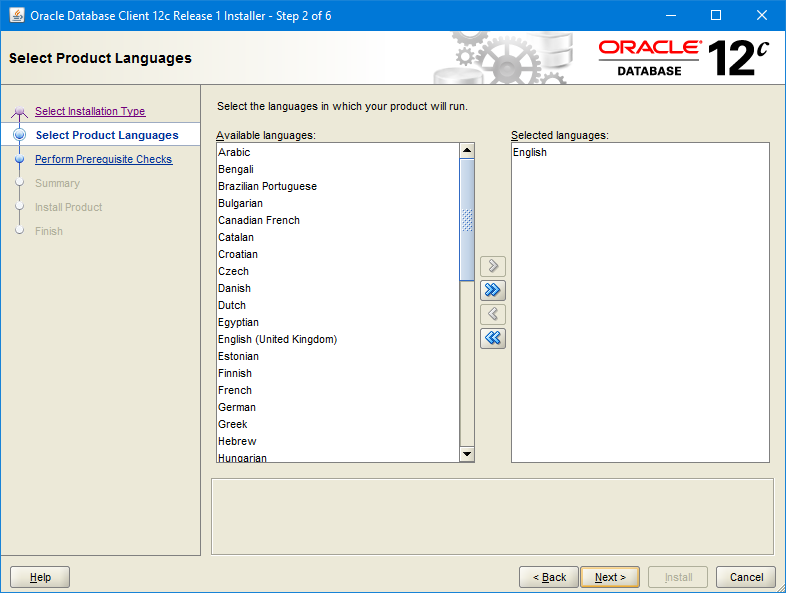


Download the file to your temporary working directory and unzip the software. Double-click on the setup.exe file to begin the install.

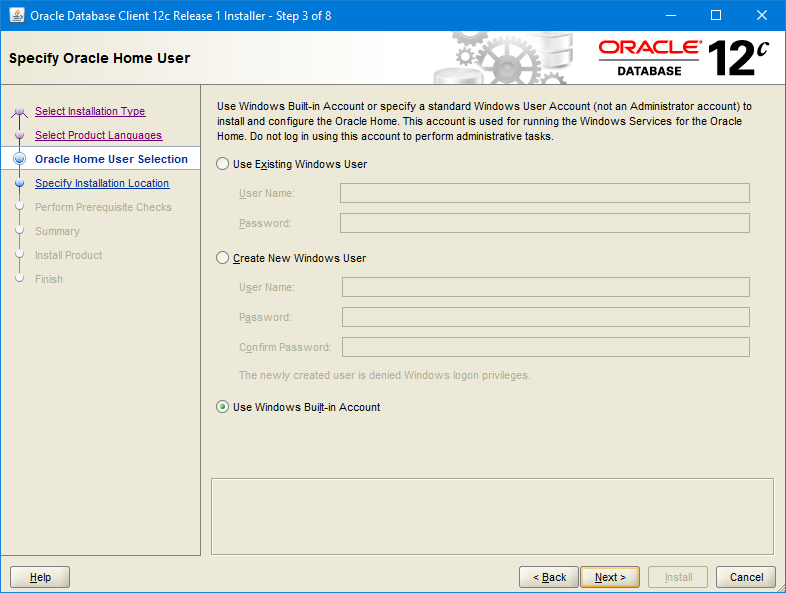


Select your install option, I choose the ‘**Administrator**‘ option, this option will lead to less steps being done than using the ‘**InstantClient**’ option to install and configure the ROracle software. The instant Client option requires more steps and a more extensive configuration.

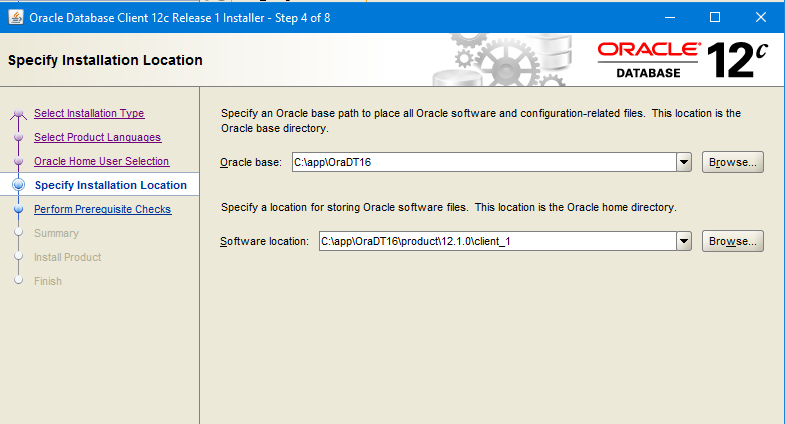
Chose a language and click **Next.**



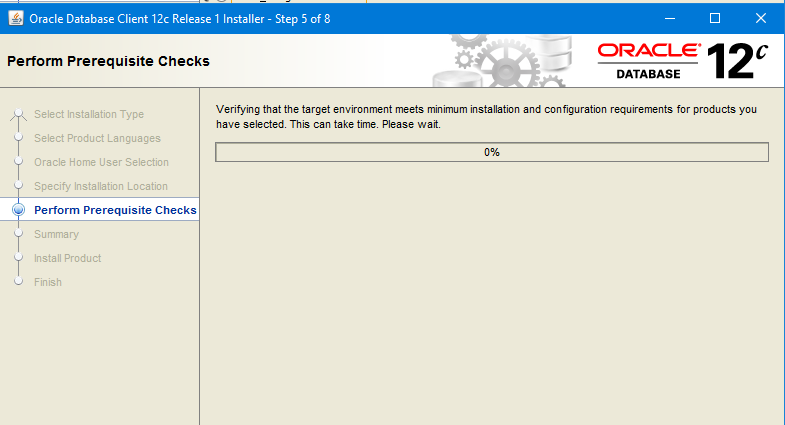
Here you will have to create a new user that does not have Admin rights to the server or the install directory. This will have to be a regular Windows user. You will have the option to choose an existing user or create a new one and when your ready enter **USER NAME** and **PASSWORD** then click the ‘**Next**’ button.



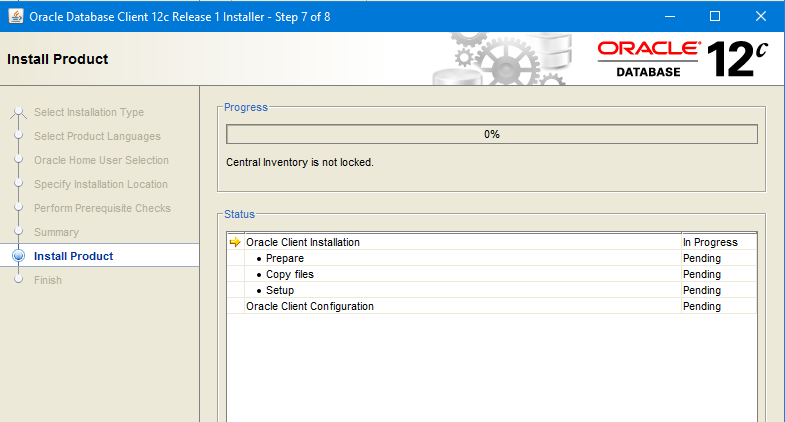
Here you can change the installation default locations .



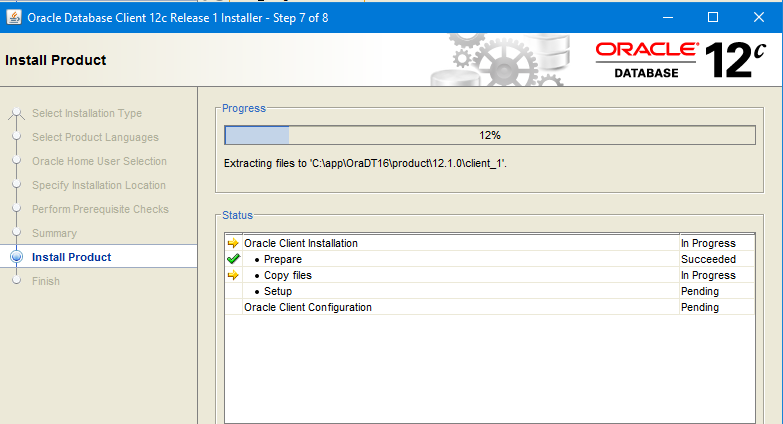
Click **Next** and you have now begun the install.

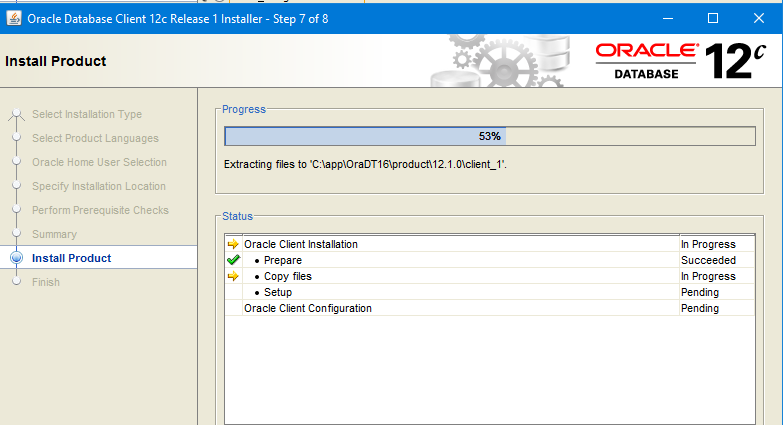


The bar will indicate the install progress.

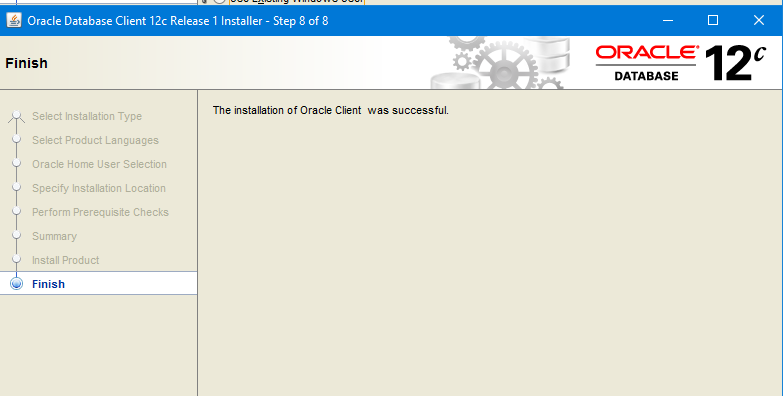


The install progress can be seen in the progress bar located at the top of the window. Towards the middle is the status panel which shows what part of the install has been **completed**, is **Pending**, **Succeeded**, and **In Progress**.





The install has now finished.

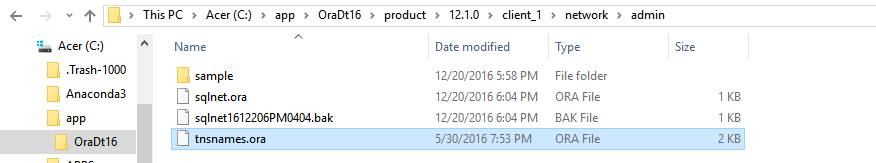


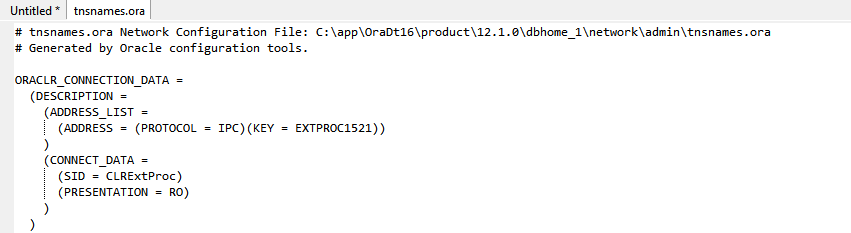
Oracle Client is now installed and you can now click the **Exit** button.

**Oracle Post Installation and configuration**

The Oracle Client 12c has now been installed, but we must now configure it to connect to an Oracle 12c database. In order to configure it you will need to follow these steps:

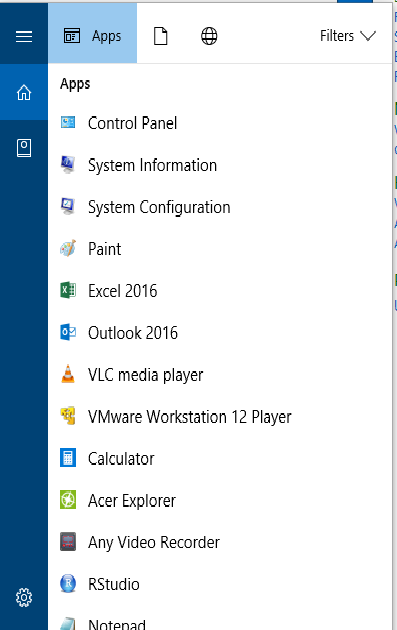
* The first step is to copy the tnsnames.ora file to the NETWORK directory in your client installation. This text file holds all your connection aliases to your various databases.



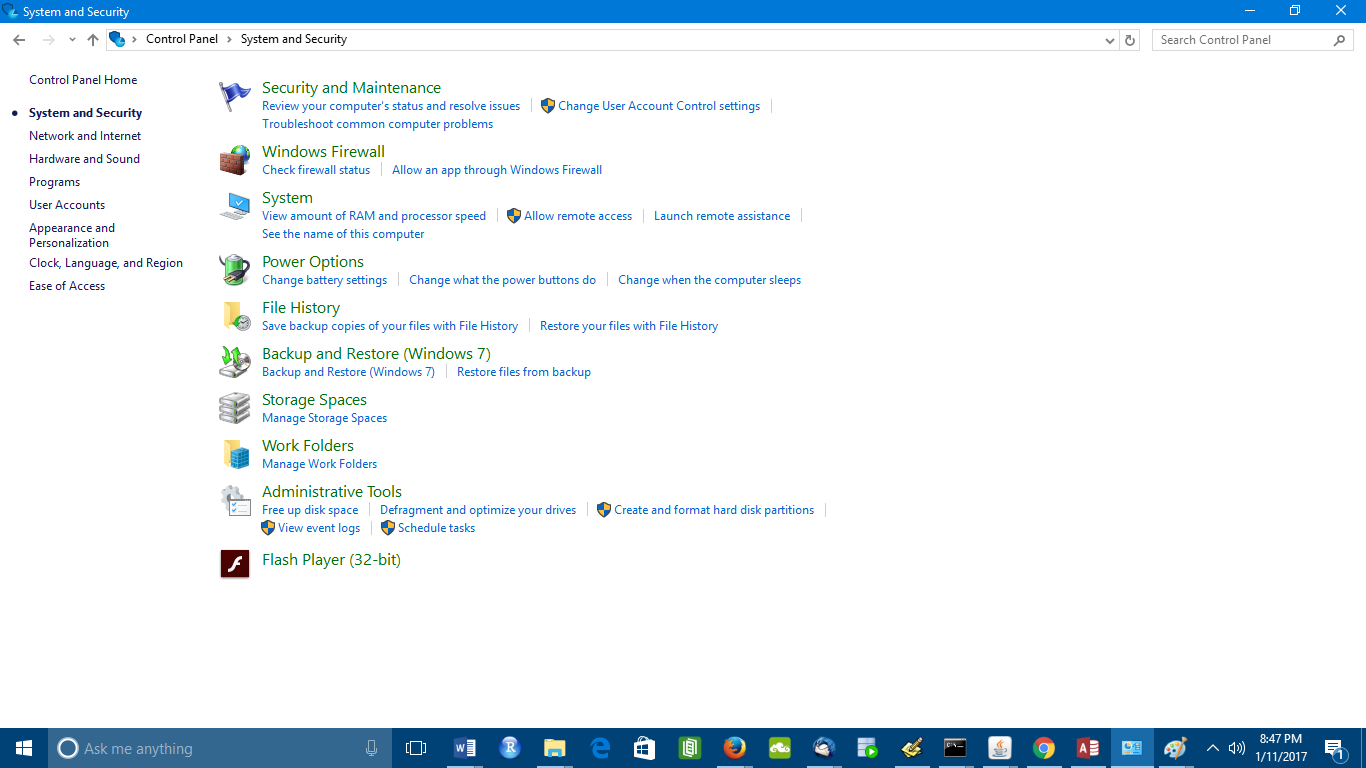


* The second step is to setup your SQL\*Plus environment by creating a SQL login script which will be executed every time you login.

Fortunately, SQL\*Plus allows you to setup a **login.sql** file, a script that is executed each and every time we start **SQL\*Plus**. You can also set an environment variable, **SQLPATH**, so that it can find the **login.sql** script, no matter what directory it is in.

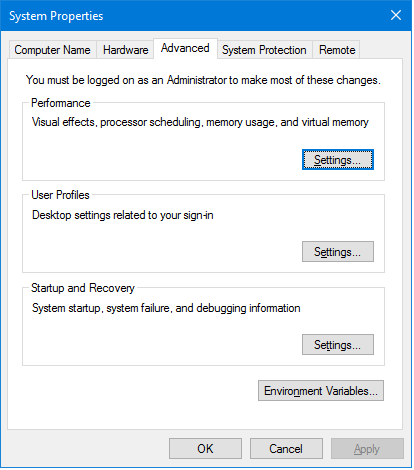


Here we will create a **SQLPATH** environmental variable, first bring up the Windows 10 search box and type ‘**Control Pannel**’ and hit enter.

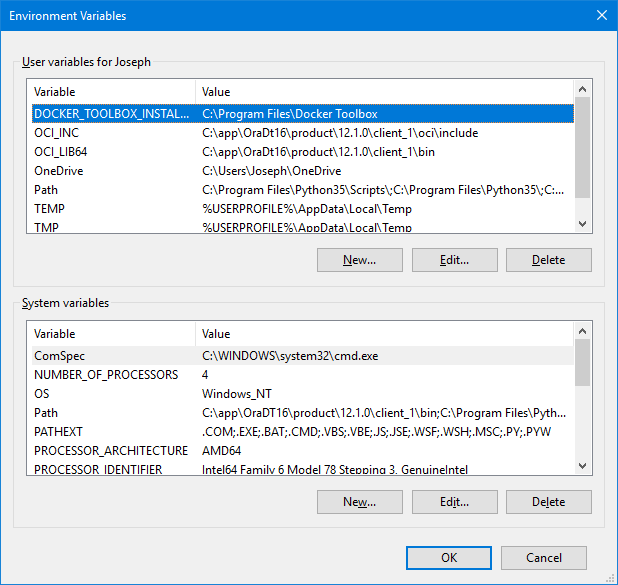


At the Control Pannel select ‘**System and Secuirty**’ and then select ‘**System**’.

Then select ‘**Advanced System Settings**’ and then you will see the ‘**System Properties**’

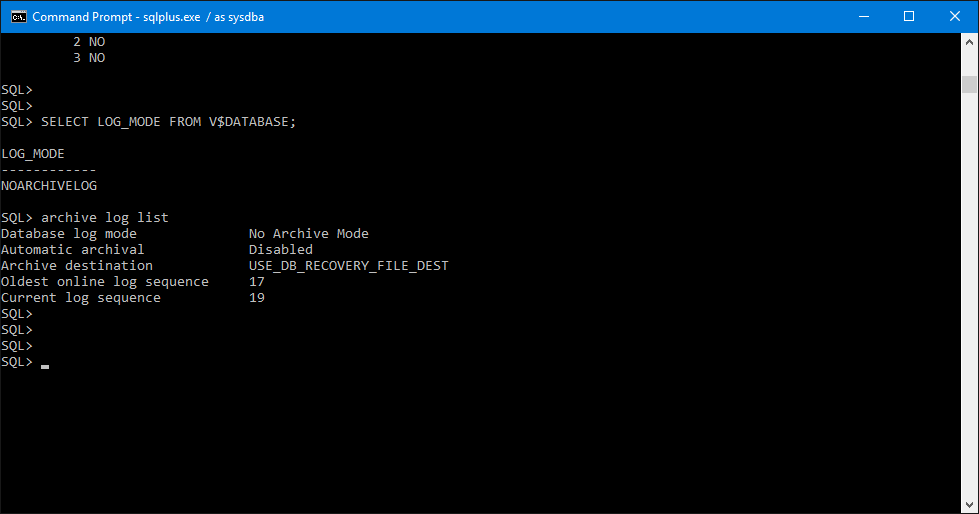


At the Windows **System Properties** window, select “**Environmental Variables**” button

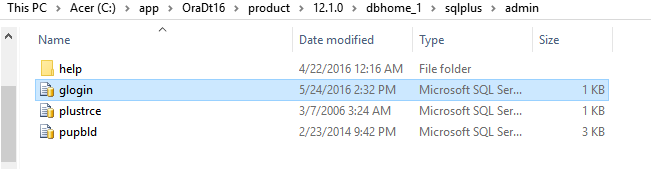


Here we will create a new variable by clicking on the ‘**New**’ button in the panel ‘**User variables for Joseph**’. Here we will add the **SQLPATH** to the system using the path ‘**C:\app\OraDt16\product\12.1.0\dbhome\_1\sqlplus\admin**’. Now click the OK button and you’re done.

The command prompt window for SQL\*Plus looks like the screen below before the script is executed



Add to the **login.sql** script the bold text below: If this is being done on a server then the script name is **glogin.sql** and this will give everyone who logs in to the database the same SQL\*Plus environment.



**define \_editor=notepad**

**set serveroutput on size 1000000**

**set trimspool on**

**set long 5000**

**set linesize 100**

**set pagesize 9999**

**column plan\_plus\_exp format a80**

**column global\_name new\_value gname**

**set termout off**

**define gname=idle**

**column global\_name new\_value gname**

**select lower(user) || '@' || substr( global\_name, 1, decode( dot, 0, length(global\_name), dot-1) ) global\_name**

**from (select global\_name, instr(global\_name,'.') dot from global\_name );**

**set sqlprompt '&gname> '**

**set termout on**

An annotated version of this file is as follows:

* **define \_editor=vi/notepad** - Set up the default editor for **SQL\*Plus** that it would use. You may set that to be your favorite text editor (not a word processor) such as Notepad or emacs.
* **set serveroutput on size unlimited** – This enables **DBMS\_OUTPUT** to be on by default (hence we don't have to type set serveroutput on every time). This also sets the default buffer size to be as large as possible.
* **set trimspool on** - When spooling text, lines will be blank-trimmed and not fixed width. If this is set off (**the default**), spooled lines will be as wide as your linesize setting
* **set long 5000** - Sets the default number of bytes displayed when selecting LONG and CLOB columns.
* **set linesize 100** - Set the width of the lines displayed by SQL\*Plus to be 100 characters which should be the right size in most cases.
* **set pagesize 9999** - Set the pagesize, which controls how frequently SQL\*Plus prints out headings, to a big number (so we get only one set of headings per output).
* **column plan\_plus\_exp format a80** - This sets the default width of the explain plan output we receive with AUTOTRACE. To 80 columns this is generally wide enough to hold the full plan.

The next bit in the **login.sql** script sets up my SQL\*Plus prompt for me:

**define gname=idle**

**column global\_name new\_value gname**

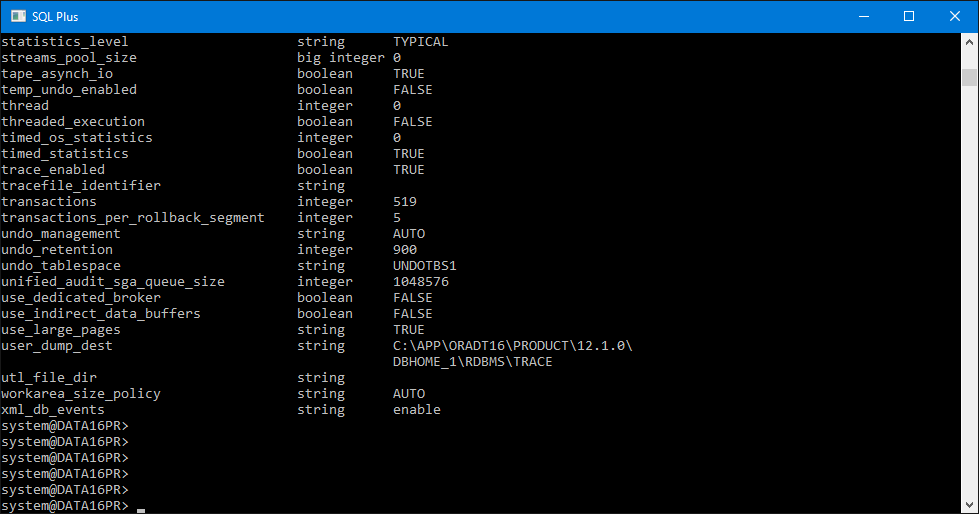
**select lower(user) || '@' || substr( global\_name, 1, decode( dot, 0, length(global\_name),**

**dot-1) ) global\_name from (select global\_name, instr(global\_name,'.') dot from global\_name );**

**set sqlprompt '&gname> '**

The directive column **global\_name** **new\_value** **gname** tells SQL\*Plus to take the last value it retrieves for any column named **global\_name**, and place it into the substitution variable **gname**. I then select the **global\_name** out of the database, and concatenate this with the username I am logged in with.

That makes my prompt look like what you see below.



Your prompt is made up of two parts one your login ID the other the database name so I will know who I am logged in as well as what database I am logged into. An example of this is ‘

**sys@DATA16PR>**’ were I am logged in as SYS and I am on DATA16PR, so you always know where you are.

**Setting Up AUTOTRACE in SQL\*Plus**

**AUTOTRACE** is a facility within **SQL\*Plus** that shows you the explain plan of the queries you’ve executed and the resources they used. If you wish to run efficient SQL queries then this facility is for you

There is more than one way to get **AUTOTRACE** configured. This is what I like to do to get **AUTOTRACE** working:

**C:\app\OraDt16\product\12.1.0\dbhome\_1\sqlplus\admin**

**C:\app\OraDt16\product\12.1.0\dbhome\_1\rdbms\admin**

**1. cd [ORACLE\_HOME]/rdbms/admin**

**2.** log into **SQL\*Plus as SYSTEM**

**3.** Run **@utlxplan**

**4.** Run **CREATE PUBLIC SYNONYM PLAN\_TABLE FOR PLAN\_TABLE;**

**5.** Run **GRANT ALL ON PLAN\_TABLE TO PUBLIC**;

You can replace the **GRANT TO PUBLIC** with some user if you want. By making the **PLAN\_TABLE** public, you let anyone use trace using **SQL\*Plus** that way you don’t have to run **utlplan** on very ID that needs it.

**C:\app\OraDt16\product\12.1.0\dbhome\_1\sqlplus\admin**

The next step is creating and granting the **PLUSTRACE** role:

**1. cd [ORACLE\_HOME]/sqlplus/admin**

**2.** Log in **to SQL\*Plus as SYS or as SYSDBA**

**3.** Run **@plustrce**

**4.** Run **GRANT PLUSTRACE TO PUBLIC;**

Again, you can replace **PUBLIC** in the **GRANT** command with some user if you want.

**About AUTOTRACE**

You can automatically get a report on the execution path used by the SQL optimizer and the statement execution statistics. The report is generated after a successful **SQL DML** (i.e., **SELECT**, **DELETE**, **UPDATE**, **MERGE**, and **INSERT**) statements. It is useful for monitoring and tuning the performance of these statements.

**Controlling the Report**

You can control the report by setting the **AUTOTRACE** system variable:

• **SET AUTOTRACE OFF**: No **AUTOTRACE** report is generated. This is the **default**.

• **SET AUTOTRACE ON EXPLAIN**: The **AUTOTRACE** report shows only the optimizer execution path.

• **SET AUTOTRACE ON STATISTICS**: The **AUTOTRACE** report shows only the SQL statement execution statistics.

• **SET AUTOTRACE ON**: The **AUTOTRACE** report includes both the optimizer execution path and the SQL statement execution statistics.

• **SET AUTOTRACE TRACEONLY**: This is like **SET AUTOTRACE** **ON**, but it suppresses the printing of the user’s query output, if any.

In Oracle 12c database we will be running the SQL\*Plus client for several of our SQL queries/scripts, see below.

Microsoft Windows [Version 10.0.14393]

(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Joseph>sqlplus sys/Dj931#7Hj@//localhost:1521/data16pr as sysdba

SQL\*Plus: Release 12.1.0.2.0 Production on Fri Jan 6 21:11:10 2017

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:

Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production

With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> SHO PDBS

CON\_ID CON\_NAME OPEN MODE RESTRICTED

---------- ------------------------------ ---------- ----------

2 PDB$SEED READ ONLY NO

3 PHGSDATA MOUNTED

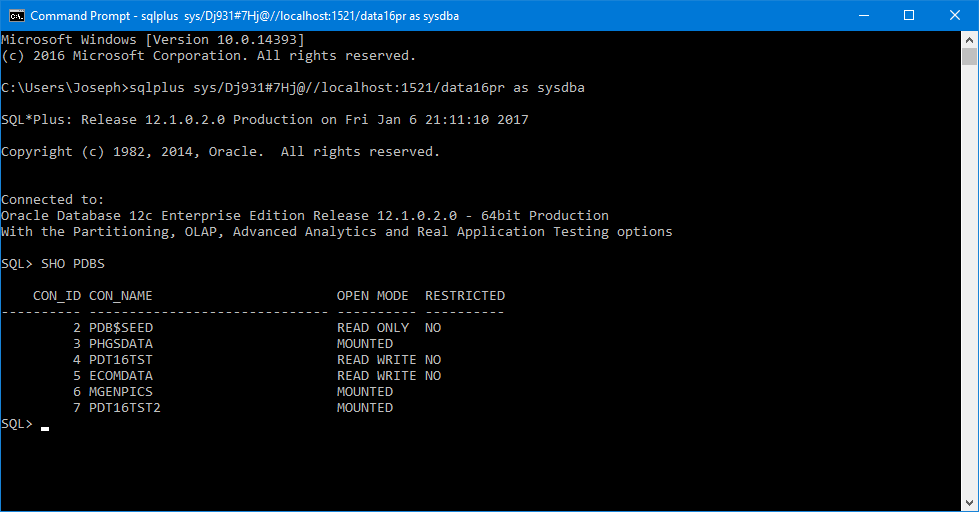
4 PDT16TST READ WRITE NO

5 ECOMDATA READ WRITE NO

6 MGENPICS MOUNTED

7 PDT16TST2 MOUNTED

SQL>



This ends the Installation prerequisites

**Installation of ROracle**

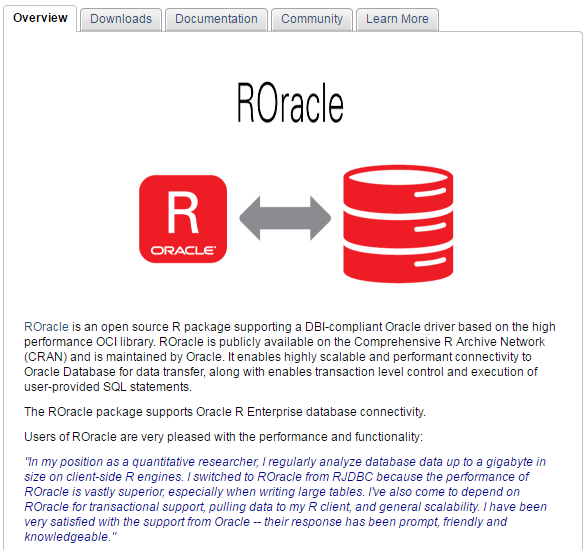
The main web page for ROracle on the **OTN** web site is the page below, besides downloads there is also other useful information, see the URL below.

In benchmark comparisons, ROracle performed up to 79 times faster than **RJDBC** and 2.5 times faster than **RODBC** for reading data across a range of 1000 rows to 1 million rows, and 10 to 1000 columns. ROracle shows scalability across **NUMBER**, **VARCHAR2**, **TIMESTAMP**, and **BINARY\_DOUBLE** data types.

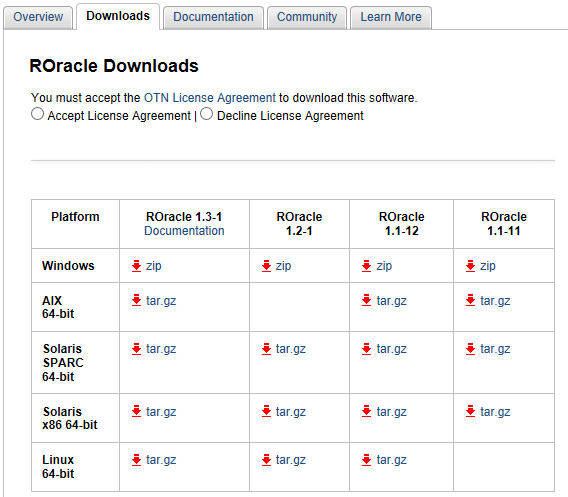
Similarly, for writing data to Oracle Database, ROracle was 61 times faster for 10 columns at 10 thousand rows than **RODBC**, and 630 times faster for the same data than **RJDBC**.

Here the ROracle page is divided into five tabs we want the downloads tab.

<http://www.oracle.com/technetwork/database/database-technologies/r/roracle/downloads/index.html>



Download the most current version of the ROracle package by clicking the download tab using the zip hyperlink for ROarcle 1.3-1 see page below.



Here we will choose Windows and we will get the most current version which is file **ROracle\_1.3-1.zip**. Here we will have to accept the license agreement and then click the zip hyper link below the **ROracle\_1.3-1** heading and the download will proceed if it doesn’t, just do it again.

The documentation tab is for downloading the documentation for this package, click the documentation tab, see the URL below to go to that page.

<http://www.oracle.com/technetwork/database/database-technologies/r/roracle/documentation/index.html>

There is only one file there and that is **ROracle.pdf** click the hyper link to begin downloading. Create a working directory on your PC call it ROracle move the files into it. In order to install this package bring up RStudio and we will run the command below.

The ‘**install.packages**’ command/function will also need three parameters the **path**, the **file name** and the other parameter will tell the function **not to use a repository** see below.

**Directory Parameter**

This the directory where you downloaded the ROracle to and where the other files you downloaded are.

C:\DATATREE\_NEW\_HOME\R\_ENTERPRISE\ROracle

Since one of the parameters is a directory path it will use escape characters, these will have to be suppressed. The function will deal with a single escape character by giving you an error, see below.

> install.packages("C:\DATATREE\_NEW\_HOME\R\_ENTERPRISE\ROracle\ROracle\_1.3-1.zip", repos=NULL)

Error: '\D' is an unrecognized escape in character string starting ""C:\D"

So the escape character ‘\’ will need to be doubled to ‘\\’ characters so that the ‘**install.packages**’ command/function will not give us an error. You will need to type a command like what you see below.

**install.packages("C:\\DATATREE\_NEW\_HOME\\R\_ENTERPRISE\\ROracle\\ROracle\_1.3-1.zip", repos=NULL)**

We will run these commands in RStudio, the log below showsthe results you will get.

R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"

Copyright (C) 2016 The R Foundation for Statistical Computing

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'license()' for distribution details.

R is a collaborative project with many contributors.

Type 'contributors()' for more information and

'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>

> install.packages("C:\\DATATREE\_NEW\_HOME\\R\_ENTERPRISE\\ROracle\\ROracle\_1.3-1.zip", repos=NULL)

Installing package into ‘C:/Users/Joseph/Documents/R/win-library/3.3’

(as ‘lib’ is unspecified)

The package ROracle/Driver successfully been unpacked and MD5 sums checked

We have now installed the following software products:

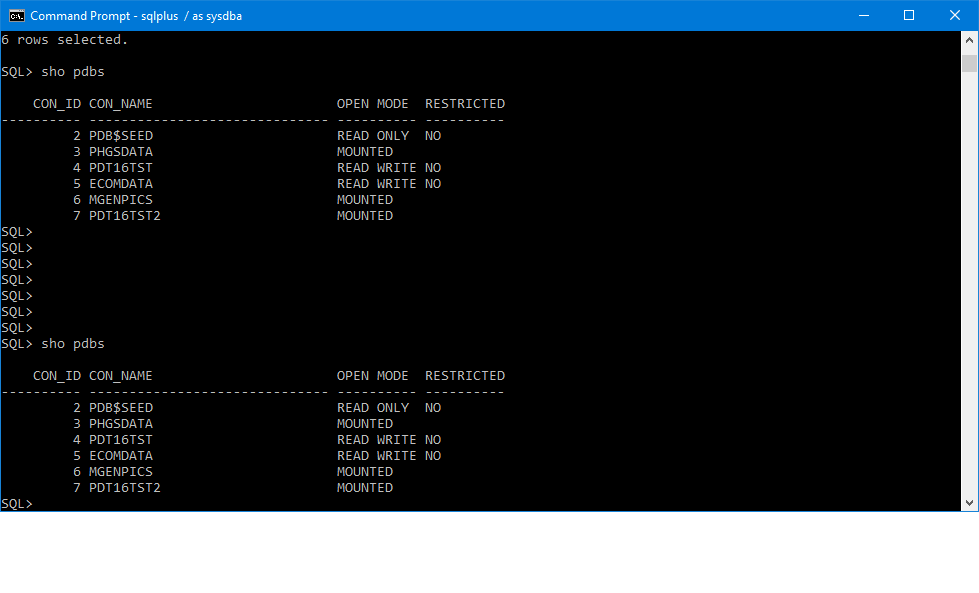
* R version 3.2.2.
* RStudio 1.0.44.
* Oracle 12c Client (12.1.0.2) and configured it.
* ROracle 1.3-1 is installed as a package under the R language

You have now installed all the prerequisites and the Roracle package itself. Now we will need to test it, for this you will need an Oracle database to connect to.

I will be using an Oracle 12c (12.1.0.2) database for this test. It has six pluggable databases on it with various datasets we will try to connect to **DATA16PR** ( **CDB$ROOT**) and **PDT16TST** which has many data sets.

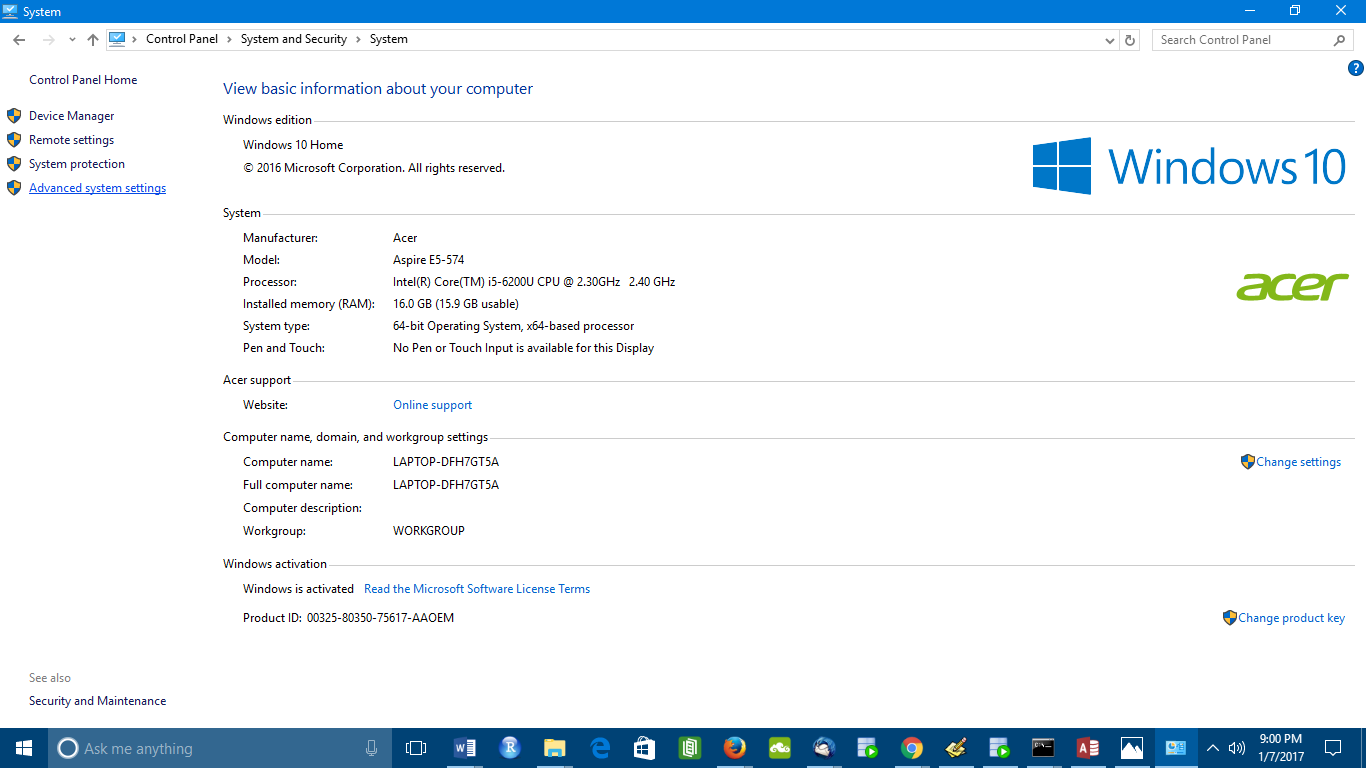
We will be using **Yellow Taxi trip data** which is a **GIS** data set for 2015 NYC Data. This data set has data for 2015 for NYC from January through June of that year. The table has over 147 million records and is 11 GB in size before it was loaded into Oracle.

The image below of a **SQL\*Plus** windows prompt window with the **show pdbs** command‘s output is were we will execute many of our SQL scripts.

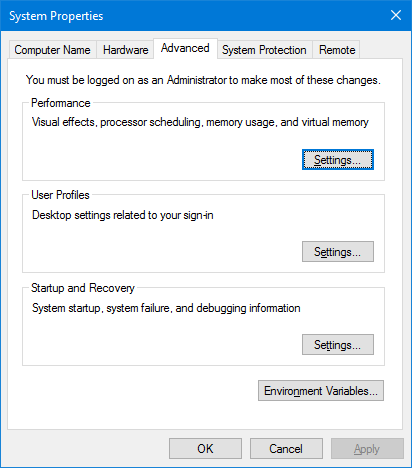


The **PATH** should be set to include the path to the Oracle Client libraries. Oracle Universal Installer should have set the **PATH** and **ORACLE\_HOME** environment variables in the registry database.

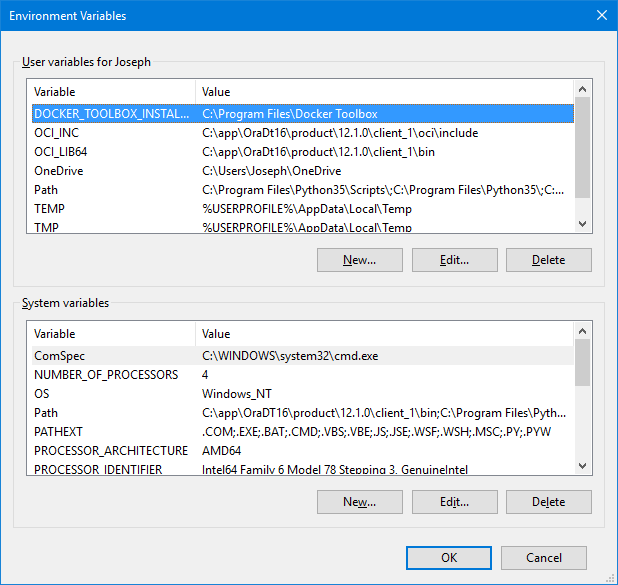
Search for ‘**Control Pannel**’ and you will see the screen listed below. Then select the ‘**Advance System Settings**’ will get you to the Environmental Variables.

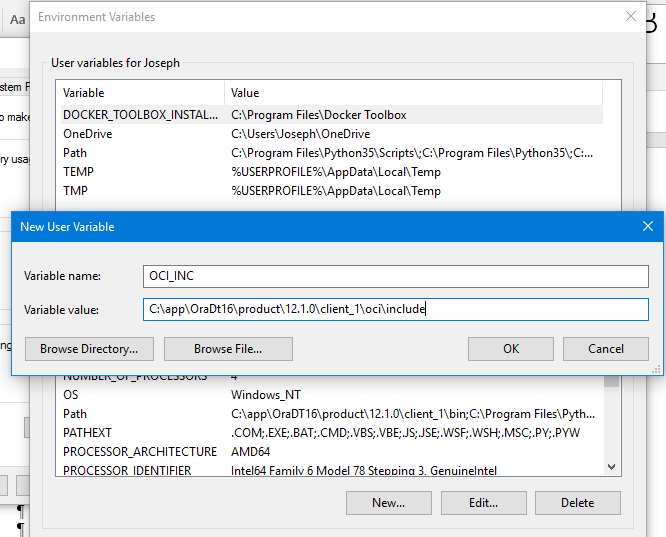


Select ‘**Environment Variables**’ and here we must create two new ones to locate the Oracle OCI libraries under the Oracle 12c Client install directories.



Click the **New** button for ‘**User variables for Joseph**’.



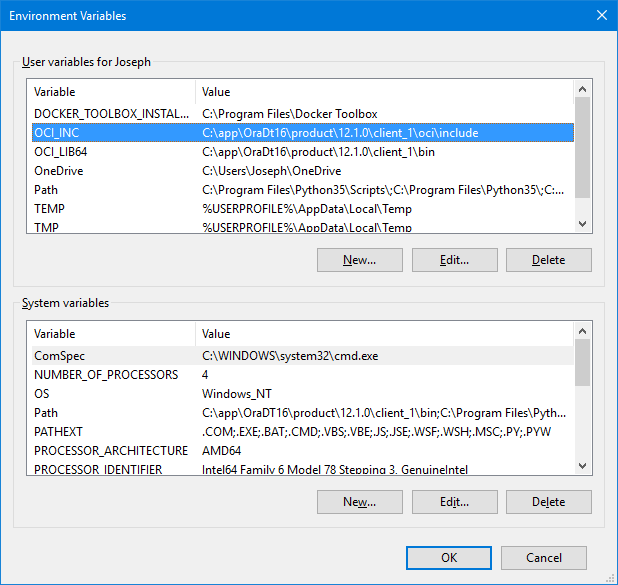


Here we will create two variables they are **OCI\_INC** and **OCI\_LIB64**. When the ‘**New User Variable**’ dialog comes up enter both the ‘**Variable Name**’ and ‘**Variable Value**’ then click the OK button. The two values are listed below.

**set OCI\_INC=C:\app\OraDt16\product\12.1.0\client\_1\oci\include**

**set OCI\_LIB64= C:\app\OraDt16\product\12.1.0\client\_1\bin**

The two variables are listed below, when finished click OK.



Load the library and use the package; you will have to change **DBNAME** to one of aliases listed in your **tnsnames.ora** file, as in the following example I am using the database container CDB$ROOT for my dbname of **data16pr**:

**library('ROracle')**

**drv <- dbDriver("Oracle")**

**#sample connect command**

**con <- dbConnect(drv, "USERNAME", "PASSWORD", dbname='DBNAME')**

**# real connect command**

**con <- dbConnect(drv, "system ", "Dj931#7&", dbname='data16pr')**

> library('ROracle')

> drv <- dbDriver("Oracle")

>

> con <- dbConnect(drv, " system ", " Dj931#7&", dbname=' data16pr')

**Error in .oci.Connect(.oci.drv(), username = username, password = password, :**

**ORA-01017: invalid username/password; logon denied**

>

The reason this happened is that spaces are in both the password and the ID, see below for the difference.

>

> library('ROracle')

> drv <- dbDriver("Oracle")

>

> con <- dbConnect(drv, "system ", "Dj931#7&", dbname=' data16pr')

>

Test the connection to the Oracle 12c database:

The **dbReadTable()** function can be used to read whatever is in the table **DUAL**.

**dbReadTable(con, 'DUAL')**

>

> library('ROracle')

> drv <- dbDriver("Oracle")

>

> con <- dbConnect(drv, "system ", "Dj931#7&", dbname=' data16pr')

>

> dbReadTable(con, 'DUAL')

DUMMY

1 X

>

>

The **dbReadTable()** function can be used to read whatever is in the table **V$DATABASE**.

**dbReadTable(con, 'V$DATABASE')**

>

> dbReadTable(con, 'V$DATABASE')

DBID NAME CREATED RESETLOGS\_CHANGE# RESETLOGS\_TIME

1 2241495645 DATA16PR 2016-04-22 01:42:05 2233668 2016-04-22 01:42:10

PRIOR\_RESETLOGS\_CHANGE# PRIOR\_RESETLOGS\_TIME LOG\_MODE CHECKPOINT\_CHANGE#

1 1 2014-09-11 09:40:48 ARCHIVELOG 9982367

ARCHIVE\_CHANGE# CONTROLFILE\_TYPE CONTROLFILE\_CREATED CONTROLFILE\_SEQUENCE#

1 9963817 CURRENT 2016-04-22 01:42:06 31070

CONTROLFILE\_CHANGE# CONTROLFILE\_TIME OPEN\_RESETLOGS VERSION\_TIME

1 9986951 2017-01-06 22:51:47 NOT ALLOWED 2016-04-22 01:42:05

OPEN\_MODE PROTECTION\_MODE PROTECTION\_LEVEL REMOTE\_ARCHIVE ACTIVATION#

1 READ WRITE MAXIMUM PERFORMANCE MAXIMUM PERFORMANCE ENABLED 2241525340

SWITCHOVER# DATABASE\_ROLE ARCHIVELOG\_CHANGE# ARCHIVELOG\_COMPRESSION

1 2241525340 PRIMARY 9982367 DISABLED

SWITCHOVER\_STATUS DATAGUARD\_BROKER GUARD\_STATUS SUPPLEMENTAL\_LOG\_DATA\_MIN

1 NOT ALLOWED DISABLED NONE NO

SUPPLEMENTAL\_LOG\_DATA\_PK SUPPLEMENTAL\_LOG\_DATA\_UI FORCE\_LOGGING PLATFORM\_ID

1 NO NO NO 12

PLATFORM\_NAME RECOVERY\_TARGET\_INCARNATION# LAST\_OPEN\_INCARNATION#

1 Microsoft Windows x86 64-bit 2 2

CURRENT\_SCN FLASHBACK\_ON SUPPLEMENTAL\_LOG\_DATA\_FK SUPPLEMENTAL\_LOG\_DATA\_ALL

1 9986964 YES NO NO

DB\_UNIQUE\_NAME STANDBY\_BECAME\_PRIMARY\_SCN FS\_FAILOVER\_STATUS

1 data16pr 0 DISABLED

FS\_FAILOVER\_CURRENT\_TARGET FS\_FAILOVER\_THRESHOLD FS\_FAILOVER\_OBSERVER\_PRESENT

1 <NA> 0 <NA>

FS\_FAILOVER\_OBSERVER\_HOST CONTROLFILE\_CONVERTED PRIMARY\_DB\_UNIQUE\_NAME

1 <NA> NO <NA>

SUPPLEMENTAL\_LOG\_DATA\_PL MIN\_REQUIRED\_CAPTURE\_CHANGE# CDB CON\_ID

1 NO NA YES 0

PENDING\_ROLE\_CHANGE\_TASKS CON\_DBID FORCE\_FULL\_DB\_CACHING

1 NOT APPLICABLE 2241495645 NO

>

>

The **dbReadTable()** function can be used to read whatever is in the table **V$PDBS**.

**dbReadTable(con, 'V$PDBS')**

|  |
| --- |
| dbReadTable(con, 'V$PDBS')  CON\_ID DBID CON\_UID  1 2 369848507 369848507  2 3 1769860667 1769860667  3 4 2647541362 2052914599  4 5 1132693728 1132693728  5 6 2975992775 2975992775  6 7 2938944833 2938944833  GUID NAME  1 4f, 24, fe, dc, 9e, 82, 4f, 44, 95, 1d, 1e, a4, 9c, 42, ff, 4e PDB$SEED  2 72, 19, 85, 7a, 52, ab, 4c, 13, ac, 78, ad, 59, bc, 58, f3, 6e PHGSDATA  3 ca, d5, c4, e6, 39, a5, 49, 1f, 83, b0, 4b, a6, 67, 11, 40, 38 PDT16TST  4 7d, eb, c2, 1e, ce, 15, 45, 49, bb, 39, 31, 3d, 18, 0c, 96, 7f ECOMDATA  5 0f, a5, f4, b5, e4, 24, 45, c9, 9c, 86, c1, f6, b6, 7a, a4, e2 MGENPICS  6 fe, 04, 38, b2, f7, f0, 49, 14, a2, 7a, cc, 10, 7d, 51, 2b, dd PDT16TST2  OPEN\_MODE RESTRICTED OPEN\_TIME CREATE\_SCN TOTAL\_SIZE BLOCK\_SIZE  1 READ ONLY NO 2016-12-25 21:24:49 2233966 796917760 8192  2 MOUNTED <NA> <NA> 3434346 0 8192  3 READ WRITE NO 2016-12-27 21:43:26 3443508 8113487872 8192  4 READ WRITE NO 2016-12-27 21:44:33 3460730 1390411776 8192  5 MOUNTED <NA> <NA> 3538655 0 8192  6 MOUNTED <NA> <NA> 5891656 0 8192  RECOVERY\_STATUS SNAPSHOT\_PARENT\_CON\_ID  1 ENABLED 0  2 ENABLED 0  3 ENABLED 0  4 ENABLED 0  5 ENABLED 0  6 ENABLED 0 |
|  |

**Performance Test of Database connection to the server.**

We will be using **Yellow Taxi trip data** which is a GIS data set for the year 2015. This data set has data for 2015 for NYC from January through June of that year. The table has 147 million records and is 11 GB in size before it was loaded into Oracle.

We will benchmark both R and SQL programs which be doing the same queries on the data set ‘**TAXI\_TRIP\_DATA**’ table. I will be timing each programs and checking the run times for comparison. In addition we will testing joins and rollups of the data from other tables. Also we will use analytic functions and all queries will be run with and without indexes.

**Examples**

## Not run:

## Create an Oracle Database instance and create one connection on the

## same machine.

drv <- dbDriver("Oracle")

## Use username/password authentication.

con <- dbConnect(drv, username = "scott", password = "tiger")

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from emp where deptno = 10")

## We now fetch records from the resultSet into a data.frame.

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

## Not run:

## Create an Oracle Database instance and create one connection to a

## remote database using the SID in the connect string.

drv <- dbDriver("Oracle")

## Refer to Oracle Database Net Services Administator's Guide for

## details on connect string specification.

host <- "myhost"

port <- 1521

sid <- "mysid"

connect.string <- paste( "(DESCRIPTION=",

"(ADDRESS=(PROTOCOL=tcp)(HOST=", host, ")(PORT=", port, "))",

"(CONNECT\_DATA=(SID=", sid, ")))", sep = "")

## Use username/password authentication.

con <- dbConnect(drv, username = "scott", password = "tiger",

dbname = connect.string)

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from emp where deptno = 10")

## We now fetch records from the resultSet into a data.frame.

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

## Not run:

## Create an Oracle Database instance and create one connection to a

## remote database using the service name.

drv <- dbDriver("Oracle")

## Refer to Oracle Database Net Services Administator's Guide for

## details on connect string specification.

host <- "myhost"

port <- 1521

svc <- "mydb.example.com"

connect.string <- paste(

"(DESCRIPTION=",

"(ADDRESS=(PROTOCOL=tcp)(HOST=", host, ")(PORT=", port, "))",

"(CONNECT\_DATA=(SERVICE\_NAME=", svc, ")))", sep = "")

## Use username/password authentication.

con <- dbConnect(drv, username = "scott", password = "tiger",

dbConnect-methods 7

dbname = connect.string)

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from emp where deptno = 10")

## We now fetch records from the resultSet into a data.frame.

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

## Not run:

## Create an Oracle Database instance and create one connection.

drv <- dbDriver("Oracle")

## Use Oracle Wallet authentication.

con <- dbConnect(drv, username ="", password="",

dbname = "<wallet\_connect\_string>")

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from emp where deptno = 10")

## We now fetch records from the resultSet into a data.frame.

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

## Not run:

## Create an Oracle Database instance and create one connection.

drv <- dbDriver("Oracle")

## Connect to a TimesTen IMDB instance using the easy connect

## naming method where SampleDb is a direct driver TimesTen DSN.

con <- dbConnect(drv, username ="scott", password="tiger",

dbname = "localhost/SampleDb:timesten\_direct")

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from dual")

## We now fetch records from the resultSet into a data.frame.

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

## Not run:

## Connect to an extproc (this assumes that the driver has already

## been initialized in the embedded R code by passing an external

## pointer representing the extproc context).

con <- dbConnect(Extproc())

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from dual")

## We now fetch records from the resultSet into a data.frame.

8 dbDriver-methods

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

## Not run:

## Create an Oracle Database instance and create one connection.

drv <- dbDriver("Oracle")

## Create connection with SYSDBA privileges.

con <- dbConnect(drv, username ="scott", password="tiger",

sysdba = TRUE)

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from emp where deptno = 10")

## We now fetch records from the resultSet into a data.frame.

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

## Not run:

## Create an Oracle Database instance and create one connection.

drv <- dbDriver("Oracle")

## Use OS authentication as an example of external authentication

## Make sure that databse user exist to allow an OS authentication

## Create connection authenticated with external credentials.

con <- dbConnect(drv, username ="", password="",

external\_credentials = TRUE)

## Above dbConnect() used OS credentials to connect with database.

## Run a SQL statement by creating first a resultSet object.

rs <- dbSendQuery(con, "select \* from emp where deptno = 10")

## We now fetch records from the resultSet into a data.frame.

data <- fetch(rs) ## extract all rows

dim(data)

## End(Not run)

dbDriver-methods Oracle Implementation of the Database Interface (DBI) Classes and

Drivers

Description

Oracle driver initialization and closing.

dbDriver-methods 9

Usage

## S4 method for signature 'OraDriver'

dbUnloadDriver(drv, ...)

## S4 method for signature 'ExtDriver'

dbUnloadDriver(drv, ...)

Arguments

drv An object that inherits from OraDriver or ExtDriver as created by dbDriver.

... Any other arguments to pass to the driver drvName.

**Details**

**dbDriver** This object is a singleton, that is, subsequent invocations of dbDriver return the same

initialized object.

This implementation allows you to connect to multiple host servers and run multiple connections

on each server simultaneously. When interruptible is set to **TRUE**, it allows for interrupting long-running queries on the server by executing the query in a thread. Main thread checks for **Ctrl-C** and issues **OCIBreak/OCIReset** to cancel the operation on the server. By default, interruptible is **FALSE**. When **unicode\_as\_utf8** is set to **FALSE**, **NCHAR**, **NVARCHAR** and **NCLOB** data is fetched using the character set specified by the **NLS\_LANG** setting. By default, **unicode\_as\_utf8** is set to **TRUE**. When **ora.attributes** is set to **TRUE**, the result set from **dbGetQuery** and fetch contains DBMS-specific attributes like **ora.encoding**, **ora.type**, and **ora.maxlength** for the corresponding column.

**dbUnloadDriver** This implementation removes communication links between the R client and the database. It frees all connections and all result sets associated with those connection objects.

**dbDriver** An object **OraDriver** or **ExtDriver** whose class extends **DBIDriver**. This object

is used to create connections, using the function **dbConnect**, to one or more Oracle Database engines.

**dbUnloadDriver** Free all resources occupied by the driver object.

**dbDriver** The **R** client part of the database communication is initialized, but note that connecting to the database engine needs to be done through calls to dbConnect.

**dbUnloadDriver** Remove the communication link between the R client and the database.