ORACLE

Session 6: ROracle

Mark Hornick, Director, Oracle Advanced Analytics Development
Oracle Advanced Analytics

Topics

- What is ROracle?
- Using ROracle
- Summary

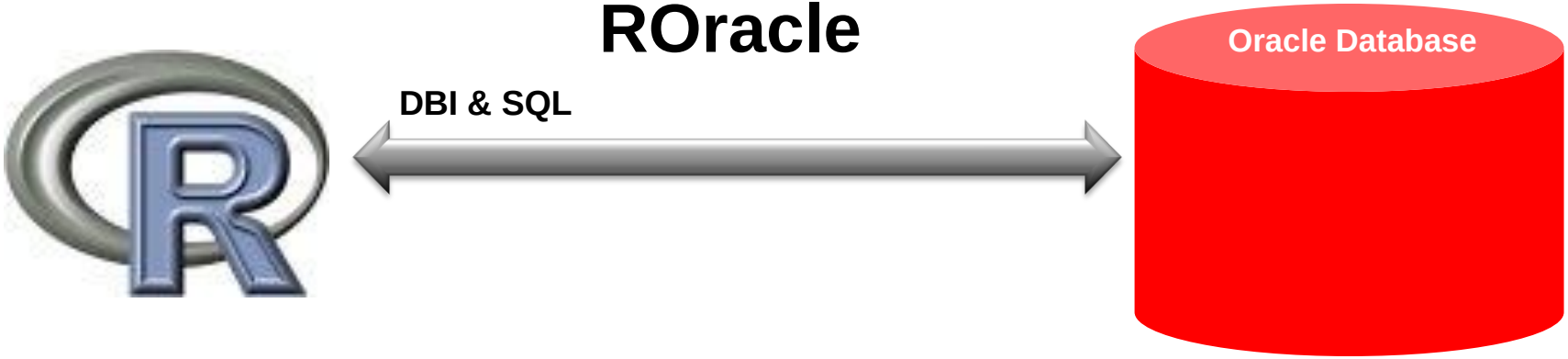
What is ROracle?

ROracle

- R package enabling connectivity to Oracle Database
  - Open source
  - Publically available on CRAN
- Execute SQL statements from R interface
- Oracle Database Interface (DBI) for R
- DBI –compliant Oracle driver based on OCI
- Requirements
  - Oracle Instant Client or Oracle Database Client

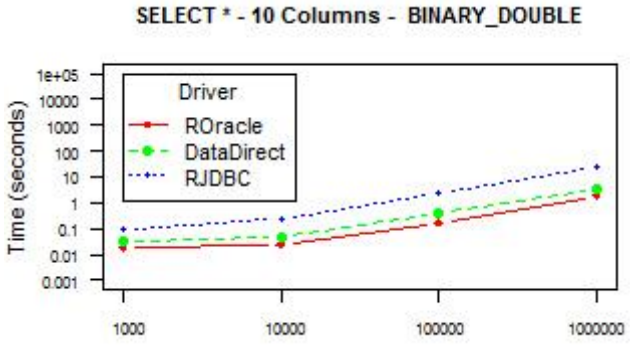
Examples from ROracle package documentation http://cran.r-project.org/web/packages/ROracle/ROracle.pdf

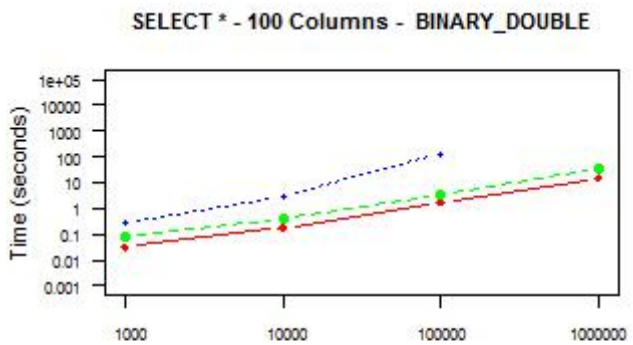
ROracle



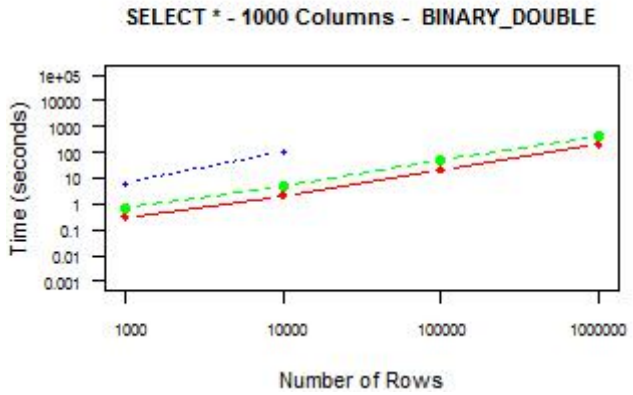
- R package enabling connectivity to Oracle Database
  - Open source, publicly available on CRAN, free to R community
- Execute SQL statements from R interface
- Oracle Database Interface (DBI) for R based on OCI for high performance
- Supports **Oracle R Enterprise** database connectivity

Comparison reading database table to R data.frame



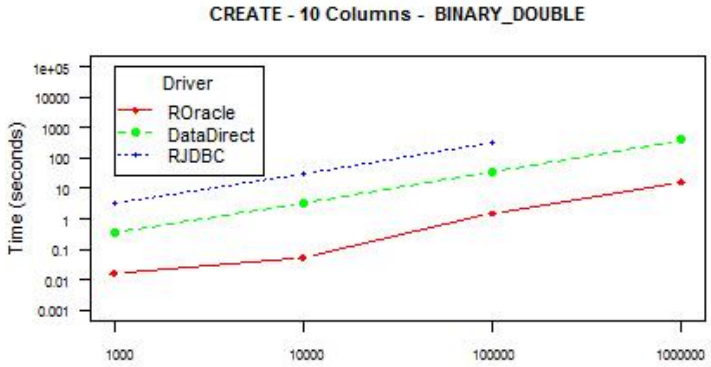


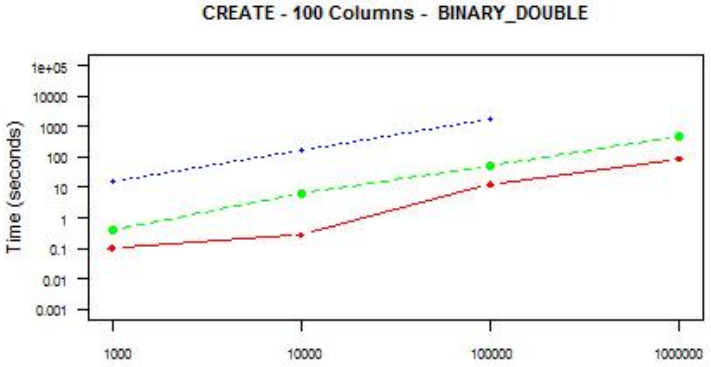
- ROracle
  - Up to 79X faster than RJDBC
  - Up to 2.5X faster than RODBC
  - Scales across NUMBER, VARCHAR2, TIMESTAMP data types



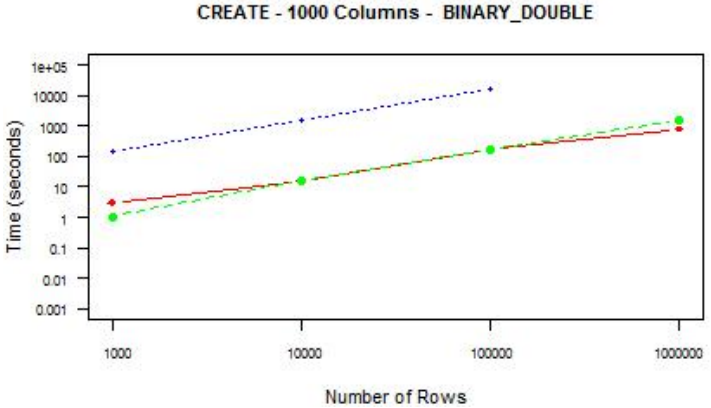
See https://blogs.oracle.com/R/entry/r_to_oracle_database_connectivity

Comparison writing database table from R data.frame





- ROracle
  - 61X faster for 10 cols x 10K rows than RODBC
  - 630X faster on 10 cols x 10K rows than RJDBC
  - Scales across the remaining data types



ROracle 1-1.11 Enhancements

- Performance enhancements for RAW data types and large result sets
- Cache resultset in memory before transferring to R to avoid unnecessary alloc and free using allocVector when result exceeds bulk_read rows
- Added session mode to connect as SYSDBA or using external authentication
- bug 17383542: Enhanced dbWritetable() & dbRemoveTable() to work on global schema

Using ROracle

Example – rolling back transactions

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

# Create the connection string
host <- "adc2100958"
port <- 1521
sid <- "orff"
connect.string <- paste(
  "(DESCRIPTION=",
  "(ADDRESS=(PROTOCOL=tcp)(HOST=", host, ")(PORT=", port, "))",
  "(CONNECT_DATA=(SID=", sid, ")))", sep = "")

con <- dbConnect(drv, username = "scott",
                 password = "tiger",dbname=connect.string)
```

Example – rolling back transactions

```
dbReadTable(con, "EMP")

rs <- dbSendQuery(con, "delete from emp where deptno = 10")

dbReadTable(con, "EMP")
if(dbGetInfo(rs, what = "rowsAffected") > 1){
   warning("dubious deletion -- rolling back transaction")
   dbRollback(con)
}

dbReadTable(con, "EMP")
```

Example – username/password authentication

```
## create an Oracle instance and create one connection
drv <- dbDriver("Oracle")

## use username/password authentication – if using local database
con <- dbConnect(drv, username = "scott", password = "tiger")

## run a SQL statement by first creating a resultSet object
rs <- dbSendQuery(con, "select * from emp where deptno = 10")

## fetch records from the resultSet into a data.frame
data <- fetch(rs)
## extract all rows
dim(data)
data
```

Example – connect to TimeTen IMDB instance

```
## create an Oracle 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 an 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)
```

Example – connect to an extproc for use within ERE

```
## connect to an extproc (this assumes that driver has already
## been initialized in the embedded code by passing an external
## pointer representing extproc context)
con <- dbConnect(Extproc())

## run a SQL statement by first creating 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)
```

Example – unload driver

```
# create an Oracle instance
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger", dbname=connect.string)
res <- dbSendQuery(con, "select * from emp")
fetch(res, n = 5)
fetch(res)

# free resources occupied by result set
dbClearResult(res)
dbUnloadDriver(drv)
```

Example – getInfo methods

```
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger", dbname=connect.string)
rs <- dbSendQuery(con, "select * from emp")
dbGetStatement(rs)
dbHasCompleted(rs)
dbGetInfo(rs)

# DBIDriver info
names(dbGetInfo(drv))

# DBIConnection info
names(dbGetInfo(con))

# DBIResult info
names(dbGetInfo(rs))
```

Example – getInfo methods

```
drv  <- dbDriver("Oracle")
con1 <- dbConnect(drv, "scott", "tiger", dbname=connect.string)
res1 <- dbSendQuery(con1, "select * from emp where deptno = 10")
res2 <- dbSendQuery(con1, "select * from emp where deptno = 20")
con2 <- dbConnect(drv, "scott", "tiger")
res3 <- dbSendQuery(con2, "select * from dept")

## get all active statements
for(con in dbListConnections(drv))
  for (res in dbListResults(con))
    print(dbGetStatement(res))
```

Example – read/write table methods

```
con <- dbConnect(Oracle(), "scott", "tiger", dbname=connect.string)
if (dbExistsTable(con, "FOO", "SCOTT"))
dbRemoveTable(con, "FOO")
foo <- dbReadTable(con, "EMP")
row.names(foo) <- foo$EMPNO
foo <- foo[,-1]
dbWriteTable(con, "FOO", foo, row.names = TRUE)
dbWriteTable(con, "FOO", foo, row.names = TRUE, overwrite = TRUE)
dbReadTable(con, "FOO", row.names = 1)
dbGetQuery(con, "delete from foo")
dbWriteTable(con, "FOO", foo, row.names = TRUE, append = TRUE)
dbReadTable(con, "FOO", row.names = 1)
dbRemoveTable(con, "FOO")
dbListTables(con)
dbListFields(con, "EMP")
if (dbExistsTable(con, "RORACLE_TEST", "SCOTT"))
dbRemoveTable(con, "RORACLE_TEST")
```

Example – read/write table methods

```
# example of POSIXct usage
# A table is created using:
createTab <- "create table RORACLE_TEST(row_num number, id1 date,
                  id2 timestamp, id3 timestamp with time zone,
                  id4 timestamp with local time zone )"
dbGetQuery(con, createTab)

# Insert statement
insStr <- "insert into RORACLE_TEST values(:1, :2, :3, :4, :5)";

# Select statement
selStr <- "select * from RORACLE_TEST";

# Insert time stamp without time values in POSIXct form
x <- 1;
y <- "2012-06-05";
y <- as.POSIXct(y);
dbGetQuery(con, insStr, data.frame(x, y, y, y, y));
```

Example – read/write table methods

```
# Insert date & times tamp with time values in POSIXct form
x <- 2;
y <- "2012-01-05 07:15:02";
y <- as.POSIXct(y);
z <- "2012-01-05 07:15:03.123";
z <- as.POSIXct(z);
dbGetQuery(con, insStr, data.frame(x, y, z, z, z));
# Insert list of date objects in POSIXct form
x <- c(3, 4, 5, 6);
y <- c('2012-01-05', '2011-01-05', '2014-01-05', '2020-01-05');
y <- as.POSIXct(y);
dbGetQuery(con, insStr, data.frame(x, y, y, y, y));
dbCommit (con)
# Selecting data and displaying it
res <- dbGetQuery(con, selStr)
res[,1]
res[,2]
res[,3]
res[,4]
res[,5]
```

Example – send query methods

```
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger")
res <- dbSendQuery(con, "select * from emp where deptno = :1",
                   data = data.frame(deptno = 10))
data <- fetch(res, n = -1)
res2 <- dbSendQuery(con, "select * from emp where deptno = :1",
                    data = data.frame(deptno = 10), prefetch=TRUE, bulk_read=2L)
data1 <- fetch(res2, n = -1)
res3 <- dbSendQuery(con, "select * from emp where deptno = :1",
                    data = data.frame(deptno = 10), bulk_read=10L)
data2 <- fetch(res3, n = -1)
res4 <- dbSendQuery(con, "select * from emp where ename = :1",
                    data = data.frame(ename = 'SMITH'))
data3 <- fetch(res4, n = -1)
```

Example – Oracle method

```
## create a Oracle instance and create one connection.
ora <- Oracle() ## or dbDriver("Oracle")
con <- dbConnect(ora, username = "scott", password = "tiger", dbname = "inst1")

## if you are connecting to a local database
con <- dbConnect(ora, username = "scott", password = "tiger")

## execute a statement and fetch output in chunks <= 5000 rows at a time
rs <- dbSendQuery(con, "select * from emp where deptno = 10")
while (!dbHasCompleted(rs)) {
  df <- fetch(rs, n = 5000)
  ## process df
}
dbClearResult(rs) ## done with this query
```

Example – Oracle method

```
## execute and fetch a statement with bind data
df <- dbGetQuery(con, "select * from emp where deptno = :1", data = data.frame(depno = 10))

## create a copy of emp table
dbGetQuery(con, "create table foo as select * from emp")

## execute and bind an INSERT statement
my.data   = data.frame(empno = c(8001, 8002), ename = c('MUKHIN', 'ABOYOUN'))
more.data = data.frame(empno = c(8003), ename = c('JAMES'))
rs <- dbSendQuery(con, "insert into foo (empno, ename) values (:1, :2)",data = my.data)

## execute with more data
execute(rs, data = more.data)
df <- dbGetQuery(con, "select * from foo")
dbClearResult(rs) ## done with this query

dbCommit(con)     ## ok, everything looks fine
summary(ora)      ## a concise description of the driver
dbDisconnect(con) ## done with this connection
```

Summary

- R package enabling connectivity to Oracle Database
  - Open source
  - Publically available on CRAN
- Execute SQL statements from R interface
- Oracle Database Interface (DBI) for R
- DBI –compliant Oracle driver based on OCI
- Requirements
  - Oracle Instant Client or Oracle Database Client

Resources

- **Book:** Using R to Unlock the Value of Big Data
- **Blog:** https://blogs.oracle.com/R/
- **Forum:** https://forums.oracle.com/forums/forum.jspa?forumID=1397
- **Oracle R Distribution**
- **ROracle**
- **Oracle R Enterprise**
- **Oracle R Connector for Hadoop**

http://oracle.com/goto/R


ORACLE®