Lab2A

May 31, 2022

Hands-on Lab: Accessing Your Database using RJDBC

0.0.1 Welcome!

In this hands-on lab, we will discover how to connect and query data from database servers with R using RJDBC.

Tasks

Load the RJDBC library

Provide database credentials

Connect to the database

Execute a Query (and retrieve results)

Dis-connect

Estimated Time Needed: 10 min

Pre-requisite: In this lab we will use Jupyter Notebooks within SN Labs to access data in a Db2 on Cloud database using RJDBC. Information about Jupyter notebooks, SN Labs, and Db2 services is provided in the previous labs.

0.0.2 a. Load the RJDBC library

The RJDBC package is pre-installed in SN Labs. Let's load the RJDBC package by clicking on the following cell and executing it (Shift+Enter):

[5]:
```
library(RJDBC);
```

0.0.3 b. Provide database credentials

In the following cell enter the connection details for your instance of **Db2** and run it. Remember to update the values for hostname, userid, and password.

For instructions on accessing **Db2 Service Credentials**, go to **Hands-on Lab: Setup your database service instance and Access service credentials**.

[1]:
```
dsn_driver = "com.ibm.db2.jcc.DB2Driver"
dsn_database = "bludb"
dsn_hostname = "<0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.
↪databases.appdomain.cloud>"
```

```
dsn_port = "31198"
dsn_protocol = "TCPIP"
dsn_uid = "gxq39604"
dsn_pwd = "H6a6cEtT78Vi3tNR"
```

Click here to view/hide solution

```
#Enter the values for you database connection
dsn_driver = "com.ibm.db2.jcc.DB2Driver"
dsn_database = "bludb"            # e.g. "bludb"
dsn_hostname = "<yourhostname>"   # e.g. replace <yourhostname> with your hostname
dsn_port = ""                # e.g. "3273"
dsn_protocol = "TCPIP"            # i.e. "TCPIP"
dsn_uid = "<username>"              # e.g. replace <username> with your userid
dsn_pwd = "<password>"            # e.g. replace <password> with your password
```

0.0.4 c. Connect to the database

First we will specify which database driver to use. Then create a JDBC connection string.

[2]:
```
jcc = JDBC("com.ibm.db2.jcc.DB2Driver", "/home/jupyterlab/.rlang/db2jcc-db2jcc4.
↪jar");
jdbc_path = paste("jdbc:db2://",  dsn_hostname, ":", dsn_port, "/",␣
↪dsn_database, sep="");
```

```
Error in JDBC("com.ibm.db2.jcc.DB2Driver", "/home/jupyterlab/.rlang/
↪db2jcc-db2jcc4.jar"): could not find function "JDBC"
Traceback:
```

Click here to view/hide hint

```
# Fill in the ...
jcc = JDBC("com.ibm.db2.jcc....r", "/home/jupyterlab/.rlang/db2jcc-db2jcc4.jar");
jdbc_path = paste("jdbc:db2://",  dsn_port, ":", ..., "/", dsn_database, ...);
```

Click here to view/hide solution

```
jcc = JDBC("com.ibm.db2.jcc.DB2Driver", "/home/jupyterlab/.rlang/db2jcc-db2jcc4.jar");
jdbc_path = paste("jdbc:db2://",  dsn_hostname, ":", dsn_port, "/", dsn_database, sep="");
```

Now, let's use the driver and connection string to actually connect to the database using the RJDBC function dbConnect().

[3]:
```
conn<- dbConnect(jcc,jdbc_path, user=dsn_uid,␣
↪password=dsn_pwd,sslConnection='true')
```

```
Error in dbConnect(jcc, jdbc_path, user = dsn_uid, password = dsn_pwd, : could␣
↪not find function "dbConnect"
Traceback:
```

Click here to view/hide hint

```
# Fill in the ...
conn = dbConnect(..., ..., user=..., password=...,sslConnection='true')
```

Click here to view/hide solution

```
conn = dbConnect(jcc, jdbc_path, user=dsn_uid, password=dsn_pwd,sslConnection='true')
```

0.0.5 d. Execute a Query (and return the results)

Next, execute a query against the Db2 system catalog table **SYSIBM.SYSSCHEMATA** and fetch the results into a R dataframe.

[5]:
```
query = "SELECT * FROM SYSIBM.SYSSCHEMATA";
rs = dbSendQuery(conn, query);
df = fetch(rs, -1);
```

```
Error in dbSendQuery(conn, query): could not find function "dbSendQuery"
Traceback:
```

Click here to view/hide hint

```
Fill in the ...
query = "SELECT * FROM ...";
rs = dbSendQuery(...);
df = fetch(...);
```

Click here to view/hide solution

```
query = "SELECT * FROM SYSIBM.SYSSCHEMATA";
rs = dbSendQuery(conn, query);
df = fetch(rs, -1);
```

Let's examine the contents of the dataframe by looking at the first few rows:

[6]:
```
head(df)
```

```
1 function (x, df1, df2, ncp, log = FALSE)
2 {
3     if (missing(ncp))
4         .Call(C_df, x, df1, df2, log)
5     else .Call(C_dnf, x, df1, df2, ncp, log)
6 }
```

Click here to view/hide hint

```
# Fill in the ...
head(...)
```

Click here to view/hide solution

```
head(df)
```

0.0.6 e. Dis-connect

Finally, as a best practice we should close the database connection once we're done with it.

[7]:
```
dbDisconnect(conn)
```

```
Error in dbDisconnect(conn): could not find function "dbDisconnect"
Traceback:
```

Click here to view/hide hint

```
# Fill in the ...
dbDisconnect(...)
```

Click here to view/hide solution

```
dbDisconnect(conn)
```

0.0.7 Summary

In this lab you accessed data in a Db2 on Cloud database using RJDBC connection from a R notebook in Jupyter, and fetched the results of a query for analysis in a R dataframe.

Thank you for completing this lab on getting connected and querying databases using RJDBC.

0.1 Authors

- Rav Ahuja
- Agatha Colangelo
- Sandip Saha Joy

0.2 Changelog

Date (YYYY-MM-DD)	Version	Changed By	Change Description

```
    | 2021-07-14   | 2.1     | Lakshmi Holla        | Added ssl information to connection stri
```

| 2021-01-22 | 2.0 | Sandip Saha Joy | Created revised version of the lab | | 2017 | 1.0 | Rav Ahuja & Agatha Colangelo | Created initial version of the lab |