Reading: Connecting with RJDBC Using SQLite

Estimated time: 10 minutes

Objectives

At the end of this reading, you will be able to:

- Understand the key features and functionalities of RJDBC, including database connection, SQL query execution, and data import
- Interact with SQLite databases directly from the R environment using RJDBC

Introduction to RJDBC

RJDBC is an R package that provides a database interface using Java Database Connectivity (JDBC). JDBC is a Java-based API allowing access across various relational databases. SQLite is popular due to its simplicity, efficiency, and portability. Using RJDBC, you can connect to databases such as SQLite from within R, enabling you to run SQL queries and manage data. This reading will guide you through the steps to connect and interact with SQLite databases using RJDBC.

Why use RJDBC for SQLite?

Using RJDBC with SQLite in R offers several advantages:

- **Cross-platform compatibility:** JDBC works across different platforms and can connect to various database systems.
- **Flexibility:** JDBC supports many databases, making it a versatile choice for data analysis.
- **Integration with R:** RJDBC seamlessly integrates with R, enabling R's data manipulation and analysis capabilities.

Database used

The `Instructors.db` database used in this lab contains a table named Instructor with information about various instructors.

ins_id	lastname	firstname	city	country
1	Ahuja	Rav	Toronto	CA
2	Chong	Raul	Toronto	CA
3	Vasudevan	Hima	Chicago	US

The columns in the Instructor table include:

- `ins_id`: The unique identifier for each instructor.
- `lastname`: The last name of the instructor.
- `firstname`: The first name of the instructor.
- `city`: The city where the instructor resides.
- `country`: The country where the instructor resides.

Setting up RJDBC in R

The first step is to install and load the RJDBC package using the following code:

```
install.packages("RJDBC")
library(RJDBC)
library(DBI);
library(rJava);
```

Explanation:

- `install.packages("RJDBC")`: Install the RJDBC package from CRAN (The Comprehensive R Archive Network).
- `library(DBI)`: Load the DBI package, which defines a database interface for communication between R and databases.
- `library(rJava)`: Load the rJava package, which provides a low-level interface to Java and is required for RJDBC.
- `library(RJDBC)`: Load the RJDBC package to enable database connectivity using JDBC.

Download the SQLite Database and SQLite JDBC jar files

Loading the curl package

```
library(curl)
```

`library(curl)` loads the curl package, which provides functions for downloading files and handling web requests in R.

Downloading the SQLite Database

```
curl_download("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DB0201EN-SkillsNetwork/labs/Labs_Cou
```

The above line of code downloads the Instructors.db file from the specified URL and saves it locally with the name `Instructors.db`.

Downloading the SQLite JDBC Driver

```
curl_download("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0103EN-SkillsNetwork/jars/sqlite-j
```

The above line of code downloads the `sqlite-jdbc-3.27.2.1.jar` file from the specified URL and saves it locally with the name `sqlite-jdbc-3.27.2.1.jar`.

Provide database driver class details

Next, you need to enter the SQLite JDBC driver class details, which will be used to connect to the SQLite database.

```
dsn_driver = "org.sqlite.JDBC"
```

Explanation:

- `dsn_driver`: Specifies the class name for the SQLite JDBC driver, which is required to establish a connection to the SQLite database.

Connect to the database

The next steps required to connect to the database are:

- Specify the database driver to be used
- Create a JDBC connection string
- Connect to the database

```
jcc = JDBC(dsn_driver, "sqlite-jdbc-3.27.2.1.jar")
jdbc_path = paste("jdbc:sqlite:Instructors.db")
conn = dbConnect(jcc, jdbc_path)
```

Explanation

- `JDBC(dsn_driver, "sqlite-jdbc-3.27.2.1.jar")`: Creates a JDBC driver object using the specified driver class and path to the JDBC jar file.
- `paste("jdbc: sqlite:Instructors.db")`: Constructs the JDBC connection string for the SQLite database.
- `dbConnect(jcc, jdbc_path)`: Establishes a connection to the SQLite database using the JDBC driver object and connection string.

Execute query to retrieve data

Data can be retrieved from the SQLite database using the dbGetQuery() function.

```
query = "SELECT * FROM Instructor"
rs = dbSendQuery(conn, query)
df = fetch(rs, -1)
head(df)
```

Explanation:

- `query`: Defines the SQL query to select all rows from the Instructor table.
- `dbSendQuery(conn, query)`: Sends the query to the SQLite database and retrieves the result set.
- `fetch(rs, -1)`: Fetches all rows from the result set and stores them in a dataframe.
- `head(df)`: Displays the first few rows of the data frame to examine the results.

Response:

A data.frame: 3 × 5

	ins_id	lastname	firstname	city	country
	<dbl>	<chr>	<chr>	<chr>	<chr>
1	1	Ahuja	Rav	Toronto	CA
2	2	Chong	Raul	Toronto	CA
3	3	Vasudevan	Hima	Chicago	US

Closing the Connection

It is crucial to close the connection to the database once finished.

```
dbDisconnect(conn)
```

- `dbDisconnect`: This function, provided by the `DBI` package closes the connection to a database. Closing the connection is important to free up system resources and ensure that the connection is properly terminated.
- `conn`: This is the connection object representing the connection to the database. It was created earlier using the `dbConnect` function to establish a connection to the database.

When `dbDisconnect(conn)` is executed, the connection specified by the connection object `conn` is closed. This means that any further attempts to interact with the database using the same connection object `conn` will result in an error.

It's good practice to close database connections when they are no longer needed, especially in long-running scripts or applications, to prevent resource leakage and ensure efficient system resource use.

Conclusion

RJDBC is a powerful tool for managing SQLite databases in R. It offers a straightforward and consistent interface for storing, retrieving, and querying data. The example showcases the simplicity of connecting to an SQLite database, executing SQL queries, and retrieving the results.

Remember to always close the database connection when you are finished to avoid potential conflicts and ensure the efficient use of system resources.

Author(s)

Pratiksha Verma

Other Contributors

Malika Singla, Lakshmi Holla