Communicating with Databases via R

Nick Murphy

28 July 2021

Introduction

The R programming language is a popular tool to manipulate, sort, and display large amounts of data. In this class, we have worked with a variety of data sources - CSVs, JSON files, and "built-in" information. Unfortunately, these are all "static" data sets - you would need to replace the entire file to get new data. While that can be automated, there are still issues such as cached/"outdated" information and file size to contain an entire dataset.

In most work environments, data is instead pulled from something called a Relational Database System, or RDBS for short. As the name implies, it is a collection of information that's described via the relationships between each column of information. An easy way to visualize it is to imagine a giant Excel sheet, with columns of datatypes and individual rows of information. In this code-through, we'll talk about how we can utilize these via R command codes.

Learning Objectives

Specifically, you'll learn how about databases, how we interact with them directly, and how we can use that knowledge to pull information into our R applications.

Main Information

Databases have their own special languages to create, send, and retrieve information. This language is called SQL (pronounced either “Sequel” or “S.Q.L” - holy wars have been fought over which is correct). Like many programming languages, R has several libraries that can be used to connect to and interact with databases via SQL. This topic would be a class in and of itself, but we’ll go over the basic ideas here.

We’ll start by comparing SQL commands with related commands in R. First, the SELECT command. This is how you retrieve specific data from a “table” in the database (again, think a single sheet in Excel). An example of that command might look like:

SELECT id, fname, lname FROM faculty;

An equivalent command we’ve used in the dplyr library looks like this:

```
faculty %>%
  select(id, fname, lname)
```

This would get every first name, last name, and ID number from the “faculty” table in the database. To prevent Dr. Crawford from getting angry at us, we can limit the number of results using the LIMIT command:

SELECT id, fname, lname FROM faculty LIMIT 10;

Additionally, if we want to filter data by certain conditions, we can write a WHERE clause like this:

SELECT id, fname, lname FROM faculty WHERE age > 40;

The filter command in dplyr is one way to recreate this in R:

```
faculty %>%
  select(id, fname, lname) %>%
  filter(age > 40)
```

Much like in R, you can add multiple conditions using AND as well as OR conditions. Negation is applied using the word NOT.

Beyond merely selecting the data, we also want to be able to update information in the database. In SQL, this can be done with UPDATE commands. Samples of this look like:

INSERT INTO faculty (fname, lname) VALUES (“Jamison”, “Crawford”);

UPDATE faculty SET age = 30 WHERE fname = “Jim”

While this is a bit more exact than what you'd likely do in R, you can simulate it with different versions of the mutate function:

```
faculty %>%
  select(id, fname, lname) %>%
  mutate(age = 30)  %>%
  filter(fname == "Jim")
```

We could spend all day reviewing SQL, but one last relevant command is the JOIN functionality. We covered this in Lab 6 with the merge functionality - in SQL, this is explictedly done using various JOIN statements.

SELECT id, fname, lname FROM faculty JOIN students ON faculty.id JOIN student.id;

This Inner Join would create a merged table showing any teacher who is also a student at the university. You can replace "JOIN" with LEFT JOIN, RIGHT JOIN, and FULL JOIN for an Outer Join command.

As a final note, the above commands are just one "flavor" of SQL. Depending on what company created your database, the syntax for the SQL statements may vary a bit (e.g. "TOP 10" instead of "LIMIT 10"). The most common versions are Microsoft's MS SQL and the open source MySQL, but there are other variants like Oracle and PostGres. Make sure to consult with your Database Administrator to determine what version your company uses before you start a project.

Basic Example

Now that we have a small understanding of SQL and how to talk to databases on their terms, let's look over how we can connect to them using R. First, we need to download the appropriately libraries and set up our connection to the database.

library(DBI)
library(dplyr)
library(dbplyr)
library(odbc)

Once we've got that set up, we need to make an actual connection to the database. This goes far beyond the scope of this tutorial (although you can read more details here (https://db.rstudio.com/getting-started/connect-to-database/)), but if a connection has been set up for you, you can create a variable for it like this:

```
con <- dbConnect(odbc::odbc(), "Your Database Connection Here")
```

We can now use the **con** variable to pass our R commands to the database, and get the results back in a way that dplyr can then process for us. Like so:

```
data <- tbl(con, "faculty") %>%
  group_by(lname) %>%
  select(id, fname, lname)
show_query(data)
```

This sample query would go through the **con** to get to our database. It would then access the "faculty" table's information. After that, it all looks familar from the last few labs with group_by and select commands. Afterwards, we'd call "show_query()" to show the results of our SQL query.

This code-through has scratched the surface of working with relational databases with R. I would strongly encourage anyone who's interested in this topic and/or using R professionally to dive into it further. It will allow you to apply what we've learned about analyzing and visualizing data on a much larger scale, and it will give you some very marketable skills!

Further Resources

Learn more about SQL and how we access it in R with the following:

- More about Database Queries in R https://db.rstudio.com/getting-started/database-queries/ (https://db.rstudio.com/getting-started/database-queries/)
- General SQL tutorial https://zetcode.com/mysql/introduction/ (https://zetcode.com/mysql/introduction/)
- Details about how to set up a database connection in R [https://db.rstudio.com/getting-started/connect-to-database/ (https://db.rstudio.com/getting-started/connect-to-database/)] (https://db.rstudio.com/getting-started/connect-to-database/ (https://db.rstudio.com/getting-started/connect-to-database/))