Self-Study Exercises

Author

Fred Hutchinson Cancer Research Center

17-18 February, 2011

1 Introduction

This set of Exercises is intended to give you practice at using and building databases with SQL from R. You will get to practice building a small database, populate it with data, and then write functions that can retrieve data from it. You will even write a function to connect your data to other databases so that you can take full advanatage of Bioconductors modular system of databases. The code that you write in these exercises is meant to be used later in subsequent exercises, so try to make sure that you understand this material since you will build on it later.

2 Exercises

Exercise 1
For the 1st exercise you need to make a database that contains 2 tables. Create a database called metadata.sqlite and populate two tables in that DB. There are two .csv files included in the extdata directory of the AdvancedR2011 course that can be located with a call to system.file. Read these files into R using read.csv and then populate the database with two tables called subjects and snps respectively. While you do so, there will be some data in the snp information that you don't need. Be sure to exclude this or it will make your life more difficult later. You will need the following fields of data from each file:

From subjectData.csv: id, subject_id, case_control
From snpData.csv: id, seqnames, start, gene_id and snp_id

Question 1
Using the information above, write an R and SQL script to create the two-table database described above and then run it. When you are done, you should have an empty database with two tables. You should then check the contents of your database usinge dbListTables() and dbListFields().

Solution:

Exercise 2
Next we need to put the data from the data.frames in R into our new tables

Question 2
Using the information above from the descriptions above, populate your two tables selectively with the contents from the two .csv files provided. Remember that we created tables to hold only some of the data from these files. So we need to only grab data from the files that matches the fields we want to use.

From subjectData.csv: ID, subjectID, disease

From snpData.csv: ID, seqnames, start, gene_id and snpIds

Solution:

Exercise 3
For the purposes of efficient data retrieval later on, you may also want to create indices on any fields that contain a legitimate identifier. This is probably a good idea because real identifiers can potentially be used in another context to join to other data, and having such information indexed can vastly speed up these joining operations. You have to be aware though that there is still no such thing as a free lunch. Adding these indices speeds up joins at the cost of a database taking up additional storage space. For this example, we are not very concerned about the space requirements for this small database, so lets plan to put indices on to anything that looks like it could be used as a legitimate identifier later on.

Please also be aware that any primary keys declared when you initially defined a table will already be indexed by SQLite, so if you specified that in your initial create table statement, you won't need to do it again. It is possible to double (or, triple etc.) the space required by an index by defining it multiple times unecessarily.

Question 3
Using the information above, create indices for the tables you have populated for any field that is called an "id".

Solution:

So now that we are done with the task of making the database, we can close the connection to it:

```
> dbDisconnect(con)

[1] TRUE
```

Exercise 4
Most R users are not familiar with SQL and the intricacies of interacting with a relational database. So it is usually a good idea to provide useful accessor method to get certain data into R in a desired format.

Question 4
Place your new database into the extdata directory of your package and install it. Then write a pair of simple accessor functions to retrieve the contents of each table called getSnps() and getSubjects().

Solution:

Exercise 5
Sometimes it is useful to connect to another Database to get more information. For this exercise you need to join to another database to find out what kegg pathway IDs can be associated with the snps that you have in the "snps" table.

In this case, the database you want to use is the org.Hs.eg.sqlite databased from the human organism package org.Hs.eg.db. The org.Hs.eg.db package uses the "genes" table as it's central table, while the KEGG information you need from it is located in the "kegg" table. To retrieve this data, you will have to BOTH join to the genes table and also to the kegg table. This is because in order to get the kegg pathway IDs associated with the entrez gene IDs, the genes table has to be joined to the kegg table by using the internal "_id" that these two tables share. So ultimately this task will involve joing three tables: "snps" (local), "genes", and "kegg".

Question 5
Using the entrez gene IDs in the snps table as a foreign key, write a function to pull information from the org.Hs.eg.sqlite database found in the org.Hs.eg.dbs package. You can use system.file() to access this other database from the extdata directory in it's respective package.

Solution:

Exercise 6
The above function creates a dependency on another database, so it may be desirable to check that the other database we are depending on is current. The dates for the recent version of the org.Hs.eg.db database can be discovered by checking the contents of the metadata table inside of the org.Hs.eg.sqlite database. For R 2.12, the version of these KEGG pathway IDs has a "KEGGSOURCEDATE" of "2010-Sep7", which means that this data was pulled down from KEGG at that time. Choosing annotations from earlier or later than that time might make results less reproducible and could confound the analysis. Ideally, annotations should be updated periodically, but you always want to know about it when it happens. IOW, you never really want annotation updates to be a surprise.

Question 6
Write a function that checks this metadata table and reports whether or not the results are equivalent to the version needed for use with R 2.12.

Solution:

The version number of R and packages loaded for generating the vignette were:

```
R version 2.12.1 (2010-12-16)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8
 [2] LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=C
 [5] LC_MONETARY=C
 [6] LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8
 [8] LC_NAME=C
 [9] LC_ADDRESS=C
[10] LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8
[12] LC_IDENTIFICATION=C

attached base packages:
[1] grid      stats     graphics
[4] grDevices utils     datasets
[7] methods   base

other attached packages:
 [1] org.Hs.eg.db_2.4.6
 [2] AnnotationDbi_1.12.0
 [3] Biobase_2.10.0
 [4] RSQLite_0.9-4
 [5] DBI_0.2-5
 [6] ggplot2_0.8.9
 [7] proto_0.3-8
 [8] reshape_0.8.4
 [9] plyr_1.4
[10] codetoolsBioC_0.0.16

loaded via a namespace (and not attached):
[1] codetools_0.2-7 tools_2.12.1
```