Stat 8054 Lecture Notes: R and SQL Databases

Charles J. Geyer

March 01, 2023

1 License

2 R

- The version of R used to make this document is 4.2.1.
- The version of the `rmarkdown` package used to make this document is 2.20.
- The version of the `RSQLite` package used to make this document is 2.2.20.
- The version of the `DBI` package used to make this document is 1.1.3.

3 History

I am not an expert on the history of databases, but at least know there are phases to this history. Most of this relies on the Wikipedia article but has some different emphases.

The history is divided into four eras, or generations, which overlap:

- the dinosaur era (1960's) where there were large databases but they were clumsy to use and relied on highly complicated and usually buggy code written by highly trained programmers,
- the relational and SQL database era (1970's through 1990's) had the following features (not all of which arrived at the same time in the same products):
    - relational databases (Wikipedia article), which to users look like real math: stored tables act like mathematical relations that are "programmed" using mathematical logic via
    - SQL (acronym for *Structured Query Language* but pronounced like the English word "sequel"; Wikipedia article), a standardized computer language for relational database operations, a language just like R or C++ except just for database operations,
    - ACID (acronym for *Atomicity, Consistency, Isolation, Durability*, pronounced like the English word "acid"; Wikipedia article) which describes the highly reliable transactions that are found in modern so-called SQL databases, like Oracle (and many other products),
- the noSQL era (2000's) in which all of the great ideas of the relational database era were dropped, putting programmers back in the dinosaur era or worse, all in the name of scaling to internet scale (Wikipedia article), leading examples of which are Amazon's Dynamo, Apache Cassandra, CouchDB, MongoDB, Redis, HBase, and MemcacheDB,
- the newSQL era (now, Wikipedia article) has the best of both worlds, relational, SQL, ACID, and highly scalable, a leading example is Google Spanner.

So while in the 2000's it looked like SQL was old hat and all "data scientists" needed to learn about noSQL that is now looking dated, although a lot of web services run on noSQL databases.

A word about pronounciation: sometimes SQL is "sequel" and sometimes S-Q-L (sounding the letters). In "Microsoft SQL server", the SQL is always "sequel". In Oracle MySQL server, the SQL is always S-Q-L so this is pronounced "my-S-Q-L". This was originally open source software before acquired by Oracle; its free software successor (fork) is MariaDB.

4 SQLite

For learning SQL the greatest thing since sliced bread is SQLite, a relational database with full SQL support that runs as a user application. It is just a software library backed by a file on disk. So you can do little database applications with no expensive database. And you can learn on it.

The author of SQLite pronounces it S-Q-L-ite "like a mineral" but does not object to other pronounciations.

5 R and SQL and SQLite

The R package that talks to all SQL databases is CRAN package `DBI` (for database interface). The R package that makes SQLite available to R is CRAN package `RSQLite`.
Section 6 of my STAT 3701 lecture notes on data is about this. The example in this document was a homework problem in that course that had numerous hints. It is better as a straightforward example.

6 Reading

- The `RSQLite` package vignette
- The SQL Tutorial at w3schools.com
- There are a bazillion books on SQL. I don't have a particular recommendation.

7 An Example

We just happen to have an SQLite database to serve as an example

```
download.file("https://www.stat.umn.edu/geyer/8054/data/cran-info.sqlite",
    "cran-info.sqlite")
```

Note: On windows this command must be

```
download.file("https://www.stat.umn.edu/geyer/8054/data/cran-info.sqlite",
    "cran-info.sqlite", mode = "wb")
```

because otherwise Windows does not treat the file as binary but rather as a text file and messes it up.

We connect to it using R using R packages `DBI` and `RSQLite`.

```
library(DBI)
mydb <- dbConnect(RSQLite::SQLite(), "cran-info.sqlite")
```

Just to see what we have, we execute some simple SQL queries.

```
dbListTables(mydb)
```

```
## [1] "depends"  "imports"  "linking"  "suggests"
```

```
dbGetQuery(mydb, "SELECT * FROM depends LIMIT 20")
```

```
##        packto     packfrom
## 1      xtable           A3
## 2     pbapply           A3
## 3    abc.data          abc
## 4    quantreg          abc
## 5      locfit          abc
## 6         abc     abctools
## 7       abind     abctools
## 8    parallel     abctools
## 9        plyr     abctools
## 10      Hmisc     abctools
## 11       grid          abd
## 12     mosaic          abd
## 13     glasso     abundant
## 14 data.table       Ac3net
## 15      mhsmm          acc
## 16       mice accelmissing
## 17       pscl accelmissing
## 18    ggplot2    accessrmd
## 19     tcltk2      accrual
## 20  lubridate  accrualPlot
```

```r
dbGetQuery(mydb, "SELECT * FROM imports LIMIT 20")
```

```
##         packto    packfrom
## 1     magrittr    AATtools
## 2        dplyr    AATtools
## 3   doParallel    AATtools
## 4      foreach    AATtools
## 5      ggplot2      ABACUS
## 6        shiny      ABACUS
## 7         httr      abbyyR
## 8          XML      abbyyR
## 9         curl      abbyyR
## 10       readr      abbyyR
## 11        plyr      abbyyR
## 12    progress      abbyyR
## 13        Rcpp      abcADM
## 14     plotrix ABCanalysis
## 15        Rcpp     abclass
## 16    parallel     abclass
## 17        Rcpp    ABCoptim
## 18       readr       abcrf
## 19 matrixStats       abcrf
## 20      ranger       abcrf
```

```r
dbGetQuery(mydb, "SELECT * FROM suggests LIMIT 20")
```

```
##          packto   packfrom
## 1  randomForest         A3
## 2         e1071         A3
## 3     rmarkdown     ABACUS
## 4         knitr     ABACUS
## 5      testthat abbreviate
## 6      testthat     abbyyR
## 7     rmarkdown     abbyyR
## 8         knitr     abbyyR
## 9         lintr     abbyyR
## 10        knitr    ABC.RAP
## 11    rmarkdown    ABC.RAP
## 12        Rglpk    abclass
## 13       qpmadr    abclass
## 14     tinytest    abclass
## 15     testthat   ABCoptim
## 16         covr   ABCoptim
## 17      ggplot2   abctools
## 18     abc.data   abctools
## 19          car        abd
## 20      ggplot2        abd
```

```
dbGetQuery(mydb, "SELECT * FROM linking LIMIT 20")
```

```
##           packto      packfrom
## 1           Rcpp        abcADM
## 2             BH        abcADM
## 3           Rcpp       abclass
## 4  RcppArmadillo       abclass
## 5           Rcpp      ABCoptim
## 6           Rcpp         abcrf
## 7  RcppArmadillo         abcrf
## 8           Rcpp         abess
## 9      RcppEigen         abess
## 10          Rcpp           abn
## 11 RcppArmadillo           abn
## 12          Rcpp        abtest
## 13          Rcpp           acc
## 14 RcppArmadillo           acc
## 15          Rcpp accelerometry
## 16          Rcpp      acebayes
## 17 RcppArmadillo      acebayes
## 18            BH          ACEt
## 19 RcppArmadillo          ACEt
## 20          Rcpp          ACEt
```

The table `depends` lists CRAN packages in column `packfrom` and in column `packto` lists other CRAN packages on which they depend (this does not include R core or recommended packages).

In R one might store data like this in R lists. Object `depends` would be a list with one component for each CRAN package, which would be a a character vector (perhaps of length zero) of all CRAN packages on which that package depends. We could use the `names` attribute of the list to indicate the “from” package. We could do this in code by

```
foo <- dbGetQuery(mydb, "SELECT * FROM depends")
depends <- split(foo$packto, foo$packfrom)
head(depends)
```

```
## $A3
## [1] "xtable"  "pbapply"
##
## $abc
## [1] "abc.data" "quantreg" "locfit"
```

```
##
## $abctools
## [1] "abc"      "abind"    "parallel" "plyr"     "Hmisc"
##
## $abd
## [1] "grid"   "mosaic"
##
## $abundant
## [1] "glasso"
##
## $Ac3net
## [1] "data.table"
```

```
rm(foo, depends)
```

But in a SQL database, everything must be a table. No lists. Hence we have a table whose rows are all from-to pairs.

We want to process these data as if we could not fit it all into R (which is false for this toy problem but might be true for big data) so we have to use the SQL database to do all operations until we get down to small results we can return to R.

I could not figure out how to do this in one SQL command, so I had to create tempory thingummies, which in a SQL database must be tables, since everything in a SQL database is a table.

```
query <- paste("CREATE TABLE temp AS",
               "SELECT packto FROM depends",
               "UNION ALL",
               "SELECT packto FROM imports",
               "UNION ALL",
               "SELECT packto FROM suggests",
               "UNION ALL",
               "SELECT packto FROM linking")
dbExecute(mydb, query)
```

```
## [1] 0
```

```
dbGetQuery(mydb, "SELECT * FROM temp LIMIT 10")
```

```
##      packto
## 1    xtable
## 2   pbapply
## 3  abc.data
## 4  quantreg
## 5    locfit
## 6       abc
## 7     abind
## 8  parallel
## 9      plyr
## 10    Hmisc
```

The `dbGetQuery` command is just to see what we got (to check that we actually did what we thought we did.

```
query <- paste("CREATE TABLE temptoo AS",
               "SELECT packto, COUNT(packto) AS packcount",
               "FROM temp GROUP BY packto")
dbExecute(mydb, query)
```

```
## [1] 0
```

```
dbGetQuery(mydb, "SELECT * FROM temptoo LIMIT 10")
```

```
##         packto packcount
## 1  ABCanalysis         5
## 2         ACDm         1
## 3       ADAPTS         1
## 4    ADGofTest        10
## 5         ADMM         1
## 6     ADPclust         1
## 7          AER        65
## 8          AGD         1
## 9    AGHmatrix         1
## 10      AHSurv         1
```

Looks OK again.

```
query <- paste("SELECT * from temptoo WHERE packcount >= 100",
               "ORDER by packcount DESC")
dbGetQuery(mydb, query)
```

```
##             packto packcount
## 1            knitr      7390
## 2         testthat      7148
## 3        rmarkdown      6795
## 4             Rcpp      5134
## 5          ggplot2      4250
## 6            dplyr      3566
## 7         magrittr      2085
## 8             covr      2020
## 9            rlang      1806
## 10          tibble      1701
## 11           tidyr      1681
## 12         stringr      1661
## 13           purrr      1411
## 14        parallel      1398
## 15      data.table      1314
## 16        jsonlite      1188
## 17   RcppArmadillo      1057
## 18           shiny      1050
## 19            httr      1007
## 20         mvtnorm       875
## 21         foreach       796
## 22          scales       775
## 23            plyr       766
## 24          igraph       747
## 25        reshape2       735
## 26       lubridate       673
## 27      doParallel       643
## 28              sp       628
## 29            grid       627
## 30       gridExtra       616
## 31           readr       612
## 32        spelling       599
## 33    RColorBrewer       575
## 34            glue       571
## 35              sf       527
## 36            xml2       510
## 37          raster       501
## 38             zoo       464
## 39        markdown       457
## 40              R6       435
## 41          glmnet       428
## 42            curl       425
## 43       htmltools       425
## 44           withr       410
## 45            lme4       409
## 46           tools       405
## 47       RcppEigen       398
## 48            coda       385
## 49         stringi       384
## 50        devtools       372
## 51             cli       367
## 52          crayon       354
## 53        roxygen2       353
## 54              DT       347
## 55        numDeriv       339
## 56          Rdpack       336
## 57           rgdal       333
## 58          digest       332
## 59          plotly       332
## 60           R.rsp       330
## 61      tidyselect       326
## 62             ape       319
## 63      rstudioapi       319
## 64          pracma       314
## 65              BH       309
## 66      assertthat       303
## 67           e1071       302
## 68             rgl       300
## 69           Hmisc       291
## 70    randomForest       287
## 71             car       281
## 72    RcppParallel       275
## 73          readxl       273
## 74     htmlwidgets       266
## 75         ggrepel       264
## 76       checkmate       256
## 77             DBI       249
## 78           rstan       248
## 79           abind       246
## 80             XML       240
## 81        progress       240
## 82         splines       237
## 83       tidyverse       232
## 84      kableExtra       231
## 85       lifecycle       230
## 86           caret       229
## 87          gtools       229
## 88          xtable       225
## 89             png       223
## 90         forcats       222
## 91         cowplot       220
## 92        tinytest       217
## 93     matrixStats       216
## 94            yaml       215
## 95          fields       212
## 96          future       212
## 97         pbapply       212
## 98           rgeos       207
## 99           broom       205
## 100          psych       205
## 101          vctrs       201
## 102          rvest       199
## 103         vdiffr       195
## 104        RSQLite       190
## 105           maps       189
## 106          RCurl       186
## 107             fs       184
## 108        shinyjs       183
## 109            xts       182
## 110     colorspace       181
## 111          tcltk       180
## 112       sandwich       179
## 113      patchwork       178
## 114          vegan       175
## 115        viridis       173
## 116        Formula       172
## 117        R.utils       169
## 118        usethis       166
## 119       forecast       165
## 120       maptools       158
## 121     reticulate       158
## 122         lmtest       154
## 123       quantreg       152
## 124        leaflet       151
## 125        plotrix       151
## 126         ranger       150
## 127     robustbase       146
## 128 shinydashboard       144
## 129         mclust       143
## 130         ggpubr       141
## 131         lavaan       136
## 132       quadprog       136
## 133         magick       135
## 134 microbenchmark       133
## 135       openxlsx       128
## 136         survey       126
## 137          rJava       125
## 138       bookdown       123
## 139          haven       123
## 140         nloptr       123
## 141        deSolve       121
## 142       kernlab       121
## 143      base64enc       120
## 144    shinyWidgets       119
## 145    RcppProgress       118
## 146          VGAM       117
## 147       pkgdown       116
## 148       corpcor       115
## 149         lintr       115
## 150       reshape       113
## 151        GGally       112
## 152      generics       112
## 153     iterators       112
## 154       xgboost       111
## 155          pROC       110
## 156 spatstat.geom       110
## 157  future.apply       109
## 158   StanHeaders       108
## 159        gplots       108
## 160       memoise       107
## 161          mice       107
## 162        rjags       107
## 163       statmod       106
## 164         RUnit       105
## 165        pander       105
## 166      rappdirs       101
## 167      MCMCpack       100
## 168        miniUI       100
## 169        stats4       100
## 170         terra       100
```

CRAN packages that are used by at least 100 other CRAN packages. In descending order.

Clean up.

```
dbListTables(mydb)
```

```
## [1] "depends"  "imports"  "linking"  "suggests" "temp"     "temptoo"
```

```
dbExecute(mydb, "DROP TABLE temp")
```

```
## [1] 0
```

```
dbExecute(mydb, "DROP TABLE temptoo")
```

```
## [1] 0
```

```
dbListTables(mydb)
```

```
## [1] "depends"  "imports"  "linking"  "suggests"
```