Lecture 17: Databases

Statistical Computing, 36-350

Wednesday November 18, 2015

Outline

- What databases are, and why
- SQL
- Interfacing R and SQL

Databases

- A **record** is a collection of **fields**
- A **table** is a collection of records which all have the same fields (with different values)
- A **database** is a collection of tables

Databases versus data frames

- Data frames in R are actually tables

R jargon	Database jargon
column	field
row	record
data frame	table
types of the columns	table **schema**
bunch of related data frames	database

So, why do we need database software?

- **Size**
  - R keeps its data frames in memory
  - Industrial databases can be much bigger
  - Work with selected subsets
- **Speed**
  - Clever people have worked very hard on getting just what you want fast (Turing award winner Michael Stonebraker!)
- **Concurrency**
  - Many users accessing the same database simultaneously
  - Lots of potential for trouble (two users want to change the same record at once)

The client-server model

- Databases live on a **server**, which manages them
- Users interact with the server through a **client** program
- Lets multiple users access the same database simultaneously

SQL

- **SQL** (**structured query language**) is the standard for database software
- Mostly about **queries**, which are like doing a selection in R

```
debt[debt$Country=="France",c("growth","ratio")]
with(debt,debt[Country=="France",c("growth","ratio")])
subset(debt,subset=(Country=="France"),select=c("growth","ratio"))
```

- We will see shortly how SQL does stuff like this

Connecting R to SQL

First, though let's see how to connect SQL to R. Note:

- SQL is a language; database management systems (DMBS) actually implement it and do the work
  - MySQL, SQLite, etc.,
- They all have somewhat different conventions
- The R package `DBI` is a unified interface to them
- Need a separate "driver" for each DBMS

Before running the following, install the packages: `DBI`, `RSQLite`. Also, download the database file http://www.stat.cmu.edu/~ryantibs/statcomp/lectures/baseball.db, and save it in your working directory.

```
library(DBI)
library(RSQLite)
drv = dbDriver("SQLite")
con = dbConnect(drv, dbname="baseball.db")
```

`con` is now a persistent connection to the database `baseball.db`

```
dbListTables(con)                    # Get tables in the database
```

```
##  [1] "AllstarFull"         "Appearances"         "AwardsManagers"
##  [4] "AwardsPlayers"       "AwardsShareManagers" "AwardsSharePlayers"
##  [7] "Batting"             "BattingPost"         "Fielding"
## [10] "FieldingOF"          "FieldingPost"        "HallOfFame"
```

```
## [13] "Managers"            "ManagersHalf"        "Master"
## [16] "Pitching"            "PitchingPost"        "Salaries"
## [19] "Schools"             "SchoolsPlayers"      "SeriesPost"
## [22] "Teams"               "TeamsFranchises"     "TeamsHalf"
## [25] "sqlite_sequence"     "xref_stats"
```

```
dbListFields(con, "Batting")          # List fields in table Batting
```

```
##  [1] "playerID"  "yearID"    "stint"     "teamID"    "lgID"
##  [6] "G"         "G_batting" "AB"        "R"         "H"
## [11] "2B"        "3B"        "HR"        "RBI"       "SB"
## [16] "CS"        "BB"        "SO"        "IBB"       "HBP"
## [21] "SH"        "SF"        "GIDP"      "G_old"
```

```
batting = dbReadTable(con, "Batting") # Import a table as a data frame
dim(batting)
```

```
## [1] 93955    24
```

SELECT

The main tool in the SQL language is `SELECT`, which allows you to perform queries on a particular table in a database. It has the form:

```
SELECT columns or computations
  FROM table
  WHERE condition
  GROUP BY columns
  HAVING condition
  ORDER BY column [ASC | DESC]
  LIMIT offset,count;
```

Example: picking out columns

Suppose we want to pick out five columns from the table `Batting`, and we only want to look at the first 10 rows.

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "LIMIT 10"))
```

```
##     playerID yearID  AB   H HR
## 1  aardsda01   2004   0   0  0
## 2  aardsda01   2006   2   0  0
## 3  aardsda01   2007   0   0  0
## 4  aardsda01   2008   1   0  0
## 5  aardsda01   2009   0   0  0
## 6  aaronha01   1954 468 131 13
```

```
## 7  aaronha01   1955 602 189 27
## 8  aaronha01   1956 609 200 26
## 9  aaronha01   1957 615 198 44
## 10 aaronha01   1958 601 196 30
```

This is our very first SQL query (congrats!). It was very efficient

---

Note that we can replicate this command on the data frame `batting`:

```
batting[1:10, c("playerID", "yearID", "AB", "H", "HR")]
```

```
##     playerID yearID  AB   H HR
## 1  aardsda01   2004   0   0  0
## 2  aardsda01   2006   2   0  0
## 3  aardsda01   2007   0   0  0
## 4  aardsda01   2008   1   0  0
## 5  aardsda01   2009   0   0  0
## 6  aaronha01   1954 468 131 13
## 7  aaronha01   1955 602 189 27
## 8  aaronha01   1956 609 200 26
## 9  aaronha01   1957 615 198 44
## 10 aaronha01   1958 601 196 30
```

This was simply to check our work, and we wouldn't actually want to do this on a large database, since it'd be much more inefficient to first read into an R data frame, and then call R commands)

Likewise, throughout this lecture, we'll be writing R code to check our SQL code, but keep in mind this is just for the sake of learning (not that you would do this in practice)

Practice problems

Enter your unique ID here:

Work through the following problems (go ahead and fill in the code below). In particular, for each of the following, explain the SQL commands, and replicate the results using R commands that you write

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "ORDER BY yearID",
                      "LIMIT 10"))
```

```
##     playerID yearID  AB  H HR
## 1  abercda01   1871   4  0  0
## 2   addybo01   1871 118 32  0
## 3  allisar01   1871 137 40  0
## 4  allisdo01   1871 133 44  2
## 5  ansonca01   1871 120 39  0
## 6  armstbo01   1871  49 11  0
## 7  barkeal01   1871   4  1  0
## 8  barnero01   1871 157 63  0
## 9  barrebi01   1871   5  1  0
## 10 barrofr01   1871  86 13  0
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "ORDER BY HR DESC",
                      "LIMIT 10"))
```

```
##     playerID yearID  AB   H HR
## 1  bondsba01   2001 476 156 73
## 2  mcgwima01   1998 509 152 70
## 3   sosasa01   1998 643 198 66
## 4  mcgwima01   1999 521 145 65
## 5   sosasa01   2001 577 189 64
## 6   sosasa01   1999 625 180 63
## 7  marisro01   1961 590 159 61
## 8   ruthba01   1927 540 192 60
## 9   ruthba01   1921 540 204 59
## 10  foxxji01   1932 585 213 58
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "WHERE HR > 55",
                      "ORDER BY HR DESC"))
```

```
##     playerID yearID  AB   H HR
## 1  bondsba01   2001 476 156 73
## 2  mcgwima01   1998 509 152 70
## 3   sosasa01   1998 643 198 66
## 4  mcgwima01   1999 521 145 65
## 5   sosasa01   2001 577 189 64
## 6   sosasa01   1999 625 180 63
## 7  marisro01   1961 590 159 61
## 8   ruthba01   1927 540 192 60
## 9   ruthba01   1921 540 204 59
## 10  foxxji01   1932 585 213 58
## 11 greenha01   1938 556 175 58
## 12 howarry01   2006 581 182 58
## 13 gonzalu01   2001 609 198 57
## 14 rodrial01   2002 624 187 57
## 15 griffke02   1997 608 185 56
## 16 griffke02   1998 633 180 56
## 17 wilsoha01   1930 585 208 56
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "WHERE yearID >= 1990 AND yearID <= 2000",
                      "ORDER BY HR DESC",
                      "LIMIT 10"))
```

```
##     playerID yearID  AB   H HR
## 1  mcgwima01   1998 509 152 70
## 2   sosasa01   1998 643 198 66
## 3  mcgwima01   1999 521 145 65
## 4   sosasa01   1999 625 180 63
## 5  griffke02   1997 608 185 56
## 6  griffke02   1998 633 180 56
## 7  mcgwima01   1996 423 132 52
## 8  fieldce01   1990 573 159 51
## 9  anderbr01   1996 579 172 50
## 10 belleal01   1995 546 173 50
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT AVG(HR)",
                      "FROM Batting"))
```

```
##    AVG(HR)
## 1 2.970549
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT SUM(HR)",
                      "FROM Batting"))
```

```
##   SUM(HR)
## 1  260431
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT teamID, yearID, playerID, MAX(HR)",
                      "FROM Batting"))
```

```
##   teamID yearID  playerID MAX(HR)
## 1    SFN   2001 bondsba01      73
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID >= 1990"))
```

```
##    AVG(HR)
## 1 4.199555
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT teamID, AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID >= 1990",
                      "GROUP BY teamID",
                      "LIMIT 5"))
```

```
##   teamID  AVG(HR)
## 1    ANA 4.678445
## 2    ARI 3.849315
## 3    ATL 4.113379
## 4    BAL 5.152174
## 5    BOS 5.126227
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT teamID, AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID >= 1990",
                      "GROUP BY teamID",
                      "ORDER BY AVG(HR) DESC",
                      "LIMIT 5"))
```

```
##   teamID  AVG(HR)
## 1    CHA 6.164251
## 2    NYA 5.986486
## 3    TOR 5.760937
## 4    CAL 5.625731
## 5    TEX 5.563961
```

```
# What's going on here? R equivalent on batting data frame?
```

```
dbGetQuery(con, paste("SELECT teamID, yearID, AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID == 1991 OR yearID == 1992",
                      "GROUP BY teamID, yearID",
                      "ORDER BY AVG(HR) DESC",
                      "LIMIT 15"))
```

```
##    teamID yearID  AVG(HR)
## 1     DET   1991 7.740741
## 2     DET   1992 7.280000
## 3     NYA   1992 7.086957
## 4     TOR   1992 7.086957
## 5     BAL   1991 6.800000
## 6     NYA   1991 6.681818
## 7     CHA   1991 6.619048
## 8     BAL   1992 5.920000
## 9     CLE   1992 5.772727
## 10    BOS   1991 5.727273
## 11    MIN   1991 5.600000
## 12    TEX   1991 5.531250
## 13    ML4   1991 5.523810
## 14    TOR   1991 5.320000
## 15    SEA   1992 5.137931
```

```
# What's going on here? R equivalent on batting data frame?
```

Summary

- A database is basically a way of dealing efficiently with lots of potentially huge data frames
- SQL is the standard language for telling databases what to do, especially what queries to run
- Pretty much everything in an SQL query is something we've practiced already in R
  - subsetting/selection, aggregation, merging, ordering
- Connect R to the database, send it an SQL query, analyse the returned data frame