Intro to ODBC with with SQLite

Revolution Analytics

1 Database access with SQL

Overview

In this session we cover importing and exporting data. The objectives are:

- Introduce remote database access and use of SQL syntax from within R environment.

Outline

Using SQL inside R

In this section we will only cover basic SQL commands, e.g. `select * from table` and `where` conditions.

We use the `RSQLite` package. SQLite is a light-weight SQL engine. According to the website, *SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine*.

```
require(RSQLite) || {
  install.packages("RSQLite")
  require(RSQLite)
}
```

The airlines data

In this course we use the famous airlines data, published by the ASA during 2009 as a data visualisation competition.

The data consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008.

This is a large dataset: there are nearly 120 million records in total, and takes up 1.6 gigabytes of space compressed and 12 gigabytes when uncompressed.

For this session, we only use data for 1987, i.e. one year of US flights.

Directory Setup

```
dataPath <- "../data"
bigDataPath <- Sys.getenv("ACADEMYR_BIG_DATA_PATH")
if (bigDataPath == "") {
  bigDataPath <- Sys.setenv(ACADEMYR_BIG_DATA_PATH = "/usr/share/BigData")
}
outdir <- "../output"
if (!file.exists(outdir)) dir.create(outdir)
```

Connect to data source

```
library("RSQLite")
db <- dbConnect(SQLite(), dbname = file.path(bigDataPath, "airlines.sqlite"))
```

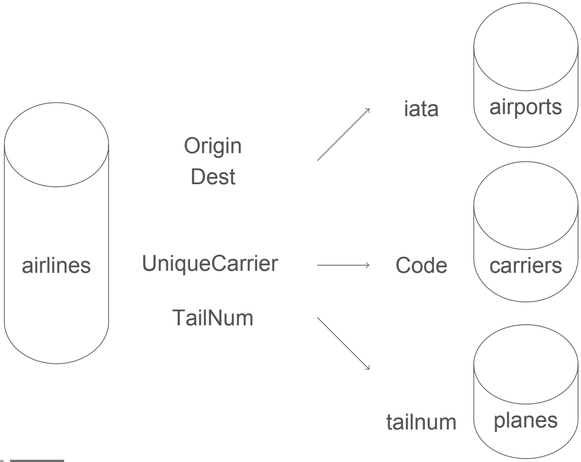
List tables in database

```
dbListTables(db)
```

```
## [1] "airlines" "airports" "carriers" "planes"
```

The tables in airlines SQL database

The four tables in airlines data:



Explore individual tables

```r
dbListFields(db, "airports")
```

```
## [1] "iata"    "airport" "city"    "state"   "country" "lat"     "long"
```

```r
dbListFields(db, "carriers")
```

```
## [1] "Code"        "Description"
```

```r
dbListFields(db, "planes")
```

```
## [1] "tailnum"       "type"          "manufacturer"  "issue_date"
## [5] "model"         "status"        "aircraft_type" "engine_type"
## [9] "year"
```

REVOLUTION ANALYTICS

Exercise:

- Identify the fields in the table that we have not looked at yet.

Get size of table

```
query <- "SELECT count(*) FROM carriers"
dbGetQuery(db, query)
```

```
##   count(*)
## 1     1491
```

Exercise 2:

- Which table has the most rows?

When things go wrong

```
dbListResults(db)
dbClearResult(dbListResults(db)[[1]])
```

Fetching data: the hard way

The “hard” way to retrieve large data sets into memory is to retrieve by chunks. Note that this may be the only way if your data is really big.

```
query <- "SELECT * FROM carriers"
res <- dbSendQuery(db, query)
x <- fetch(res, n = -1)
str(x)
```

```
## 'data.frame':    1491 obs. of  2 variables:
##  $ Code       : chr  "02Q" "04Q" "05Q" "06Q" ...
## $ Description: chr  "Titan Airways" "Tradewind Aviation" "Comlux Aviation, AG" "Master Top Linhas
```

Fetching data: more on the hard way

Can retrieve additional information from the results of the query

```
dbListResults(db)
```

```
## [[1]]
## <SQLiteResult: DBI RES (716, 1, 8)>
```

```
print(res)
```

```
## <SQLiteResult: DBI RES (716, 1, 8)>
```

Fetching data: sequential reads

- What would happen if we tried to fetch more records from the current query?

Fetching data: sequential reads

- What would happen if we tried to fetch more records from the current query?

```
try(x2 <- fetch(res, n = -1))
try(str(x2))
```

```
## 'data.frame':    0 obs. of  0 variables
```

Fetching data: Closing the query

```
dbClearResult(res)
```

```
## [1] TRUE
```

Exercise 3

- Can you do 3 sequential reads of the airlines data set, each of which has 2,000 records in it?
- Show that they read in different records.
- Close the connection to the query results.

Fetching data: the easy way

```
query <- "SELECT * FROM carriers"
x <- dbGetQuery(db, query)
str(x)
```

```
## 'data.frame':    1491 obs. of  2 variables:
##  $ Code       : chr  "02Q" "04Q" "05Q" "06Q" ...
##  $ Description: chr  "Titan Airways" "Tradewind Aviation" "Comlux Aviation, AG" "Master Top Linhas
```

```
head(x)
```

```
##   Code                     Description
## 1  02Q                   Titan Airways
## 2  04Q              Tradewind Aviation
## 3  05Q             Comlux Aviation, AG
## 4  06Q Master Top Linhas Aereas Ltd.
## 5  07Q             Flair Airlines Ltd.
## 6  09Q                  Swift Air, LLC
```

Interim summary of Intro to SQL

- RSQLite
- dbConnect(), dbListTables(), dbListFields()
- queries and dbGetQuery()
- hard way to get rows: dbSendQuery(); fetch(); dbClearResult()

Any quetions up to this point?

Frequent problem: do not want *ALL* rows

Just another (more complex) query: include a "WHERE" component

```
query <- "SELECT * FROM airlines where Dest=\"DAL\""
x <- dbGetQuery(db, query)
nrow(x)
```

```
## [1] 9090
```

More general “WHERE”: use a wild card

```
query <- "SELECT * FROM carriers where Description LIKE \"American%\""
x <- dbGetQuery(db, query)
x
```

```
##     Code                  Description
## 1     AA       American Airlines Inc.
## 2    AFA  American Flag Airlines Inc.
## 3    AFG        American Flight Group
## 4    AMI   American Inter-Island Inc.
## 5    AMT       American Air Transport
## 6     MQ American Eagle Airlines Inc.
...
```

Use a wild card search

```r
query <- 'SELECT * FROM carriers where Description LIKE "%American%"'
x <- dbGetQuery(db, query)
x
```

```
##      Code                          Description
## 1      AA                 American Airlines Inc.
## 2     AFA            American Flag Airlines Inc.
## 3     AFG                American Flight Group
## 4     AMA           North American Airlines Inc.
## 5     AMI            American Inter-Island Inc.
## 6     AMT                 American Air Transport
...
```

More complex analysis

- Produce a map of all of the cities served by American Airlines

Where to start?

How to identify unique information

Step 1. identify what airports are served by AA

Remind ourselves of columns in airlines

```
dbListFields(db, "airlines")
```

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
...
```

Unique airport codes served by AA

```r
query <- 'SELECT DISTINCT Origin FROM airlines
  WHERE UniqueCarrier = "AA"'
aaCities <- dbGetQuery(db, query)
str(aaCities)
```

```
## 'data.frame':    116 obs. of  1 variable:
##  $ Origin: chr  "JFK" "LAX" "HNL" "OGG" ...
```

Unique airport codes served by AA v2

```r
query <- 'SELECT DISTINCT Origin as apcode
  FROM airlines WHERE UniqueCarrier = "AA"'
aaCities <- dbGetQuery(db, query)
str(aaCities)
```

```
## 'data.frame':    116 obs. of  1 variable:
##  $ apcode: chr  "JFK" "LAX" "HNL" "OGG" ...
```

Identify mapping between code and city

Remind ourselves of columns in airports

```r
dbListFields(db, "airports")
```

```
## [1] "iata"     "airport" "city"     "state"    "country" "lat"      "long"
```

The airports table

```
query <- "SELECT * FROM airports"
airports <- dbGetQuery(db, query)
head(airports)
```

```
##   iata              airport             city state country   lat    long
## 1  00M              Thigpen      Bay Springs    MS     USA 31.95  -89.23
## 2  00R Livingston Municipal       Livingston    TX     USA 30.69  -95.02
## 3  00V          Meadow Lake Colorado Springs    CO     USA 38.95 -104.57
## 4  01G         Perry-Warsaw            Perry    NY     USA 42.74  -78.05
## 5  01J     Hilliard Airpark         Hilliard    FL     USA 30.69  -81.91
## 6  01M    Tishomingo County          Belmont    MS     USA 34.49  -88.20
```

Merge airline origin with airports, using R

```
aaAirportLocs <- merge(x = airports, by.x = "iata", y = aaCities,
  by.y = "apcode")
dim(aaAirportLocs)
```

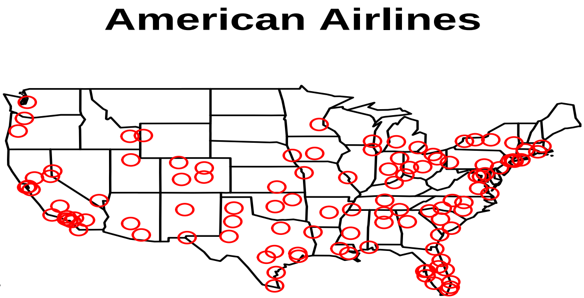
```
## [1] 116   7
```

```
head(aaAirportLocs)
```

```
##   iata                              airport        city state country   lat
## 1  ABQ          Albuquerque International Albuquerque    NM     USA 35.04
## 2  ALB                          Albany Cty      Albany    NY     USA 42.75
## 3  AMA             Amarillo International    Amarillo    TX     USA 35.22
## 4  ANC Ted Stevens Anchorage International   Anchorage    AK     USA 61.17
## 5  ATL    William B Hartsfield-Atlanta Intl     Atlanta    GA     USA 33.64
## 6  AUS       Austin-Bergstrom International      Austin    TX     USA 30.19
...
```

Plot on map

```
library(maps)
map("state")
title("American Airlines")
with(aaAirportLocs, points(long, lat, col = "red"))
```



{

}

A word about data processing

- Database engines are designed to efficiently run queries
- R is designed to perform data analysis

In general, do most of the data manipulation in SQL

How could we do these steps with SQL?

Reminder: Getting unique Origins

```
query <- "SELECT DISTINCT Origin FROM airlines WHERE UniqueCarrier = \"AA\""
x <- dbGetQuery(db, query)
str(x)
```

```
## 'data.frame':    116 obs. of  1 variable:
##  $ Origin: chr  "JFK" "LAX" "HNL" "OGG" ...
```

Merge origin with airports, using JOIN

```
query <- '
  SELECT *
  FROM airports
  JOIN (
    SELECT DISTINCT Origin FROM airlines
    WHERE UniqueCarrier = "AA"
  ) as airlines
  ON airlines.Origin = airports.iata
'
x <- dbGetQuery(db, query)
str(x)
```

```
## 'data.frame':    116 obs. of  8 variables:
##  $ iata   : chr  "JFK" "LAX" "HNL" "OGG" ...
##  $ airport: chr  "John F Kennedy Intl" "Los Angeles International" "Honolulu International" "Kahulu
##  $ city   : chr  "New York" "Los Angeles" "Honolulu" "Kahului" ...
##  $ state  : chr  "NY" "CA" "HI" "HI" ...
##  $ country: chr  "USA" "USA" "USA" "USA" ...
##  $ lat    : num  40.6 33.9 21.3 20.9 32.9 ...
...
```

REVOLUTION ANALYTICS

Exercise 4

- Retrieve the necessary data and plot a bar chart of the number of observations (flights) from SFO to each destination.
- Select and plot just the top 10 destinations by observation count.

Module review questions

- Why is it useful to pass SQL commands to a database from within R, rather than import the data and manipulate from the R environment?

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR