借助RSQLite包在R中执行SQL语句

王柳盈

2017年1月23日

### SQL:一种与数据库通信的语言

关系型数据库（RDBMS）是基于关系模型的特定类型数据库管理管理系统，它的强大在于足够通用和灵活，可以保存**所有**不同类型的数据结构。目前流行的RDBMS有Oracle, MySQL, Microsoft SQL Server, SQLite等。

SQL则是与RDBMS通信的**通用**语言，在刚刚提到的所有DBMS里，同一句SQL语句总能得到相同的结果。

SQL语言可用于如下任务：

* *数据控制语言(Data Control Language, DCL)* 进行精细的用户权限定义
* *数据定义语言(Data Definition Language, DDL)* 定义数据结构及其关系
* *数据操作语言(Data Manipulation Language, DML)* 填写和提取信息
* *事务控制语言(Transacion Control Language,TCL)* 提交或回滚之前的查询操作

……好我猜大家看定义已经看烦了（其实是我不想写了），说了这么多其实中心思想就是：**这货很有用！** 让我们赶紧进入正题吧！

### RSQLite

在R中有多个可以连接DBMS的包，最常见的是RSQLite和RODBC。后者需要依赖ODBC(Open Database Connectivity， 开放数据库连接)驱动，用人话说就是用之前还要装一些驱动程序，所以本文我们不用它！但是它也很好用，同样可以用来处理CSV，XLS/XLSX格式（导入EXCEL文件就要用它啦~）。我们使用RSQLite包，它依赖于R中原有的DBI组件，是迄今为止唯一能做到**跨多个平台开机即用**的组件包。

### 示例数据

我们用的数据是由Francisco S. Velazquez采集的Pokemon数据，真是无（meng）聊（dao）透（bao）顶（zha）。

### 开工！

#### 连接

首先我们要做的当然是加载package连接到数据库啦

library(RSQLite)

## Warning: package 'RSQLite' was built under R version 3.3.2

sqlite <- dbDriver("SQLite")  
file.exists("pokedex.sqlite")

## [1] TRUE

con <- dbConnect(sqlite, "pokedex.sqlite")

dbDriver用来指定所使用的驱动，dbConnect用来建立连接信息

#### 一些简单操作

有一些简单操作其实不需要用到SQL语言，但是有必要交代一下

dbGetInfo(con)#获取数据库基本信息  
dbListTables(con)#列出数据库中的所有表  
  
##事务控制  
dbBeginTransaction(con)#设置保存点  
dbRollback(con)#回滚操作  
dbCommit(con)#提交操作

解释一下事务控制。一般而言，我们对数据库发出的修改命令都是永久性的，但事务控制支持我们设置一个保存点，然后对数据进行临时修改，如果发生错误，使用回滚命令即可回复到最近一个保存点；如确认进行修改，则可进行提交。

pokemon <- dbReadTable(con, "pokemon")#将数据库中的表读入R并存储  
dbWriteTable(con, "test", pokemon)#直接将一张表写入数据库  
dbExistsTable(con, "test")#查看某表是否存在  
dbRemoveTable(con, "test")#删除这张表  
##查看数据类型  
dbDataType(con, pokemon$id)  
dbDataType(con, pokemon$base\_experience)

#### 用到SQL语言的查询

具体套路就一个，先把SQL语句写成字符串格式，然后用dbGetQuery()发送查询。

sql <- "SQL language"  
dbGetQuery(con,sql)

除了用户权限的设定DCL，在SQLite驱动下无法实现，其它所有的SQL语句都可以用这个思路实现。这里仅给出一些例子供大家复习，SQL的水太深了，更多细节大伙儿自个探索哈！

我们可以用SQL创建新的表、在表中插入值，或对表中数据进行修改：

#创建表   
sql <- "CREATE TABLE mytest (  
id INTEGER NOT NULL ,  
identifier VARCHAR(100) NOT NULL ,  
generation\_id INTEGER NOT NULL ,  
evolution\_chain\_id INTEGER NOT NULL ,  
PRIMARY KEY (id)  
) ;"  
dbGetQuery(con, sql)

## data frame with 0 columns and 0 rows

dbReadTable(con,"mytest")

## [1] id identifier generation\_id   
## [4] evolution\_chain\_id  
## <0 rows> (or 0-length row.names)

#往表中填数据  
sql <- "INSERT INTO mytest (id, identifier,  
 generation\_id, evolution\_chain\_id)  
 VALUES (1, 'bulbasaur', 1, 1),  
 (2, 'ivysaur', 1, 1),  
 (3, 'venusaur', 1, 1),  
 (4, 'charmander', 1, 2),  
 (5, 'charmenleon', 1, 2);"  
dbGetQuery(con, sql)

## data frame with 0 columns and 0 rows

dbReadTable(con,"mytest")

## id identifier generation\_id evolution\_chain\_id  
## 1 1 bulbasaur 1 1  
## 2 2 ivysaur 1 1  
## 3 3 venusaur 1 1  
## 4 4 charmander 1 2  
## 5 5 charmenleon 1 2

#对表进行修改  
sql <- "ALTER TABLE mytest ADD COLUMN color\_id INT ; "  
dbGetQuery(con, sql)

## data frame with 0 columns and 0 rows

dbReadTable(con,"mytest")

## id identifier generation\_id evolution\_chain\_id color\_id  
## 1 1 bulbasaur 1 1 NA  
## 2 2 ivysaur 1 1 NA  
## 3 3 venusaur 1 1 NA  
## 4 4 charmander 1 2 NA  
## 5 5 charmenleon 1 2 NA

#删除某行数据  
sql <- "DELETE FROM mytest WHERE identifier = 'bulbasaur' ;"  
dbGetQuery(con, sql)

## data frame with 0 columns and 0 rows

dbReadTable(con, "mytest")

## id identifier generation\_id evolution\_chain\_id color\_id  
## 1 2 ivysaur 1 1 NA  
## 2 3 venusaur 1 1 NA  
## 3 4 charmander 1 2 NA  
## 4 5 charmenleon 1 2 NA

#删除表  
sql <- "DROP TABLE mytest"  
dbGetQuery(con, sql)#也可以用dbRemoveTable()方法

## data frame with 0 columns and 0 rows

最好用的就是各种检索，各种join

#检索体重超过4000的pokemon  
sql <- "SELECT \* FROM pokemon  
 WHERE weight >=4000 ;"  
dbGetQuery(con, sql)

## id species\_id height weight base\_experience order is\_default  
## 1 143 143 21 4600 154 174 1  
## 2 208 208 92 4000 196 104 1  
## 3 376 376 16 5500 210 408 1  
## 4 383 383 35 9500 218 415 1  
## 5 483 483 54 6830 220 496 1  
## 6 485 485 17 4300 215 498 1  
## 7 486 486 37 4200 220 499 1  
## 8 487 487 45 7500 220 500 1  
## 9 656 487 69 6500 220 501 0

#检索名字中带有Nido的pokemon  
sql <- "SELECT identifier FROM pokemon\_species  
 WHERE identifier LIKE '%Nido%' ;"  
dbGetQuery(con, sql)

## identifier  
## 1 nidoran-f  
## 2 nidorina  
## 3 nidoqueen  
## 4 nidoran-m  
## 5 nidorino  
## 6 nidoking  
## 7 cranidos

#检索体重超过4000的pokemon（联合表查询）  
sql <- "SELECT pokemon.id, identifier, height, weight FROM pokemon  
 INNER JOIN pokemon\_species   
 ON pokemon.species\_id = pokemon\_species.id  
 WHERE weight >= 4000 ;"  
dbGetQuery(con, sql)

## id identifier height weight  
## 1 143 snorlax 21 4600  
## 2 208 steelix 92 4000  
## 3 376 metagross 16 5500  
## 4 383 groudon 35 9500  
## 5 483 dialga 54 6830  
## 6 485 heatran 17 4300  
## 7 486 regigigas 37 4200  
## 8 487 giratina 45 7500  
## 9 656 giratina 69 6500

#按种类计算pokemon的平均身高、体重并列出身高超过50的pokemon  
sql <- "SELECT identifier, AVG(height), AVG(weight) FROM pokemon  
 INNER JOIN pokemon\_species  
 ON pokemon.species\_id = pokemon\_species.id  
 GROUP BY pokemon.species\_id  
 HAVING AVG(height) >= 50 ;"  
dbGetQuery(con,sql)

## identifier AVG(height) AVG(weight)  
## 1 onix 88 2100  
## 2 gyarados 65 2350  
## 3 steelix 92 4000  
## 4 lugia 52 2160  
## 5 wailord 145 3980  
## 6 milotic 62 1620  
## 7 rayquaza 70 2065  
## 8 dialga 54 6830  
## 9 giratina 57 7000

dbGetQuery()还有一个孪生兄弟叫dbSendQuery(),但是这个兄弟比较懒，dbGetQuery()会将返回值整理成data.frame输出，dbSendQuery()则不会输出，本次都要它老婆fetch()来敲打它一下下才会把结果吐出来。

test <- dbSendQuery(con, sql)  
fetch(test)

最后，一定要记得关闭数据库哦！

dbDisconnect(con)

## [1] TRUE

### 补充

好啦，这个萌到爆炸的数据库送给你们~ <https://github.com/kikin81/pokemon-sqlite/blob/master/pokedex.sqlite>