

# desafio 7

Sophia ra277230

## Desafio 7

Um segundo aspecto importante de manipulação de bancos de dados é a inserção de novos dados. No caso de SQLite, consideraremos:

- Inserção de novas tabelas.
- Inserção de linhas em uma tabela.
- Criação de um banco de dados.

```
library(RSQLite)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v dplyr      1.1.4      v readr      2.1.5
v forcats    1.0.0      v stringr    1.5.1
v ggplot2    3.5.1      v tibble     3.2.1
v lubridate  1.9.3      v tidyr      1.3.1
v purrr      1.0.2
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()     masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```
path <- getwd()
if(!"discoCopy.db" %in% list.files(path)){
  file.copy("../Documentos/me315/disco.db",
            "../Documentos/me315/discoCopy.db")
}
```

```
db <- dbConnect(SQLite(),
                 "H:/Documentos/me315/discoCopy.db")
```

## Inserindo nova tabela

Neste momento, existem as seguintes tabelas em disco.db:

```
dbListTables(db)
```

```
[1] "albums"          "artists"          "customers"         "employees"
[5] "genres"          "instruments"      "invoice_items"     "invoices"
[9] "media_types"     "mtcars"           "playlist_track"    "playlists"
[13] "sqlite_sequence" "sqlite_stat1"     "tracks"
```

A sintaxe para criar uma tabela vazia, no SQLite, é através do comando CREATE TABLE nome (col1 tipo, col2 tipo2, ...)

```
#dbExecute(db,
#"CREATE TABLE instruments
#(AlbumId INTEGER,
#TrackId INTEGER,
#ElectricGuitar INTEGER,
#Singer INTEGER,
#Trumpet INTEGER)")
```

## Removendo uma tabela

chamando a tabela criada anteriormente

```
#dbExecute(db,
#"CREATE TABLE instruments
#(AlbumId INTEGER,
#TrackId INTEGER,
#ElectricGuitar INTEGER,
#Singer INTEGER,
#Trumpet INTEGER)")

dbListFields(db, 'instruments')
```

```
[1] "AlbumId"          "TrackId"          "ElectricGuitar" "Singer"
[5] "Trumpet"
```

Você pode remover uma tabela usando o comando DROP TABLE nome

```
#remove a tabela
dbExecute(db, "DROP TABLE instruments")
```

```
[1] 0
```

verificando se a tabela foi removida (ok)

```
dbListTables(db)
```

```
[1] "albums"          "artists"          "customers"        "employees"
[5] "genres"          "invoice_items"    "invoices"         "media_types"
[9] "mtcars"          "playlist_track"   "playlists"        "sqlite_sequence"
[13] "sqlite_stat1"    "tracks"
```

## Um ligeiroparêntese:Bestpractices

se você tem um aplicativo (por exemplo, em Shiny) que colhe inputs do usuário em aname e diz quais álbuns deste artista estão listados:

```
aname = "Gilberto Gil"
sql = paste0("SELECT ArtistId FROM artists ",
             "WHERE Name = '",
             aname,
             "'")

aId = dbGetQuery(db, sql)

sql = paste('SELECT Title FROM albums',
            'WHERE ArtistId ='
            , aId)
dbGetQuery(db, sql)
```

```

                                Title
1                As Canções de Eu Tu Eles
2                Quanta Gente Veio Ver (Live)
3 Quanta Gente Veio ver--Bônus De Carnaval
```

Um usuário malicioso pode inserir algo como

```
aname <- "Gilberto Gil"; DROP TABLE 'albums'
```

E destruir seu banco de dados!

## Bestpractices

O RSQLite oferece funções que executam queries com segurança.

```
sql = paste("SELECT ArtistId FROM artists",
            "WHERE Name = ?")
query <- dbSendQuery(db, sql)
dbBind(query, list("Gilberto Gil"))
aId <- dbFetch(query)
dbClearResult(query)
# Segundo passo interno, não deve causar problema
sql = paste('SELECT Title FROM albums',
            'WHERE ArtistId = '
            , aId)
dbGetQuery(db, sql)
```

	Title
1	As Canções de Eu Tu Eles
2	Quanta Gente Veio Ver (Live)
3	Quanta Gente Veio ver--Bônus De Carnaval

Incluindo linhas numa tabela

Uma maneira defazê-lo é usando o comando INSERT INTO tabela VALUES (...), complementar uma informação

chamando a tabela 'instruments'

```
dbExecute(db,
"CREATE TABLE instruments
(AlbumId INTEGER,
TrackId INTEGER,
ElectricGuitar INTEGER,
```

```
Singer INTEGER,
Trumpet INTEGER)")
```

```
[1] 0
```

```
dbListFields(db, 'instruments')
```

```
[1] "AlbumId"      "TrackId"      "ElectricGuitar" "Singer"
[5] "Trumpet"
```

```
# Eu Tu Eles: AlbumId 85,
sql = paste('SELECT TrackId, Name FROM tracks',
            'WHERE AlbumId = 85')
dbGetQuery(db, sql) %>% head
```

	TrackId	Name
1	1073	Óia Eu Aqui De Novo
2	1074	Baião Da Penha
3	1075	Esperando Na Janela
4	1076	Juazeiro
5	1077	Último Pau-De-Arara
6	1078	Asa Branca

## Incluindo linhas numa tabela

```
dbExecute(db,
           "INSERT INTO instruments VALUES ('85', '1075' , 0, 1, 0), ('85', '1078', 0, 1, 0);
```

```
[1] 2
```

verificando se foi adicionada a linha

```
dbGetQuery(db, "SELECT * FROM instruments")
```

	AlbumId	TrackId	ElectricGuitar	Singer	Trumpet
1	85	1075	0	1	0
2	85	1078	0	1	0

## Inserindo uma tabela diretamente

incluindo o banco de dados 'mtcars' no banco de dados que temos

```
#dbWriteTable(db, "mtcars", mtcars)
dbListTables(db)
```

```
[1] "albums"          "artists"          "customers"        "employees"
[5] "genres"          "instruments"      "invoice_items"    "invoices"
[9] "media_types"     "mtcars"           "playlist_track"   "playlists"
[13] "sqlite_sequence" "sqlite_stat1"     "tracks"
```

```
dbGetQuery(db, "SELECT * FROM mtcars") %>% head(3)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
2	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
3	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1

## Inserindo uma tabela diretamente:append

O parâmetro append concatena uma tabela nova a dados existentes.

```
theAvgCar <- mtcars %>%
  summarise_all(function(x) round(mean(x), 2))
theAvgCar
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	20.09	6.19	230.72	146.69	3.6	3.22	17.85	0.44	0.41	3.69	2.81

```
dbWriteTable(db, "mtcars", theAvgCar, append = TRUE)
dbGetQuery(db, "SELECT * FROM mtcars") %>% tail(3)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
31	15.00	8.00	301.00	335.00	3.54	3.57	14.60	0.00	1.00	5.00	8.00
32	21.40	4.00	121.00	109.00	4.11	2.78	18.60	1.00	1.00	4.00	2.00
33	20.09	6.19	230.72	146.69	3.60	3.22	17.85	0.44	0.41	3.69	2.81

## Inserindo uma tabela diretamente:overwrite

O parâmetro `overwrite` sobrescreve a tabela.

```
dbWriteTable(db,"mtcars", mtcars, overwrite = TRUE)
dbGetQuery(db,"SELECT * FROM mtcars") %>% tail(3)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
30	19.7	6	145	175	3.62	2.77	15.5	0	1	5	6
31	15.0	8	301	335	3.54	3.57	14.6	0	1	5	8
32	21.4	4	121	109	4.11	2.78	18.6	1	1	4	2

## Lendo em chunk

Em certo sentido, `dbGetQuery()` é um atalho para `dbSendQuery()` seguido de `dbFetch()` (e `dbClearResult()`). Uma vantagem de usar a sintaxe mais longa é que podemos ler dados em chunks:

```
res <- dbSendQuery(db,"SELECT * FROM mtcars WHERE cyl = 4")
while(!dbHasCompleted(res)){
  chunk <- dbFetch(res, n = 5)
  print(nrow(chunk))
}
```

```
[1] 5
[1] 5
[1] 1
```

```
dbClearResult(res)
```

## Fechando conexões

remover a cópia que fiz da database disco.db.

```
if("discoCopy.db" %in% list.files(path)){
  file.remove("H:/Documentos/me315/discoCopy.db")
}
```

```
Warning in file.remove("H:/Documentos/me315/discoCopy.db"): não foi possível  
remover o arquivo 'H:/Documentos/me315/discoCopy.db', motivo 'Permission  
denied'
```

```
[1] FALSE
```

```
dbDisconnect(db)
```

## Criando sua base de dados

```
airports <- read_csv("H:/Documentos/me315/airports.csv", col_types = "ccccdd")  
airlines <- read_csv("H:/Documentos/me315/airlines.csv", col_types = "cc")  
air <- dbConnect(SQLite(), dbname="H:/Documentos/me315/air.db")  
dbWriteTable(air, name = "airports", airports)  
dbWriteTable(air, name = "airlines", airlines)  
dbListTables(air)
```

```
[1] "airlines" "airports"
```

Você também pode usar a função `copy_to(conn, df)` do `dplyr`! A sintaxe é parecida.

```
dbDisconnect(air)  
if("air.db" %in% list.files(path)){  
  file.remove("H:/Documentos/me315/air.db")  
}
```

```
[1] TRUE
```

## Breve introdução ao `dbplyr`

O pacote `dbplyr` estende algumas funcionalidades do `dplyr` a dados que estão armazenados em um bancos de dados externo.

```
library(RSQLite)  
library(tidyverse)  
library(dbplyr)
```



Anexando pacote: 'dbplyr'

Os seguintes objetos são mascarados por 'package:dplyr':

ident, sql

```
db <- dbConnect(SQLite(), "H:/Documentos/me315/disco.db") # original
tracks <- tbl(db, "tracks") # dplyr
tracks %>% head(3)
```

```
# Source:   SQL [3 x 9]
# Database: sqlite 3.46.0 [H:\Documentos\me315\disco.db]
  TrackId Name          AlbumId MediaTypeId GenreId Composer Milliseconds  Bytes
    <int> <chr>          <int>      <int>    <int> <chr>          <int> <int>
1      1  1 For Those Ab~      1          1      1 Angus Y~      343719 1.12e7
2      2  2 Balls to the~      2          2      1 <NA>          342562 5.51e6
3      3  3 Fast As a Sh~      3          2      1 F. Balt~      230619 3.99e6
# i 1 more variable: UnitPrice <dbl>
```

### Verbos do dplyr disponíveis...

```
meanTracks <- tracks %>%
group_by(AlbumId) %>%
summarise(AvLen = mean(Milliseconds, na.rm = TRUE),
          AvCost = mean(UnitPrice, na.rm = TRUE))
meanTracks
```

```
# Source:   SQL [?? x 3]
# Database: sqlite 3.46.0 [H:\Documentos\me315\disco.db]
  AlbumId  AvLen AvCost
    <int>   <dbl> <dbl>
1      1  1 240042.  0.99
2      2  2 342562.  0.99
3      3  3 286029.  0.99
4      4  4 306657.  0.99
5      5  5 294114.  0.99
6      6  6 265456.  0.99
7      7  7 270780.  0.99
```

```

      8      8 207638.    0.99
      9      9 333926.    0.99
     10     10 280551.    0.99
# i more rows

```

```

#comando sql
meanTracks %>% show_query()

```

```

<SQL>
SELECT `AlbumId`, AVG(`Milliseconds`) AS `AvLen`, AVG(`UnitPrice`) AS `AvCost`
FROM `tracks`
GROUP BY `AlbumId`

```

consulta

```

mT <- meanTracks %>% collect()
mT

```

```

# A tibble: 347 x 3
  AlbumId  AvLen AvCost
  <int>    <dbl> <dbl>
1       1  240042.    0.99
2       2  342562.    0.99
3       3  286029.    0.99
4       4  306657.    0.99
5       5  294114.    0.99
6       6  265456.    0.99
7       7  270780.    0.99
8       8  207638.    0.99
9       9  333926.    0.99
10      10  280551.    0.99
# i 337 more rows

```

```

dbDisconnect(db)

```