Acceder a bases SQL usando Python o R

Guillermo Bustos-Pérez

12/3/2020

Introducción

SQL es uno de los sistemas de bases de gestión de bases de datos relacionales más usados del mundo. Aunque presenta ventajas como su gran capacidad de almacenamiento, manejo desde diferents plataformas o habilidad para ser estructurar el almacenamiento de datos, posee un potencial limitado a la hora de aplicar procesos de análisis estadístico o construcción de modelos predictivos de machine learning o deep learning. El presente código procede del *Python programmer track* disponible en Datacamp Inc. Este curso supone más de 50h lectivas en las que se enseñan los procedimientos de importado y limpieza de datos, manejo de dataframes, web scraping, etc.

Acceder a una base de datos SQLite usando Python y pandas

En este código se emplea un database engine y un context manager para hacer los querys deseados a la base de datos. A continuación se importan los datos a un data frame de Pandas que permite manejar mejor su análisis. Lo primero es **importar** los paquetes **Pandas** y **SQLAlchemy**. Existen varios paquetes para acceder a bases de datos relacionales como sqlite3, pero SQLAlchemy permite funciiona también con otras bases de datos relacionales commo Postgres y MySQL.

La base de datos empleada es la *Indian Premier League SQLite Database* disponible en kaggle (https://www.kaggle.com/harsha547/ipldatabase). Esta es una base de datos relacional que contiene 21 tablas de la liga de críquet de la India y que suele ser comunmente empleada para practicar querries con SQL.

```
# Import needed packages
# Importar los paquetes necesarios
import pandas as pd

from sqlalchemy import create_engine
```

El siguiente paso es usar la función **create engine** para establecer el acceso al archivo sqlite y tras esto podemos obtener el nombre de las tablas de la base de datos relacional.

```
# Conect engine to sqlite database
# Conectar el engine a la base de datos sqlite
engine = create_engine('sqlite:///Data/database.sqlite'
```

```
# Get table names using engine
# Obtener el nombre de las tablas usando el engine
table_names = engine.table_names()
print(table_names)
```

```
## ['Ball_by_Ball', 'Batsman_Scored', 'Batting_Style', 'Bowling_Style', 'City', 'Country', 'Extra_Runs'
```

A partir de aquí tenemos dos opciones:

1) Importar las bases de datos que queramos como dataframes de Pandas y realizar el inner join empleando python

2) Hacer directamente un querring empleando el context manager y guardar directamente el resultado como un data frame

En ambos casos se empleará unn context manager. Esto se debe a que los context managers son más eficientes al usar menos líneas de código y no ser necesario especificar el cierre a la conexión del archivo sqlite.

Opción 1: importar data frames independientes y realizar un inner join usando Python

Para este ejemplo vamos a hacer un importado e inner join sencillos. Las tablas sobre las que realizar un inner join son la de *players* y *countires*. En ambos casos se importan ambas tablas como data frames usando un context manager.

```
# Import players table from .sqlite and store as a dataframe
# Immportar la tabla players del archivo sqlite como un dataframe de pandas
with engine.connect() as con :
  rs = con.execute("SELECT * FROM Player")
  df_players = pd.DataFrame(rs.fetchall())
  df_players.columns = rs.keys()

print(df_players.head())
```

```
##    Player_Id      Player_Name  ... Bowling_skill  Country_Name
## 0          1       SC Ganguly  ...           1.0             1
## 1          2      BB McCullum  ...           1.0             4
## 2          3       RT Ponting  ...           1.0             5
## 3          4        DJ Hussey  ...           2.0             5
## 4          5  Mohammad Hafeez  ...           2.0             6
##
## [5 rows x 6 columns]
```

```
# Import countries table from .sqlite and store as a dataframe
# Immportar la tabla countries del archivo sqlite como un dataframe de pandas
with engine.connect() as con :
  rs = con.execute("SELECT * FROM Country")
  df_country = pd.DataFrame(rs.fetchall())
  df_country.columns = rs.keys()

print(df_country.head())
```

```
##    Country_Id  Country_Name
## 0           1         India
## 1           2  South Africa
## 2           3         U.A.E
## 3           4   New Zealand
## 4           5     Australia
```

Una vez disponemos de ambos dataframes podemos realizar el inner join indicando que las columnas de los data frames tienen nombres diferentes.

```
players_countries = pd.merge(df_players, df_country,
                             left_on = 'Country_Name', right_on = 'Country_Id')
print(players_countries.head())
```

```
##    Player_Id Player_Name  ... Country_Id  Country_Name_y
## 0          1  SC Ganguly  ...          1           India
## 1          6    R Dravid  ...          1           India
## 2          7    W Jaffer  ...          1           India
## 3          8     V Kohli  ...          1           India
## 4         12     B Akhil  ...          1           India
##
## [5 rows x 8 columns]
```

```
print(players_countries.tail())
```

```
##      Player_Id        Player_Name  ... Country_Id  Country_Name_y
## 464        195   Mashrafe Mortaza  ...         11      Bangladesh
## 465        202  Mohammad Ashraful  ...         11      Bangladesh
## 466        276    Shakib Al Hasan  ...         11      Bangladesh
## 467        460  Mustafizur Rahman  ...         11      Bangladesh
## 468        284   RN ten Doeschate  ...         12     Netherlands
##
## [5 rows x 8 columns]
```

Opción 2: hacer un querring directamente y guardar como data frame de Pandas

Esta opción requiere de menos líneas de código, pero requiere del conocimiento de la sintaxis de SQL. En este caso el querry se realiza directamente usando el engine y se guarda como un dataframe.

```
df_InJoin = pd.read_sql_query(
  "SELECT * FROM Player INNER JOIN Country on Player.Country_Name = Country.Country_Id",
  engine)

print(df_InJoin.head())
```

```
##    Player_Id      Player_Name  ... Country_Id  Country_Name
## 0          1       SC Ganguly  ...          1         India
## 1          2      BB McCullum  ...          4   New Zealand
## 2          3       RT Ponting  ...          5     Australia
## 3          4        DJ Hussey  ...          5     Australia
## 4          5  Mohammad Hafeez  ...          6      Pakistan
##
## [5 rows x 8 columns]
```

Acceder a una base de datos SQLite usando R

Acceder a archivos SQLite usando R y la IDE RStudio es muy sencillo también. Basta con usar los paquetes **RSQLite** y **tidyverse** (este último constituye un conjunto de paquetes para lectura, gestión y análisis de datos).

```r
# Load required packages
# Cargar los paquetes necesarios
library(RSQLite)
library(tidyverse)
```

En este caso el procedimiento vuelve a ser similar que con Python. Se emplea la función **dbConnect()** del paquete **RSQLite** para establecer una conexión con el archivo .sqlite.

```r
# Connect to .sqlite and get table names
# Conectarse a .sqlite y obtener los nombres de las tablas
con <- dbConnect(SQLite(), "Data/database.sqlite")
as.data.frame(dbListTables(con))
```

Una vez establecida la conexión volvemos a tener las dos opciones previamente observadas:

1) Importar los datos como data frames y realizar el inner join usando R

2) Hacer directamente un querring empleando el engine y guardar directamente el resultado como un data frame

Opción 1. Dataframes e inner joins en R

Obtenemos nuevamente las tablas de Players y de Countries. Podemos hacer el querry y guardarlo como un data frame o leer directamente la tabla y guardarla como un data frame. Ambas líneas de código generan el mismo resultado.

```r
# Get table using a querry
players <- dbGetQuery(con, "SELECT * FROM Player")

# Read inn table
players <- dbReadTable(con, 'Player')
head(players, 3)
```

```
##   Player_Id Player_Name                 DOB Batting_hand Bowling_skill
## 1         1  SC Ganguly 1972-07-08 00:00:00            1             1
## 2         2 BB McCullum 1981-09-27 00:00:00            2             1
## 3         3  RT Ponting 1974-12-19 00:00:00            2             1
##   Country_Name
## 1            1
## 2            4
## 3            5
```

```r
# Get Country table
country <- dbReadTable(con, 'Country')
```

Una vez tenemos ambos data frames podemos establecer el inner join empleando la función **merge()** y especificando los nombres de las columans ya que tienen nombres diferentes en los data frames.

```
Player_Countries <- merge(players, country, by.x = "Country_Name", by.y = "Country_Id",
      all = T)

head(Player_Countries %>% select(Player_Id, Player_Name, Country_Name.y), 3)
```

```
##   Player_Id Player_Name Country_Name.y
## 1         1  SC Ganguly          India
## 2       123    DT Patil          India
## 3       124    A Kumble          India
```

Opción 2. Hacer un querring y guardarlo como un data frame

Al igual que en python se puede realizar el querring directamente y guardarlo como un data frame. Esto ahorra líneas de código, pero es necesario aplicar la sintaxis de SQL.

```
# Directly make the inner join
# Hacer directamente el inner join
Inner_J <- dbGetQuery(con,"SELECT * FROM Player INNER JOIN Country on
                      Player.Country_Name = Country.Country_Id")
head(Inner_J)
```

```
##   Player_Id     Player_Name                 DOB Batting_hand Bowling_skill
## 1         1      SC Ganguly 1972-07-08 00:00:00            1             1
## 2         2     BB McCullum 1981-09-27 00:00:00            2             1
## 3         3      RT Ponting 1974-12-19 00:00:00            2             1
## 4         4       DJ Hussey 1977-07-15 00:00:00            2             2
## 5         5 Mohammad Hafeez 1980-10-17 00:00:00            2             2
## 6         6        R Dravid 1973-01-11 00:00:00            2             2
##   Country_Name Country_Id Country_Name
## 1            1          1        India
## 2            4          4  New Zealand
## 3            5          5    Australia
## 4            5          5    Australia
## 5            6          6     Pakistan
## 6            1          1        India
```

Importante

Una vez terminadas las operaciones en R, debemos terminar la conexión a la base de datos. Esta es una medida de seguridad muy importante.

```
dbDisconnect(con)
```