Universidade Federal do Rio Grande do Sul
Instituto de Informática

INF01145 – Fundamentos de Bancos de Dados
Prof. Karin Becker

Astélio Weber
Rodrigo N. Wuerdig

DataFlix - Programa para Manipular a Base de Dados

1 Desenvolvimento

Para desenvolver a o programa de interface e manipulação do banco de dados desenvolvido, foi escolhida a linguagem Python versão 3.7. Foi escolhida a linguagem Python devido a sua simplicidade, portabilidade e eficiência quando se diz respeito de tempo de projeto. Foi utilizado a biblioteca 'psycopg2', que é a interface PostgreSQL para Python mais popular.

1.1 Conexão com a Base de Dados

```
1     connection = psycopg2.connect(user = "postgres",
2                                   password = "postgres",
3                                   host = "127.0.0.1",
4                                   port = "5432",
5                                   database = "postgres")
6     cursor = connection.cursor()
```

Para conectar com a base de dados PostgreSQL utilizando a biblioteca 'psycopg2', utiliza-se a função *connect*(), passando os parâmetros definidos na criação da base de dados (linhas 1-5 no código acima). Parâmetros como: usuario (*user*), senha (*password*), IP do servidor (*host*), porta de acesso (*port*) e nome da base de dados (*database*).

Apos utilizar o comando *connect*(), o mesmo vai retornar um objeto (que no exemplo chamamos de 'connection'), utilizaremos o objeto junto com o seu método *cursor*() que retorna um objeto cursor para efetivamente realizarmos consultas e manipulações na nossa base de dados.

1.2 Encerrando uma Conexão com a Base de Dados

```
1         cursor.close()
2         connection.close()
```

O processo para encerrar a comunicação com a base de dados é muito simples e intuitiva. Para encerrar a conexão,primeiro chama-se o metodo *close*() do objeto cursor para após chamar o metodo *close*() do objeto connection definido na etapa de conexão (Sec. 1.1).

1.3 Fazendo Consultas

Para realizar consultas se usa o comando *execute*() do objeto cursor, onde um parâmetro contendo o comando SQL deve ser colocado. No exemplo utilizou-se de uma consulta que retorna a versão do PostgreSQL que está rodando no servidor (que aqui retorna '*PostgreSQL 11.3, compiled by Visual C++ build 1914, 64-bit*').

```
    cursor.execute("SELECT version();")
    record = cursor.fetchone()
    print("You are connected to - ", record,"\n")
```

Para capturar os dados da consulta, utiliza-se um dos comandos mostrados na tabela 1. No exemplo foi utilizado o comando *fetchone*() que retorna apenas uma tupla, que é armazenada na variável 'record'.

Tabela 1: Comandos para Obter Dados do Cursor

Função	**Objetivo**
cursor.fetchall()	Retorna Todas as Tuplas.
cursor.fetchone()	Retorna Apenas uma Tupla.
cursor.fetchmany(SIZE)	Retorna um Número Limitado de Tuplas

1.4 Fazendo Consultas no Contexto do DataFlix

Para fazer as consultas relacionadas ao DataFlix não segredo, basta seguir o modelo ER para fazer as corretas modelagens e rodar as mesmas no comando *execute*() explicado nas Seção 1.3. Dentro do programa desenvolvido para manipular o BD, criou-se funções que chamam cada comando SQL. Como por ex.:

```
def tempo_assistido():
  name = input("Tempo total assistido por?");
  postgreSQL_select_Query = "select sum(TEMPO_ASSISTIDO) from assiste where NOME_PERFIL='" + name +
   ↪  "'"
  cursor.execute(postgreSQL_select_Query)
  tempo = cursor.fetchone()
  print("O perfil" + name + ", assistiu DataFlix por" + str(tempo))
  input("Press [Enter] to continue...")
  menu()
```

Onde um nome de um perfil(Variável 'name', no BD chamado de NOME_PERFIL) é dado como entrada utilizando o comando *input*() do Python. Após o preenchimento da variável, em tempo de execução, é feita uma consulta que contabiliza o total de horas utilizadas por um determinado perfil. Como por ex. a consulta SQL:

```
select sum(TEMPO_ASSISTIDO) from assiste where NOME_PERFIL='joao'
```

que retorna uma tupla contendo o valor '00:10:20', que é a contabilização de todo tempo que o perfil 'joao' passou assistindo filmes e series.

A consulta é contabilizada no objeto cursor que após a execução do comando SQL é extraido a tupla que é armazenada na variável chamada de 'tempo' (linha 5 do código). Como a variável é do tipo tupla (*Tuple*) foi feito um *casting* para imprimir o valor (linha 6).

2 Consultas Desenvolvidas

Utilizando a mesma ideia da seção 1.3, criou-se funções para chamar todas as funções desenvolvidas.

- **Consulta o Titulo de Todas as Mídias Disponíveis**

```
def consulta_midias():
  postgreSQL_select_Query = "select distinct NOME_MIDIA from midia"
  cursor.execute(postgreSQL_select_Query)
  print("Selecting rows from midia table using cursor.fetchall")
  filmes = cursor.fetchall()
  for row in filmes:
    print(row)
  input("Press [Enter] to continue...")
  menu()
```

- **Consulta a Conta que tem a Fatura Mais Cara e Retorna seu *Nickname***

```
def conta_maior_fatura():
  postgreSQL_select_Query = "select NICKNAME from conta natural join cobranca natural join
   ↪  fatura group by VALOR, NICKNAME having VALOR = (select max(VALOR)from fatura);"
  cursor.execute(postgreSQL_select_Query)
  print("Selecionando Tupla contendo o NICKNAME que tem a maior Fatura")
  user = cursor.fetchone()
  print("A conta" + str(user[0]) + "tem a maior fatura!")
  input("Press [Enter] to continue...")
  menu()
```

- **Consulta que Retorna o Nome do Diretor da Midia mais Antiga**

```
def diretor_filme_mais_antigo():
  postgreSQL_select_Query = "select distinct NOMED from diretor where IDd in (select distinct
   ↪  IDd from dirige natural join midia group by ANO, IDd having ANO = (select min(ANO) from
   ↪  midia));"
  cursor.execute(postgreSQL_select_Query)
  diretores = cursor.fetchall()
  for diretor in diretores:
          print("O diretor" + str(diretor[0]) + ", dirigiu o filme mais antigo registrado.")
  input("Press [Enter] to continue...")
  menu()
```

- **Consulta que Retorna Contas que não tem Perfis KID**

```
def contas_adult_only():
  postgreSQL_select_Query = "select distinct NICKNAME from conta c1 Where not exists (select *
   ↪  From possui natural join perfil         Where EMAIL = c1.EMAIL and NOME_PERFIL IN (select
   ↪  distinct NOME_PERFIL From possui natural join perfil        Where KIDS = '1'));"
  cursor.execute(postgreSQL_select_Query)
  adults_only = cursor.fetchall()
  for adult in adults_only:
    print("O conta" + str(adult[0]) + ", não é utilizada por crianças.")
  input("Press [Enter] to continue...")
  menu()
```

- **Retorna E-mail e Senha de Usuários com Senha Fraca (8¿caracteres)**

```
def senha_fraca():
  postgreSQL_select_Query = "select EMAIL, SENHA_DIGEST from conta group by EMAIL having 8 >
   ↪  CHAR_LENGTH(SENHA_DIGEST);"
  cursor.execute(postgreSQL_select_Query)
```

```
    noobies = cursor.fetchall()
    for noob in noobies:
      print("A conta de EMAIL:" + str(noob[0]) + "\nTem uma senha muito fraca! Senha:" +
      ↪  str(noob[1]))
    input("Press [Enter] to continue...")
    menu()
```

- **Consulta que Retorna o Titulo Mais Favoritado no DataFlix**

```
def titulo_maisfav():
  postgreSQL_select_Query = "select max(distinct TITULO) from favoritos;"
  cursor.execute(postgreSQL_select_Query)
  user = cursor.fetchone()
  print("A midia mais favoritada é:" + str(user[0]))
  input("Press [Enter] to continue...")
  menu()
```

- **Consulta que Retorna o Número de Temporadas de Cada Série**

```
def temp_series():
  postgreSQL_select_Query = "select distinct(NOME_MIDIA), count(IDt) from series natural join
  ↪  contem group by NOME_MIDIA;"
  cursor.execute(postgreSQL_select_Query)
  series = cursor.fetchall()
  for serie in series:
    print("A série:" + str(serie[0]) + " tem " + str(serie[1])+" temporadas")
  input("Press [Enter] to continue...")
  menu()
```

- **Consulta o Tempo Total de uma Conta, [Entrada em Tempo de Execução], Assistindo.**

```
def tempo_assistido():
  name = input("Tempo total assistido por?");
  postgreSQL_select_Query = "select sum(TEMPO_ASSISTIDO) from assiste where NOME_PERFIL='" +
  ↪  name + "'"
  cursor.execute(postgreSQL_select_Query)
  tempo = cursor.fetchone()
  print("O perfil " + name + ", assistiu DataFlix por " + str(tempo[0])) + " (hh:mm:ss)"
  input("Press [Enter] to continue...")
  menu()
```

- **Consulta o Valor das Faturas de uma Determinada Conta [Entrada em Tempo de Execução]**

```
def fatura():
  email = input("Entre o Email ao qual se quer descobrir o valor das faturas: ");
  postgreSQL_select_Query = "select VALOR from fatura natural join cobranca where EMAIL='" +
  ↪  email + "'"
  cursor.execute(postgreSQL_select_Query)
  faturas = cursor.fetchall()
  print("Faturas do EMAIL:"+ email)
  print("----------------------------------------------------")
  for fatura0 in faturas:
          print(fatura0[0])
  input("Press [Enter] to continue...")
  menu()
```

3 Procedure

Foi desenvolvida uma função que chama a *procedure* criada. A *procedure* $aumenta\_fatura(din\ money)$ recebe um valor monetario e distribui por todas as faturas. Este procedimento foi desenvolvido com o intuito de compensar o valor de uma fatura deletada nas outras faturas.

$$ValorUnitario = Parâmetro/N_{Faturas}$$

Há uma subconsulta dentro da procedure que atribui a uma variável ($N_{Faturas}$) o numero de faturas.

```
1  def imprime_todas_faturas():
2    postgreSQL_select_Query = "select VALOR from fatura"
3    cursor.execute(postgreSQL_select_Query)
4    filmes = cursor.fetchall()
5    for row in filmes:
6      print(row)
7
8  def distribui_valor_nas_faturas():
9    print("-------------------------------------------------------")
10   print("Faturas:")
11   imprime_todas_faturas();
12   print("-------------------------------------------------------")
13   Valor = input("Entre o Valor que Gostaria de Distribuir no Valor de Todas as Faturas (00.00):\n");
14   postgreSQL_select_Query = "CALL aumenta_fatura('" + Valor + "')"
15   cursor.execute(postgreSQL_select_Query)
16   print("-------------------------------------------------------")
17   print("Resultado:")
18   imprime_todas_faturas();
19   print("-------------------------------------------------------")
20   input("Press [Enter] to continue...")
21   menu()
```

Basicamente a função desenvolvida recebe um valor em tempo de execução (linha 13), e depois executa a chamada ($CALL$) da *procedure* (linhas 14-15).

4 Apêndice

4.1 instancias.sql

```
#!/usr/bin/env python2.7
import psycopg2
import os


def print_logo():
  print("---------------------------------------------------------------");
  print("Your Database for Films and Series");
  print("---------------------------------------------------------------");
  print("Made by: Astelio Weber and Rodrigo N. Wuerdig");
  print("---------------------------------------------------------------");

def consulta_filmes():
  postgreSQL_select_Query = "select distinct NOME_MIDIA from midia"
  cursor.execute(postgreSQL_select_Query)
  print("Selecting rows from midia table using cursor.fetchall")
  filmes = cursor.fetchall()
  for row in filmes:
    print(row)
  input("Press [Enter] to continue...")
  menu()

def conta_maior_fatura():
  postgreSQL_select_Query = "select NICKNAME from conta natural join cobranca natural join fatura
   ↪  group by VALOR, NICKNAME having VALOR = (select max(VALOR)from fatura);"
  cursor.execute(postgreSQL_select_Query)
  print("Selecionando Tupla contendo o NICKNAME que tem a maior Fatura")
  user = cursor.fetchone()
  print("A conta" + str(user[0]) + "tem a maior fatura!")
  input("Press [Enter] to continue...")
  menu()

def diretor_filme_mais_antigo():
  postgreSQL_select_Query = "select distinct NOMED from diretor where IDd in (select distinct IDd from
   ↪  dirige natural join midia group by ANO, IDd having ANO = (select min(ANO) from midia));"
  cursor.execute(postgreSQL_select_Query)
  diretores = cursor.fetchall()
  for diretor in diretores:
          print("O diretor" + str(diretor[0]) + ", dirigiu o filme mais antigo registrado.")
  input("Press [Enter] to continue...")
  menu()

def contas_adult_only():
  postgreSQL_select_Query = "select distinct NICKNAME from conta c1 Where not exists (select * From
   ↪  possui natural join perfil         Where EMAIL = c1.EMAIL and NOME_PERFIL IN (select distinct
   ↪  NOME_PERFIL From possui natural join perfil        Where KIDS = '1'));"
  cursor.execute(postgreSQL_select_Query)
  adults_only = cursor.fetchall()
  for adult in adults_only:
    print("O conta" + str(adult[0]) + ", não é utilizada por crianças.")
  input("Press [Enter] to continue...")
  menu()

def senha_fraca():
  postgreSQL_select_Query = "select EMAIL, SENHA_DIGEST from conta group by EMAIL having 8 >
   ↪  CHAR_LENGTH(SENHA_DIGEST);"
  cursor.execute(postgreSQL_select_Query)
  noobies = cursor.fetchall()
  for noob in noobies:
    print("A conta de EMAIL:" + str(noob[0]) + "\nTem uma senha muito fraca! Senha:" + str(noob[1]))
  input("Press [Enter] to continue...")
```

```
57    menu()
58
59  def titulo_maisfav():
60    postgreSQL_select_Query = "select max(distinct TITULO) from favoritos;"
61    cursor.execute(postgreSQL_select_Query)
62    user = cursor.fetchone()
63    print("A midia mais favoritada é:" + str(user[0]))
64    input("Press [Enter] to continue...")
65    menu()
66
67  def temp_series():
68    postgreSQL_select_Query = "select distinct(NOME_MIDIA), count(IDt) from series natural join contem
       ↪  group by NOME_MIDIA;"
69    cursor.execute(postgreSQL_select_Query)
70    series = cursor.fetchall()
71    for serie in series:
72      print("A série:" + str(serie[0]) + " tem " + str(serie[1])+" temporadas")
73    input("Press [Enter] to continue...")
74    menu()
75
76
77  def tempo_assistido():
78    name = input("Tempo total assistido por?");
79    postgreSQL_select_Query = "select sum(TEMPO_ASSISTIDO) from assiste where NOME_PERFIL='" + name +
       ↪  "'"
80    cursor.execute(postgreSQL_select_Query)
81    tempo = cursor.fetchone()
82    print("O perfil " + name + ", assistiu DataFlix por " + str(tempo[0])) + " (hh:mm:ss)"
83    input("Press [Enter] to continue...")
84    menu()
85
86
87  def fatura():
88    email = input("Entre o Email ao qual se quer descobrir o valor das faturas: ");
89    postgreSQL_select_Query = "select VALOR from fatura natural join cobranca where EMAIL='" + email +
       ↪  "'"
90    cursor.execute(postgreSQL_select_Query)
91    faturas = cursor.fetchall()
92    print("Faturas do EMAIL:"+ email)
93    print("-----------------------------------------------------")
94    for fatura0 in faturas:
95            print(fatura0[0])
96    input("Press [Enter] to continue...")
97    menu()
98
99  def imprime_todas_faturas():
100   postgreSQL_select_Query = "select VALOR from fatura"
101   cursor.execute(postgreSQL_select_Query)
102   filmes = cursor.fetchall()
103   for row in filmes:
104     print(row)
105
106 def distribui_valor_nas_faturas():
107   print("-----------------------------------------------------")
108   print("Faturas:")
109   imprime_todas_faturas();
110   print("-----------------------------------------------------")
111   Valor = input("Entre o Valor que Gostaria de Distribuir no Valor de Todas as Faturas (00.00):\n");
112   postgreSQL_select_Query = "CALL aumenta_fatura('" + Valor + "')"
113   cursor.execute(postgreSQL_select_Query)
114   print("-----------------------------------------------------")
115   print("Resultado:")
116   imprime_todas_faturas();
117   print("-----------------------------------------------------")
118   input("Press [Enter] to continue...")
119   menu()
```

```
120
121
122 menuItems = [
123     { "Tempo assistido por [entrada]": tempo_assistido },
124     { "Faturas do EMAIL [entrada]": fatura },
125     { "Dilui um Valor Monetario por Todas as Faturas [Procedure]": distribui_valor_nas_faturas },
126     { "Contas com Senha Fraca": senha_fraca },
127     { "Consultar Midias": consulta_filmes },
128     { "Conta que tem a maior fatura": conta_maior_fatura },
129     { "Diretor do Filme mais Antigo Registrado": diretor_filme_mais_antigo },
130     { "Contas que não são usadas por crianças": contas_adult_only },
131     { "Midia Mais Favoritada": titulo_maisfav },
132     { "Numero de Temporadas por Series": temp_series },
133     { "Exit": exit },
134 ]
135
136 def menu():
137   while True:
138     os.system('clear')
139     print_logo();
140     for item in menuItems:
141         print("["+str(menuItems.index(item))+"]" + list(item.keys())[0])
142     choice = input(">> ")
143     try:
144         if int(choice) < 0 : raise ValueError
145         list(menuItems[int(choice)].values())[0]()
146     except (ValueError, IndexError):
147         pass
148
149 try:
150     connection = psycopg2.connect(user = "postgres",
151                                   password = "postgres",
152                                   host = "127.0.0.1",
153                                   port = "5432",
154                                   database = "postgres")
155     cursor = connection.cursor()
156     print ( connection.get_dsn_parameters(),"\n")
157     cursor.execute("SELECT version();")
158     record = cursor.fetchone()
159     print("You are connected to - ", record,"\n")
160 except (Exception, psycopg2.Error) as error :
161     print ("Error while connecting to PostgreSQL", error)
162 finally:
163     #closing database connection.
164         if(connection):
165             menu();
166             cursor.close()
167             connection.close()
168             print("PostgreSQL connection is closed")
```