Nicholas Capriotti
CS317
12/10/2022

Exercise 5

This exercise is an introduction to connectors, which will allow us to interact with the database indirectly using other languages such as Python and PHP, the examples in this document are done in both Python and Sql, with various parts of the assignment requiring different implementations of the procedures. We will be working on the soccer database that we designed and filled in exercises 3A and 3B. By the end of the exercise, we'll be able to demonstrate the ability to use Procedures and Connectors to find and display data within the database.

We'll begin by creating a basic connector using Python. Theconnector is used to execute a query in our database, and display the result to the terminal. For the first example, to test the connector, the query will only return the names of the soccer teams from Exercise 3B. The Python code used to execute the query is shown below

```
import mysql.connector

try:
  mydb = mysql.connector.connect(
    host='localhost',
    user='prescott',
    password='embryriddle',
    database='capriotn_db',
    # prclab1
    unix_socket='/var/lib/mysql/mysql.sock'
  )
  cursor = mydb.cursor(dictionary=True)
  query = 'select team from teams'
  cursor.execute(query)
  res = cursor.fetchall()
  print('\nteams:')
  for i in res:
      print(i)

except mysql.connector.Error as error:
    print(error)
finally:
    if mydb.is_connected():
        cursor.close()
        mydb.close()
```

The first 11 lines are going to be featured in every connector in this exercise, it is the code that contains the login information required to establish a connection to the database. The last 6 lines of code are for error checking, they make sure that a connection is established and will trigger if the connection fails. The query is defined on the line 'query ='. The execution takes place on the next line 'cursor.execute(query)'. The response is read on the line following that. After that, a simple for loop goes through the entire table and prints each team name.

```
[capriotn@prclab1:~/CS317/assignment_five]$ python2 hw5_1.py

teams:
{u'team': u'Bears'}
{u'team': u'Cardinals'}
{u'team': u'Cowboys'}
{u'team': u'Packers'}
{u'team': u'Rampage'}
{u'team': u'Ropers'}
```

For the next part, we will create an SQL procedure that does the same thing, the procedure is created with the ‘create procedure’ tag, where the name is defined and the query is given within. There is much less code required here because there is no need for a connector as the procedure runs directly in the database.

```
drop procedure if exists teamNames;

delimiter //
create procedure teamNames()
begin
  select team from teams;
end //
delimiter ;
```

We can run the procedure in the MySql database by calling ‘source hw5_2.sql’, followed by ‘call teamNames’ to run the procedure. The results are shown below.

```
Database changed
mysql> source hw5_2.sql
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call teamNames;
+-----------+
| team      |
+-----------+
| Bears     |
| Cardinals |
| Cowboys   |
| Packers   |
| Rampage   |
| Ropers    |
+-----------+
```

We will get the exact same info, however in this way, it's presented as a table.

For the next step, we are asked to create a procedure that uses parameters to adjust the output. In this case, we are asked to create a procedure that will return the team name of whichever player we pass as an argument. For the procedures that require the player names as inputs, I was not getting any errors in my compiling, but the names would crash when they were entered. I believe that this may have to do with the way that I created the database, with player names being one single row labeled ‘players’. I went back and rewrote my soccer_up.sql to

separate player names into first and last names, but the issue persisted. Below is the code for the procedure and my adjusted soccer_up.sql.

```
drop procedure if exists playerTeam;

delimiter //
create procedure playerTeam(
  in f_n varchar(30),
  in l_n varchar(30)
) begin
select team from players where first_name = f_n and last_name = l_n;
end //
delimiter ;
```

```
CREATE TABLE players (
    first_name VARCHAR(32) NOT NULL,
    last_name VARCHAR(32) NOT NULL,
    team VARCHAR(32),
    PRIMARY KEY(player)
    -- FOREIGN KEY(team) REFERENCES teams(team) --
    -- ON UPDATE CASCADE --
);
INSERT INTO players VALUES
    ('Nick','C.','Cardinals'),
    ('Baylor','R.','Cardinals'),
    ('Gunner','C.','Cardinals'),
    ('Mack','J.','Cardinals'),
    ('Tyler','T.','Cardinals'),
    ('Juice','W.','Cowboys'),
    ('Kanye','W.','Cowboys'),
    ('Aaron','R.','Cowboys'),
    ('Michael','J.','Cowboys'),
    ('Donald','T.','Cowboys'),
    ('Joe','B.','Rampage'),
    ('Mich','J.','Rampage'),
    ('Josh','D.','Rampage'),
    ('Ian','C.','Rampage'),
    ('Pablo','C.','Rampage'),
    ('DeShaun','W.','Ropers'),
    ('Kyler','M.','Ropers'),
    ('Josh','Z.','Ropers'),
    ('Jack','L.','Ropers'),
    ('Jonsey','L.','Ropers'),
    ('Kirby','N.','Packers'),
    ('Chris','P.','Packers'),
    ('Robert','D.','Packers'),
    ('Clark','N.','Packers'),
    ('Josiah','K.','Packers'),
    ('Jordan','L.','Bears'),
    ('Brett','F.','Bears'),
    ('Joe','N.','Bears'),
    ('Throckmorton','S.','Bears'),
    ('Master','C.','Bears');
```

With this query we are passing it 2 variables, a first_name → f_n and a last_name → l_n these 2 values are used to check for first_name and last_name matches in the where clause, creating a filter where the only team name will be that of the child with the matching first and last names, however my query would crash when the first_name was entered without throwing an error, I was unable to rectify the issue.

Next, we are asked to create a connector that calls a procedure. We're going to recycle the procedure from earlier in this assignment that gets the list of all of the team names. Our connector will then need to display the data returned by the procedure, much like the earlier connector, the code is shown below.

```
import mysql.connector

try:
  mydb = mysql.connector.connect(
    host='localhost',
    user='prescott',
    password='embryriddle',
    database='capriotn_db',
    # prclab1
    unix_socket='/var/lib/mysql/mysql.sock'
  )
  cursor = mydb.cursor()
  cursor.callproc('teamNames')
  print('\nTeams\n')
  for res in cursor.stored_results():
      for row in res:
          print(row[0])
  print('')

except mysql.connector.Error as error:
    print(error)
finally:
    if mydb.is_connected():
        cursor.close()
        mydb.close()
```

You can see that we have the basic connector code to establish a connection, then we use “cursor.callproc(‘procedure_name’)” to call the procedure ‘teamNames’, then the for loop goes through and displays all of the team names on their own line, the result is shown below.

```
[capriotn@prclab1:~/CS317/assignment_five]$ python2 hw5_4.py

Teams

Bears
Cardinals
Cowboys
Packers
Rampage
Ropers
```

Each Team has their name printed to the terminal on its own line.

Next, we are tasked with adding a connector to the other procedure we created earlier, the playerTeam procedure. This will be simple enough, with the only real difference being that in the connector, we need to add the arguments as inputs. The code for this is shown below. As with the prior examples with the playerTeam, I don't get any errors when compiling, but I do get errors when I go to enter the players name. The code is shown below.

```
import mysql.connector

print(' ')
f = input('First: ')
l = input('Last: ')
try:
  mydb = mysql.connector.connect(
    host='localhost',
    user='prescott',
    password='embryriddle',
    database='capriotn_db',
    # prclab1
    unix_socket='/var/lib/mysql/mysql.sock'
  )
  a = [f, l]
  cursor = mydb.cursor()
  cursor.callproc('playerTeam', a)
  print('\n' + f + ' ' + l + '\'s Team:')
  for res in cursor.stored_results():
      for row in res:
          print(row[0])
  print(' ')

except mysql.connector.Error as error:
    print(error)
finally:
    if mydb.is_connected():
        cursor.close()
        mydb.close()
```

This is like the other connectors, but the inputs are taken in and stored in the values ‘f’ and ‘l’ for the first and last names. Then we place the values we just declared and place them in the array. This array of arguments is then passed in on line 15, along with the name of the procedure with the same ‘callproc()’ command. Then we use the same basic for loop that we have been using to iterate through and print all of the results.

The last step is to create another procedure and another connector, both of which will have input arguments like the prior example. In this example, we are going to display the schedule for any given player, however this is a query we already have written in a prior exercise. So we only need to create the procedure and the connector with the ability to take the first and last names as inputs, as done above.

```
import mysql.connector

print(' ')
f = input('First: ')
l = input('Last: ')
try:
  mydb = mysql.connector.connect(
    host='localhost',
    user='prescott',
    password='embryriddle',
    database='capriotn_db',
    # prclab1
    unix_socket='/var/lib/mysql/mysql.sock'
  )
  a = [f, l]
  cursor = mydb.cursor()
  cursor.callproc('playerSchedule', a)
  print('\n' + f + ' ' + l + '\'s Schedule:')
  for res in cursor.stored_results():
      for row in res:
          dateTime = row[1].strftime("%B %d at %I:%M %p")
          print(row[0] + ' plays the ' + row[3] + ' on ' + dateTime)
          print('  Their coach is ' + row[4] + '.')
          if (row[2] > 0):
              print('  ' + row[0] + ' brings the snacks.')
          else:
              print('  They bring the snacks.')
  print(' ')

except mysql.connector.Error as error:
    print(error)
finally:
    if mydb.is_connected():
        cursor.close()
        mydb.close()
```

As with the prior queries, The result was not an error, but a complete failure when the players first_name is entered. I spoke with another student in the class who could not help fix the crashing error.

With the connector, the only change I had to make was how the data gets displayed to the console. Everything else was kept pretty much the same, however I could not get the table to produce an output, I assume that the formatting is correct for the output. I think the fault lies in the soccer_up file for exercise 3B. I only had issues performing operations on the one table, all of the other tables (appear) to be working with the connectors and procedures.

```
drop procedure if exists playerSchedule;

delimiter //
create procedure playerSchedule(
  in fn varchar(30),
  in ln varchar(30)
) begin
select
  concat(p.first_name, ' ', p.last_name) as player,
  g.game_date,
  s.host,
  (
    select
      s2.team
    from
      schedule s2
    where
      s.game_id = s2.game_id
      and s.team <> s2.team
  ) as playing_against,
  (
    select
      coach
    from
      teams
    where
      team = playing_against
  ) as against_coach
from
  players p
  inner join schedule s on s.team = p.team
  inner join games g on g.game_id = s.game_id
where
  p.first_name = fn
  and p.last_name = ln;

end //
delimiter ;
```

Overall, I was successful in implementing the connectors and procedures. The issues that I faced were (in my opinion) not related to this assignment, but rather the implementation of other tables in the earlier exercise. This assignment was a good way to practice using connectors and procedures directly on MySql. I think that if I had been more careful with the separation of first and last names in assignment 3B I would have had an easier time trying to debug these procedures.