**Laboratoire 3-4**

par

Alexandre Prud’Homme 7293804

Julien Desautels 7331746

Soumis à

Dr. Mok Beldjehem

dans le cadre du cours

Base de données II

CSI 3530

Université d’Ottawa

Le 2 décembre 2018

1. « Impedance mismatch » se produit en raison que le langage SQL fonctionne avec les ensembles de records. Comparativement à d’autre langages de programmation, comme le langage C, il ne supporte pas l’abstraction de l’ensemble de records.
2. Il y a deux étapes à suivre afin de passer la valeur d’une variable dans un langage hôte à une requête SQL. Premièrement, il faut déclarer la variable entre les deux commandes suivantes.

EXEC SQL BEGIN DECLARE SECTION

EXEC SQL END DECLARE SECTION

Deuxièmement, les variables peuvent être utilisées lorsqu’il est précédé par la ponctuation du deux points (:).

1. La commande « WHENEVER » vérifie la valeur de « SQLSTATE » pour une erreur spécifique. Lors d’une erreur, la commande « WHENEVER » va transférer le contrôle à la section spécifique. La commande « WHENEVER » est utile puisqu’elle est simple dans la vérification d’erreurs de d’exceptions.
2. Il y a un besoin pour les curseurs puisqu’ils fournissent un mécanisme de récupération de rangées une à la fois d’une relation.
3. Un exemple d’une situation où l’utilisation de commandes SQL intégrées est nécessaire est lors du calcul de la variance. Par défaut, le langage SQL n’a pas une fonction d’agrégation qui peut calculer la variance. Alors, afin de faire la manipulation d’agrégats plus sophistiqués, l’utilisation de code SQL intégré est nécessaire.

À noter, les numéros 6, 7, 8 utilise tous le même code de base pour importer l’information de la base de données.

import psycopg2

import numpy

class sailor:

def \_\_init\_\_(self,theID,name,rating,age):

self.theID = theID

self.name = name

self.rating = rating

self.age = age

try:

# Connection

connection = psycopg2.connect(user = "postgres",

password = "\*\*\*\*\*\*\*\*",

host = "localhost",

port = "5432",

database = "CSI3530")

cursor = connection.cursor()

print ( connection.get\_dsn\_parameters(),"\n")

cursor.execute("SELECT version();")

record = cursor.fetchone()

print("You are connected to - ", record,"\n")

#Cursor to fetch data from the database

cursor = connection.cursor()

#Get all sailors from the database

get\_sailors\_query = "SELECT \* FROM sailors;"

cursor.execute(get\_sailors\_query)

sailors = cursor.fetchall()

#Create a list of sailor object to facilitate the data usage

listOfSailors = []

for aSailor in sailors:

listOfSailors.append(sailor(aSailor[0],aSailor[1], aSailor[2], aSailor[3]))

#Get the age of all sailors

listOfAges = []

for aSailor in listOfSailors:

listOfAges.append(aSailor.age)

Avec en fin de programme un peut de code pour contrôler les erreurs ainsi que s’assurer que la connexion a la base de données soit fermée à la fin.

except (Exception, psycopg2.Error) as error :

print ("Error while connecting to PostgreSQL", error)

finally:

#closing database connection.

if(connection):

cursor.close()

connection.close()

print("\nPostgreSQL connection is closed")

Le code de numéros 6, 7 et 8 se trouve entre le premier et le deuxième bloc de code ci-dessus.

1. La variance de « sailor’s age » avec le langage de programmation Python.

#Find the age variance

ageVariance = numpy.var(listOfAges)

print("The age variance is: " + str(ageVariance) + "\n")

1. La déviation standard de « sailor’s age » avec le langage de programmation Python.

#Find the age standard deviation

ageStanDev = numpy.std(listOfAges)

print("The age standard deviation is: " + str(ageStanDev) + "\n")

1. La déviation standard de la moyenne de « sailor’s age » avec le langage de programmation Python.

#Find the average age of sailors

ageAvr = numpy.average(listOfAges)

print("The averag age is: " + str(ageAvr) + "\n")

#Find all the sailors in one standard variation from the average

listOfSailorsInSD = []

upperBound = ageAvr + ageStanDev

lowerBound = ageAvr - ageStanDev

print("This is a list of all the sailors with ages in one standard deviation of the average: ")

for aSailor in listOfSailors:

if (aSailor.age < upperBound) and (aSailor.age > lowerBound):

listOfSailorsInSD.append(aSailor)

print(str(aSailor.name) + ": " + str(aSailor.age))

1. Afin de calculer la fermeture transitive d’un graphe, il faut utiliser l’algorithme de Floyd-Warshall, tel que nous avons appris dans le cours de Conception et analyse des algorithmes (CSI3505). L’algorithme consiste à sélectionner tous les records de la table des arêtes afin de créer la matrice d’adjacence. De cette matrice d’adjacence on peut calculer les algorithmes de fermetures transitives.
2. « Updatability » : « Updatability » est une propriété des curseurs qui détermine si le curseur peut être utilisé pour les commandes SQL de mise-à-jour (UPDATE) ou de délétion (DELETE). Un curseur déclaré pour une mise-à-jour (FOR UPDATE) est capable d’exécuter des requêtes qui modifient des relations.

« Sensitivity » : Lorsqu’un curseur est déclaré non sensible (INSENSITIVE), il a les propriétés privées dont seulement la lecture de la copie des résultats de la requête. Des mis-à-jour concourants de l’information ne change pas le curseur. Sans la déclaration non sensible, il peut se produire plusieurs problèmes de concurrence.

« Scrollability » : « Scrolability » est directement lié avec la commande de récupération (FETCH). La récupération est capable de récupérer un défilé (SCROLL) où le défilé est en mesure de parcourir des options comme le dernier (LAST) et le précédent (PRIOR).

1. Dans postgreSQL les curseur updatables ne sont pas supportés, l’exemple 1 représente un index no updatable en postgreSQL et l’exemple 2 représente un curseur en SQL pure qui lui est updatable.
2. DECLARE cursor\_SailorAge SCROLL CURSOR WITH HOLD

FOR

SELECT S.sname, S.age

FROM Sailors S

ORDER BY S.age ASC

1. DECLARE SailorAge SCROLL CURSOR

FOR

SELECT S.sname, S.age

FROM Sailors S

ORDER BY S.age ASC

FOR UPDATE

Il faut note que ces curseurs ne peuvent pas updater la valeur de l’âge des sailors, puisque le curseur est en ordre d’âge des sailors.

1. L’exécution de SQL dynamique est requise lorsque les déclarations ne sont pas connues au moment de la compilation. Un exemple de ceci est lorsqu’un utilisateur a le choix de faire l’insertion, la délétion ou la mise-à-jour des valeurs d’une table par le moyen d’une interface graphique.