Ronalds Rundāns rr11043

2.praktiskais darbs (PD2)
"Lielais futbols" risinājuma apraksts

Izstrādes OS un vide : 22 Ubuntu Mate, VisualStudioCode
Programmēšanas valoda : Python valoda
Datu bāze : SQLite fails
Datu ievade : XML faili

DB ir 2 tabulas : TEAM un PLAYER
dati glabājas SQLite failā, kas atrodas tajā pašā mapē, kur ir darbināmā programma. Šādi veido datu tabulas. Nav saites starp tabulām, bet, ja būtu jāveido JOIN, tad var izmantot "NAME" un "KOM_NOS" kolonnas, kas abās tabulās glabā komandas nosaukumu.

```
CREATE TABLE TEAM (Name VARCHAR(255) NOT NULL,PUN_TOT INT, USKPM INT, ZSKPM INT, UZSKPL INT, ZSKPL INT, IEGV INT, ZAUV INT,RINKIS VARCHAR(255)
```

```
CREATE TABLE PLAYER (NR INT, KOM_NOS VARCHAR(255) NOT NULL,VARDS VARCHAR(255) NOT NULL,UZVARDS VARCHAR(255) NOT NULL,VARTI INT,PIESP INT,SODI INT,SPELUSK INT,SPELUSEC INT,KARTEDZE INT,KARTESAR INT,LOMA VARCHAR(5))
```

Ja nepieciešams dzēst tabulas:

```
DROP TABLE IF EXISTS TEAM
DROP TABLE IF EXISTS PLAYER
```

Programmā ir tikai viena klase "Team", kas glabā informāciju par komandas rezultātiem, kuros beigās piešķir punktus atkarībā no uzvaras vai zaudējuma pamatlaikā vai papildlaikā un tikai tad ievada šo informāciju datu bāzē. Katrā spēlē masīvā "tmp" pievieno 2 "Team" objektus t.i. viena komanda("tmp[0]") un otra komanda(tmp[1]).

```
class Team:
def __init__(self, name):
self.name = name #Komandas nosaukums
self.PUN_TOT=0 #Iegūto punktu skaits
self.USKPM=0 #Uzvaru skaits pamatlaikā
self.ZSKPM=0 #Zaudējumu skaits pamatlaikā
self.UZSKPL=0 #Uzvaru skaits papildlaikā
self.ZSKPL=0 #Zaudējumu skaits papildlaikā
self.IEGV=0 #Iegūto vārtuu skaits
self.ZAUV=0 #Zaudēto vārtu skaits
self.vpm=0 #Iegūto vārtu skaits pamatlaikā (tikai šajā spēlē) - nosaka piešķirtos punktus
self.vpl=0 #Iegūto vārtu skaits papildlaikā (tikai šajā spēlē) - nosaka piešķirtos punktus
self.laukuma=[]#Pamatsastāva numuri
self.mainaNr1=[]#Spēlētāju numuri, kuri tiek nomainīti
self.mainaNr2=[]#Spēlētāja numuri, kuri uzsāk spēli nomainītā spēlētāja vietā
self.mainaT=[]#Maiņas laiks
self.sodiNr=[]#Spēlētājiem sodi
#END of class Team
```

Datu bāzē informāciju par katru komandas spēles rezultātiem ievada šādi(str funkcija int mainīgo vērtības pārveido par string vērtībām, lai var konkatenēt vaicājumu):

```
"INSERT INTO TEAM VALUES ('"+tmp[0].name+"', '"+str(tmp[0].PUN_TOT)+"',
'"+str(tmp[0].USKPM)+"','"+str(tmp[0].ZSKPM)+"','"+str(tmp[0].UZSKPL)
+"','"+str(tmp[0].ZSKPL)+"','"+str(tmp[0].IEGV)+"','"+str(tmp[0].ZAUV)+"','"+rinkis+"')"
"INSERT INTO TEAM VALUES ('"+tmp[1].name+"', '"+str(tmp[1].PUN_TOT)+"',
'"+str(tmp[1].USKPM)+"','"+str(tmp[1].ZSKPM)+"','"+str(tmp[1].UZSKPL)
+"','"+str(tmp[1].ZSKPL)+"','"+str(tmp[1].IEGV)+"','"+str(tmp[1].ZAUV)+"', '"+rinkis+"')"
```

Statistiku par komandu atlasa šādi:

```
"SELECT name,sum(PUN_TOT), sum(USKPM), sum(ZSKPM), sum(UZSKPL), sum(ZSKPL),
sum(IEGV), sum(ZAUV) FROM TEAM "+where+" group by name order by sum(PUN_TOT)
desc"
```

, kur mainīgais "where" var glabāt papildus infomāciju, piemēram " WHERE RINKIS='1.rinkis' ", vai glabāt tukšu simbolu virkni "".

Lai izvairītos no dublikātiem, kas rodas no vairākām spēlēm, tad pirms pievieno spēlētāju datu bāzei, vispirms pārbauda, vai jau eksistē šāds spēlētājs. Ja nav, tad pievieno jaunu.

```
"SELECT DISTINCT * FROM PLAYER WHERE NR="+str(nr)+" AND VARDS='"+name+"' AND
UZVARDS='"+surname+"' AND KOM_NOS='"+team+"'"
"INSERT INTO PLAYER VALUES ("+str(nr)+",'"+team+"','"+name+"','"+surname+"',0,0,0)"
```

, kur pēdējas trīs nulles ir VARTI, PIESP un SODI.

Ja pēdējās vērtības mainās spēles gaitā – iesit vārtus, piespēle vai sods, tad mainīgajā nokopē esošo vērtību, palielina par 1 un izpilda attiecīgo Update komandu(piemēram iesit vārtus):

```
"UPDATE PLAYER SET VARTI = "+str(varti+1)+" WHERE NR = "+str(nr)+" AND
KOM_NOS='"+team+"'"
```

Top 100 spēlētāju statistiku iegūst šādi:

```
"SELECT VARDS, UZVARDS, NR , KOM_NOS, VARTI, PIESP FROM PLAYER ORDER BY VARTI
DESC, PIESP DESC LIMIT 100"
```

XML failu lasīšanu nodrošina Python bibliotēka

```
import xml.etree.ElementTree as ET
```

, kas ļauj viegli iterēt ciklos attiecīgo XML elementu

```
for speletaji in komanda.iter('Speletaji'):          #<Komanda Nosaukums="Barcelona">
        for speletajs in speletaji.iter('Speletajs'):#       <Speletaji>
        surname = speletajs.get ("Uzvards")      #<Speletajs Loma="V" Uzvards="Sam"
                                                 #                Vards="Sidney" Nr="16"/>
```

Programmas funkcija, ko izmantoju pārbaudei:

```
def mainTest():
        print("1.rinkis")
        path="/home/ubuntu/Documents/ModProgrMet/PD2/XML_TestData/XMLFirstRound/"
        m1=" WHERE RINKIS='1.rinkis'"
        for i in range (0,3):
                readinputfile(i, path,'1.rinkis')
        teamStatistics(m1)
        print("2.rinkis")
        path1="/home/ubuntu/Documents/ModProgrMet/PD2/XML_TestData/XMLSecondRound/"
        m2=" WHERE RINKIS='2.rinkis'"
        for i in range (0,3):
                readinputfile(i, path1,'2.rinkis')
        teamStatistics(m2)
        print("1.un 2.rinkis")
        teamStatistics("")
        print(" ")
        playerStatistics()
```