НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Лабораторна робота №2

з дисципліни
«Бази даних і засоби управління»

Тема: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав: студент III курсу
ФПМ групи КВ-84
Мелюх В. В.
Перевірив:Петрашенко А.В.

Київ – 2020

Загальне завдання:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

URL репозиторію (повний):

https://github.com/PointProgram/DataBase/tree/master/Database%20and%20management%20tools(Lab%202)

URL репозиторію (неповний):

https://github.com/PointProgram/

Завдання 1:

Insert:

Table: match

```
Choose option:
PRESS: 1 to add... 2 to update...  3 to delete...  4 to specific_select... 5 to show_table...
1
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
1
Do ypo want to insert rows manually or randomly?
Press M if manually, R if randomly: M
What number of row do you want to add? - 1
Enter values following this sequence: Match_id, opp_score, own_score
Match_id: 1232
opp_score: 12
own_score: 5

1 Record successfully inserted
Do you want to continue? Press Y if yes, N if no:
```

```
PRESS: 1 to add... 2 to update...  3 to delete...  4 to specific_select... 5 to show_table...
5
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
1
Show each row and it's columns values
Match_id  opp_score  own_score
1         2          1
2         4          5
3         2          2
```

1123	2	3
1232	12	5
2342	2	3

```
Enter values following this sequence: Match_id, opp_score, own_score
Match_id: 1
opp_score: 3
own_score: 2

Failed inserting record into table ПОМИЛКА:  повторювані значення ключа порушують обмеження унікальності "Match_pkey"
DETAIL:  Ключ (match_id)=(1) вже існує.

Do you want to continue? Press Y if yes, N if no:
```

```
Match_id: f
You entered wrong instance, please try again!
Do you want to exit? Press M to go to the main menu, or E to exit:
```

```
Enter values following this sequence: Match_id, opp_score, own_score
Match_id: 2366
opp_score: r
You entered wrong instance, please try again!
Do you want to exit? Press M to go to the main menu, or E to exit:
```

```
Enter values following this sequence: Match_id, opp_score, own_score
Match_id: 5353
opp_score: 2
own_score: .
You entered wrong instance, please try again!
Do you want to exit? Press M to go to the main menu, or E to exit:
```

Table: stadium

```
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
2
Do ypo want to insert rows manually or randomly?
Press M if manually, R if randomly: M
What number of row do you want to add? - 1
Enter values following this sequence: Stadium_id, Name, City
Stadium_id: 3234
Name: Great stadium
City: Kyiv

1 Record successfully inserted
Do you want to continue? Press Y if yes, N if no:
```

54	fe	wr
445		lile
3234	Great stadium	Kyiv

```
Enter values following this sequence: Stadium_id, Name, City
Stadium_id: 2
Name: FR
City: GRY

Failed inserting record into table ПОМИЛКА:  повторювані значення ключа порушують обмеження унікальності "Stadium_pkey"
DETAIL:  Ключ (stadium_id)=(2) вже існує.
```

```
Enter values following this sequence: Stadium_id, Name, City
Stadium_id: 445
Name:
City: lile

1 Record successfully inserted
```

Table: startdate

```
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
3
Do ypo want to insert rows manually or randomly?
Press M if manually, R if randomly: M
What number of row do you want to add? - 1
Enter values following this sequence: Team_id, Match_id, Start_date(yyyy-mm-dd)
Team_id: 2
Match_id: 3
Start_date: 2018-09-12

1 Record successfully inserted
Do you want to continue? Press Y if yes, N if no:
```

2	2	2020-09-23
2	3	2018-09-12
3	4	2018-02-02

```
Enter values following this sequence: Team_id, Match_id, Start_date(yyyy-mm-dd)
Team_id: 2
Match_id: 1234
Failed, there are no records with such id: list index out of range
Do you want to exit? Press M to go to the main menu, or E to exit:
```

```
Enter values following this sequence: Team_id, Match_id, Start_date(yyyy-mm-dd)
Team_id: 3
Match_id: 4
Start_date: 2018-02-02

1 Record successfully inserted
Do you want to continue? Press Y if yes, N if no: Y
What number of row do you want to add? - 1
Enter values following this sequence: Team_id, Match_id, Start_date(yyyy-mm-dd)
Team_id: 3
Match_id: 4
Start_date: 2018-03-03

Failed inserting record into table ПОМИЛКА:  повторювані значення ключа порушують обмеження унікальності "PK_StartDate"
DETAIL:  Ключ (team_id, match_id)=(3, 4) вже існує.
```

```
Enter values following this sequence: Team_id, Match_id, Start_date(yyyy-mm-dd)
Team_id: 1
Match_id: 4
Start_date: 2-2-=2
You entered wrong date value, please try again!
```

```
Enter values following this sequence: Team_id, Match_id, Start_date(yyyy-mm-dd)
Team_id: 1
Match_id: 3
Start_date:
You entered wrong date value, please try again!
```

Table: team

```
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
4
Do ypo want to insert rows manually or randomly?
Press M if manually, R if randomly: M
What number of row do you want to add? - 1
Enter values following this sequence: Team_id, Name, Opponent, Coach, Stadium_id
Team_id: 1242
Name: Beste
Opponent: Worse
Coach: Coach
Stadium_id: 2

1 Record successfully inserted
```

956	NCPCRW	LEIMYU	ABILVD	5
1242	Beste	Worse	Coach	2
2326	ret	gef		3

```
Enter values following this sequence: Team_id, Name, Opponent, Coach, Stadium_id
Team_id: 1234
Name: fff
Opponent: ddd
Coach: dfdfd
Stadium_id: 456774
Failed, there are no records with such id: list index out of range
```

```
Enter values following this sequence: Team_id, Name, Opponent, Coach, Stadium_id
Team_id: 2326
Name: ret
Opponent: gef
Coach:
Stadium_id: 3

1 Record successfully inserted
```

```
Enter values following this sequence: Team_id, Name, Opponent, Coach, Stadium_id
Team_id: 2535
Name:
Error that column cannot contain NULL values!
```

Update:

Table: match

```
Choose option:
PRESS: 1 to add... 2 to update...  3 to delete...  4 to specific_select... 5 to show_table...
2
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
1
What number of rows do you want to update? - 1
Enter values following this sequence opp_score, own_score, Match_id:
opp_score: 4
own_score: 4
Match_id: 5

1 Records Updated
```

4	1	0
5	2	2
8	5	7

4	1	0
5	4	4
8	5	7

```
Enter values following this sequence opp_score, own_score, Match_id:
opp_score:
Error that column cannot contain NULL values!
```

```
Enter values following this sequence opp_score, own_score, Match_id:
opp_score: 1
own_score: 1
Match_id: 452635
Failed, there are no records with such id: list index out of range
```

```
Enter values following this sequence opp_score, own_score, Match_id:
opp_score: 3
own_score: g
You entered wrong instance, please try again!
```

Table: stadium

```
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
2
What number of rows do you want to update? - 1
Enter values following this sequence Name, City, Stadium_id:
Name: Lviv
City: Paris
Stadium_id: 6

1 Records Updated
```

```
5          Kotram     Kamyanka
6          FRRR       dfFF
7          XTFRKJ     QOMFVD
```

```
5          Kotram     Kamyanka
6          Lviv       Paris
7          XTFRKJ     QOMFVD
```

```
Enter values following this sequence Name, City, Stadium_id:
Name: Roty
City: ddf
Stadium_id: 225735
Failed, there are no records with such id: list index out of range
```

```
Enter values following this sequence Name, City, Stadium_id:
Name: Frop
City:
Error that column cannot contain NULL values!
```

Table: startdate

```
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
3
What number of rows do you want to update? - 1
Enter values following this sequence Start_date(yyyy-mm-dd), Team_id, Match_id:
Start_date: 2000-02-01
Team_id: 3
Match_id: 4

1 Records Updated
```

```
2          3              2018-09-12
3          4              2018-02-02
5          5              2020-09-25
```

```
2          3              2018-09-12
3          4              2000-02-01
5          5              2020-09-25
```

```
Enter values following this sequence Start_date(yyyy-mm-dd), Team_id, Match_id:
Start_date: 45
You entered wrong date value, please try again!
```

```
Enter values following this sequence Start_date(yyyy-mm-dd), Team_id, Match_id:
Start_date: 2017-03-03
Team_id: 1
Match_id: 56
Failed, there are no records with such id: list index out of range
```

```
Enter values following this sequence Start_date(yyyy-mm-dd), Team_id, Match_id:
Start_date: 2020-02-02
Team_id: 3
Match_id: 3

0 Records Updated
```

```
Enter values following this sequence Start_date(yyyy-mm-dd), Team_id, Match_id:
Start_date: 2010-02-02
Team_id: f
You entered wrong instance, please try again!
```

Table: team

```
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
4
What number of rows do you want to update? - 1
Enter values following this sequence Name, Opponent, Coach, Stadium_id, Team_id:
Name: Lokre
Opponent: Orek
Coach: Trainee
Stadium_id: 4
Team_id: 6

1 Records Updated
```

```
5          Lutsechk   Prypiat    Tymochek D.F.   5
6          Bglgl      gfgfgfgf   gfgfgf          1
7          Broslaw    Corole     Lortus          4
```

```
5          Lutsechk   Prypiat    Tymochek D.F.   5
6          Lokre      Orek       Trainee         4
7          Broslaw    Corole     Lortus          4
```

```
Enter values following this sequence Name, Opponent, Coach, Stadium_id, Team_id:
Name: fff
Opponent: ddd
Coach: hhh
Stadium_id: 13424
Failed, there are no records with such id: list index out of range
```

```
Enter values following this sequence Name, Opponent, Coach, Stadium_id, Team_id:
Name:
Error that column cannot contain NULL values!
```

```
 Enter values following this sequence Name, Opponent, Coach, Stadium_id, Team_id:
 Name: Frrg
 Opponent: Dfd
 Coach:
 Stadium_id: 3
 Team_id: 3

 1 Records Updated
```

```
Enter values following this sequence Name, Opponent, Coach, Stadium_id, Team_id:
Name: rr
Opponent: rr
Coach: rr
Stadium_id: 4
Team_id: f
You entered wrong instance, please try again!
```

Delete:

Table: match

```
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
1
What number of row do you want to delete? - 1
Enter value that marks Match_id:460
Match_id:  460

1 Record deleted
```

418	976	296
460	221	619
486	88	228

418	976	296
486	88	228

```
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
1
What number of row do you want to delete? - 1
Enter value that marks Match_id:4
Match_id:  4

Failed deleting record into table ПОМИЛКА:  update або delete в таблиці "match" порушує
```

```
обмеження зовнішнього ключа "StartDate_Match_id_fkey" таблиці "startdate"
```

```
DETAIL:  На ключ (match_id)=(4) все ще є посилання в таблиці "startdate".
```

```
Enter value that marks Match_id:134565
Match_id:  134565
Failed, there are no records with such id: list index out of range
```

Table: stadium

```
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
2
What number of row do you want to delete? - 1
Enter value that marks Stadium_id:34
Stadium_id:  34

1 Record deleted
```

```
12          222f        dfd
34          fff         fff
54          fe          wr
```

```
12          222f        dfd
54          fe          wr
```

```
Enter value that marks Stadium_id:3
Stadium_id:  3

Failed deleting record into table ПОМИЛКА:  update або delete в таблиці "stadium" порушує
```

```
порушує обмеження зовнішнього ключа "Team_Stadium_id_fkey" таблиці "team"
```

```
 DETAIL:  На ключ (stadium_id)=(3) все ще є посилання в таблиці "team".
```

```
Enter value that marks Stadium_id:4674337
Stadium_id:  4674337
Failed, there are no records with such id: list index out of range
```

Table: startdate

```
Enter values following this sequence team_id, match_id:
team_id: 5
match_id: 5

1 Record deleted
```

```
3           4               2000-02-01
5           5               2020-09-25
7           3               2019-12-12
```

```
3           4               2000-02-01
7           3               2019-12-12
```

```
Enter values following this sequence team_id, match_id:
team_id: 5
match_id: 1353
Failed, there are no records with such id: list index out of range
```

```
Enter values following this sequence team_id, match_id:
team_id: 5
match_id: 4

0 Record deleted
```

Table: team

```
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
4
What number of row do you want to delete? - 1
Enter value that mark Team_id:432
Team_id:  432

1 Record deleted
```

```
428         SLWIQS      YQCXPA      FTLUVO          5
432         gg          gg          gg              1
434         ff          ff          ff              1
```

```
428         SLWIQS      YQCXPA      FTLUVO          5
434         ff          ff          ff              1
```

```
Enter value that mark Team_id:3
Team_id:  3

Failed deleting record into table ПОМИЛКА:  update або delete в таблиці "team" порушує
```

```
обмеження зовнішнього ключа "StartDate_Team_id_fkey" таблиці "startdate"
```

```
DETAIL:  На ключ (team_id)=(3) все ще є посилання в таблиці "startdate".
```

```
Enter value that mark Team_id:336787764
Team_id:  336787764
Failed, there are no records with such id: list index out of range
```

```
Enter value that mark Team_id:
Team_id:
Error that column cannot contain NULL values!
```

Завдання 2:

```
Choose option:
PRESS: 1 to add... 2 to update...  3 to delete...  4 to specific_select... 5 to show_table...
1
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
4
Do ypo want to insert rows manually or randomly?
Press M if manually, R if randomly: R
What number of row do you want to add(type 'd' to set default)? - 100000
100000 Record successfully inserted
```

2072055	639	2000-03-08
2072056	1	2001-01-23
2072059	19	2001-01-03
2072059	544	2001-06-26
2072062	614	2001-06-01
2072063	990	2001-06-14
2072067	16	2000-08-15
2072067	954	2000-01-24
2072067	990	2000-07-18
2072068	87	2000-10-08
2072068	331	2000-04-03
2072070	17	2000-12-20
2072070	35	2000-10-12
2072070	896	2001-04-08
2072072	544	2001-06-12
2072072	568	2000-04-12
2072074	956	2001-03-16
2072076	5	2000-09-24
2072076	990	2000-06-28
2072078	87	2001-07-09
2072082	21	2000-04-08
2072082	3433	2001-06-11
2072086	33	2000-12-06
2072088	1	2001-06-20
2072088	486	2000-04-11
2072088	805	2000-12-08

```
1  Select count(*) from startdate
```

Data Output | Explain | Messages | Notificati

	count bigint
1	100008

```sql
INSERT INTO startdate (team_id, match_id, start_date)
        SELECT team_id, match_id, dat FROM
        (SELECT
        team_id, match_id,
        date '2000-01-10' + trunc(random()*10 * (date '2020-05-20' - date '2010-01-10'))::int as dat
        FROM team, match tablesample BERNOULLI(100)
        ORDER BY random()) k ,generate_series(1, 10000) LIMIT 5
Returning team_id, match_id, start_date
```

Data Output | Explain | Messages | Notifications

	team_id [PK] integer	match_id [PK] integer	start_date date
1	2032770	805	2000-01-10
2	2081946	544	2000-01-10
3	2019369	87	2000-01-10
4	2087644	30	2000-01-10
5	2094100	212	2000-01-10

```
Choose option:
PRESS: 1 to add... 2 to update...  3 to delete...  4 to specific_select... 5 to show_table...
1
Choose table:
 press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team
3
Do ypo want to insert rows manually or randomly?
Press M if manually, R if randomly: R
What number of row do you want to add(type 'd' to set default)? - 100000
100000 Record successfully inserted
```

```
2100434    3c916c9d3d757505d23b4db22a4817d1 6497ff75a0c3ff3d5fa92de05f46deb4 7e5b2d8d00c7928887ca9107851688e9 6
2100435    f22e515c478ecda32db7420e899c29f3 1cd5c3a92626f5ef13f97904f84017dc 49cfdcfd6a40ba7556115e8d5df9811f 7
2100436    f1d93ed604917d2f1b6b9164e2478f21 35ab70d0afa1bd86cca908d6fff8df99 29776b94b076a56dfb4a1324ec169e27 12
2100437    31fadfc59fe2b1a9550b4dd88a1b8553 62adf6e97b3ffa57b78d3a8e26ed34da 3c2af9ce834809a79e0d4f388c2c57e0 54
2100438    c8fb751cbf2e144aa21e10495c193a77 258465016040e27147f6fffec0358fa4 fdaaa303207741b0a4ac2293a531fa5e 5
2100439    0ad187dfba76e5269d13390b1fbac9c0 652d404815dc97d098d1253a107c2a88 420b8d08e8488bfbb25c27b5e98d37ab 445
2100440    f0ae990fd44d65601d3bb555c0d553fc 82c66077ed55c5e2acbbf50d9b9decd0 67d9222506f84c7eed420975bd3686f8 2
2100441    a645cba1592f1ba80a6011f678b5334e ea1b93d7ca2e170546c2f618c9ba10f4 ea32fffd0d9f312c6f1156477a012008 8
2100442    5a6d81773ec721807a44b107a33256c1 082a7339cea645898b799b92e57774d3 6e762f03bc9a0a331ae6c5b714c3d55c 1
2100443    b1055311f4ec77c953061e0525a0a0bb dafedda8ffe0795cb9cdb459fc388304 ace03b27506423df4e3ea8a730a604a6 9
2100444    b19e2a6493f057ca251b748439242c7c 939c987992c43c228a6f3191eadc120e 3bd0ae83be59d2d0eec4e30864e11739 3
2100445    26ca64ae01aa9b9b65e22ddc1f9c25f8 9a9b29a22280bb1407fe0ee8ae3b39c8 e9cd1bd0c6483cc8a355da328e868d21 3234
2100446    a98113f5749cdf9ba1f1d55766ba1cc8 f1ed6cc589a6b8daa1fff9dd0fded8dd ea39488dc8a30b2544cc0bb1a640577e 4
2100447    3c916c9d3d757505d23b4db22a4817d1 6497ff75a0c3ff3d5fa92de05f46deb4 7e5b2d8d00c7928887ca9107851688e9 6
2100448    f22e515c478ecda32db7420e899c29f3 1cd5c3a92626f5ef13f97904f84017dc 49cfdcfd6a40ba7556115e8d5df9811f 7
2100449    f1d93ed604917d2f1b6b9164e2478f21 35ab70d0afa1bd86cca908d6fff8df99 29776b94b076a56dfb4a1324ec169e27 12
2100450    31fadfc59fe2b1a9550b4dd88a1b8553 62adf6e97b3ffa57b78d3a8e26ed34da 3c2af9ce834809a79e0d4f388c2c57e0 54
2100451    c8fb751cbf2e144aa21e10495c193a77 258465016040e27147f6fffec0358fa4 fdaaa303207741b0a4ac2293a531fa5e 5
2100452    0ad187dfba76e5269d13390b1fbac9c0 652d404815dc97d098d1253a107c2a88 420b8d08e8488bfbb25c27b5e98d37ab 445
2100453    f0ae990fd44d65601d3bb555c0d553fc 82c66077ed55c5e2acbbf50d9b9decd0 67d9222506f84c7eed420975bd3686f8 2
2100454    a645cba1592f1ba80a6011f678b5334e ea1b93d7ca2e170546c2f618c9ba10f4 ea32fffd0d9f312c6f1156477a012008 8
2100455    5a6d81773ec721807a44b107a33256c1 082a7339cea645898b799b92e57774d3 6e762f03bc9a0a331ae6c5b714c3d55c 1
2100456    b1055311f4ec77c953061e0525a0a0bb dafedda8ffe0795cb9cdb459fc388304 ace03b27506423df4e3ea8a730a604a6 9
2100457    b19e2a6493f057ca251b748439242c7c 939c987992c43c228a6f3191eadc120e 3bd0ae83be59d2d0eec4e30864e11739 3
2100458    26ca64ae01aa9b9b65e22ddc1f9c25f8 9a9b29a22280bb1407fe0ee8ae3b39c8 e9cd1bd0c6483cc8a355da328e868d21 3234
```

```sql
Select count(*) from team
```

Data Output | Explain | Messages | Notifications

	count bigint
1	100053

```sql
INSERT INTO team (name, opponent, coach, stadium_id)
        SELECT nam, opp, coach, stadium_id FROM
        (SELECT md5((random()*1)::text) as nam,
            md5((random()*2)::text) as opp,
            md5((random()*3)::text) as coach,
            stadium_id
        FROM stadium tablesample BERNOULLI(100)
        ORDER BY random()) k, generate_series(1, 10000) LIMIT 5
Returning team_id ,name, opponent, coach, stadium_id
```

Data Output | Explain | Messages | Notifications

	team_id [PK] integer	name character varying (50)	opponent character varying (50)	coach character varying (50)	stadium_id integer
1	2100474	c487fefe98be7de3e12e854...	2f2f4648fcd1e0d13e3bd698...	116e8a42b5d0b50533dccae...	6
2	2100475	5a2321708d56fd002907bc1...	8e072eb3b5d92e7f2dc1ceff...	60c70d1d82e983385bd24d...	54
3	2100476	acc7a2fdfca12c1ad695497...	d6af7d47540eb9d1cad3cc7...	f3d473b72d82ff553735cc6a...	445
4	2100477	5be8b5154f708795053bf6a...	e34a6d3c96519136e0ec8bd...	55214af4bd0e969190a0311...	2
5	2100478	6427ab105780510fc393ff6...	b41c45397aaaf7040526b5d...	6a098a260b1c953535c2c6e...	1

```sql
sql_random_query = """ INSERT INTO match (opp_score, own_score)
SELECT
      trunc(random()*1000)::int,
      trunc(random()*1000)::int
FROM
    generate_series(1, %s)
"""
```

```sql
sql_random_query = """ INSERT INTO stadium (name, city)
SELECT
        chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
        chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
        chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),
        chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
        chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
        chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)
FROM
    generate_series(1, %s)
"""
```

```
sql_random_query = """ INSERT INTO startdate (team_id, match_id, start_date)
SELECT team_id, match_id, dat FROM
(SELECT
team_id, match_id,
date '2000-01-10' + trunc(random()*10 * (date '2020-05-20' - date '2010-01-10'))::int as dat
FROM team, match tablesample BERNOULLI(100)
ORDER BY random()) k ,generate_series(1, 10000) LIMIT %s
"""
```

```
sql_random_query = """ INSERT INTO team (name, opponent, coach, stadium_id)
SELECT nam, opp, coach, stadium_id FROM
(SELECT md5((random()*1)::text) as nam,
    md5((random()*2)::text) as opp,
    md5((random()*3)::text) as coach,
    stadium_id
FROM stadium tablesample BERNOULLI(100)
ORDER BY random()) k, generate_series(1, 10000) LIMIT %s
"""
```

Завдання 3:

1.

```
sql_specific_query = """ SELECT team_id, name as team_name, coach, opponent, stadium_name
FROM team  AS a INNER JOIN
(SELECT name AS stadium_name, stadium_id FROM stadium WHERE name LIKE %s) AS aa
ON a.stadium_id=aa.stadium_id
WHERE opponent != %s AND team_id in (SELECT team_id FROM startdate WHERE start_date > %s)
"""
```

2.

```
sql_specific_query = """ WITH t_n AS (SELECT team_id FROM team WHERE length(name) < %s
AND name NOT SIMILAR TO %s)
SELECT start_date, opp_score, own_score FROM startdate  AS a INNER JOIN
(SELECT match_id, opp_score, own_score FROM
match WHERE opp_score = own_score OR opp_score - own_score = %s) as aa
ON a.match_id=aa.match_id
WHERE team_id IN (SELECT team_id FROM t_n)
"""
```

3.

```
sql_specific_query = """ SELECT tid AS team_id,  sum(cnt) sum_team, cc.n AS team_name,
cc.o AS opponent_name FROM (
SELECT team_id tid, count(team_id) cnt, n, o FROM startdate INNER JOIN
(SELECT team_id, name AS n, opponent AS o FROM team WHERE name LIKE %s) AS t USING (team_id)
WHERE start_date BETWEEN %s AND %s
GROUP BY team_id, t.n, t.o) cc
WHERE tid < %s
GROUP BY tid, cc.n, cc.o
ORDER BY tid
"""
```

```
Choose option:
PRESS: 1 to add... 2 to update...  3 to delete...  4 to specific_select... 5 to show_table...
4
1) Show all info about team(name, coach, opponent), which played game on specific 'stadium', but
                  not with that 'opponent', when matches took place later than that 'date'.

2) Show date, opp_score, own_score from those teams that have a draw or the difference between
                  home and away teams in 'number of goal' and the length of home team is less then 'that number', except of 'such team'.

3) Show team_id, sum_team, team_name, opponent_name from those teams that have 'specific name',
                  where match took place between 'date_1' and 'date_2' and team_id is less than 'that number'

Choose table:
Press: 1 or 2 or 3
```

```
1
Enter values following this sequence stadium_name, opponent, start_date:
stadium_name: %
opponent: Worskla
start_date: 2018-01-01

Time of request 25  ms
Show each row and it's columns values
team_id  team_name  coach     opponent    stadium_name
7          Broslaw    Lortus     Corole      Karpaty
1          Shakhtar   Mokerc A.V. Beshekt    Olimp
2          Dynamo     Bytuchuk B.L. Hover       Datum
```

```
1  select team_id, name as team_name, coach, opponent, stadium_name from team  as a inner join
2    (select name as stadium_name, stadium_id from stadium where name LIKE '%') as aa
3    on a.stadium_id=aa.stadium_id
4    where opponent != 'Worskla' and team_id in (Select team_id from startdate where start_date > '2018-01-01')
```

Data Output | Explain | Messages | Notifications

	team_id integer	team_name character varying (50)	coach character varying (50)	opponent character varying (50)	stadium_name character varying (50)
1	7	Broslaw	Lortus	Corole	Karpaty
2	1	Shakhtar	Mokerc A.V.	Beshekt	Olimp
3	2	Dynamo	Bytuchuk B.L.	Hover	Datum

```
1  EXPLAIN ANALYZE
2  select team_id, name as team_name, coach, opponent, stadium_name from team  as a inner join
3    (select name as stadium_name, stadium_id from stadium where name LIKE '%') as aa
4    on a.stadium_id=aa.stadium_id
5    where opponent != 'Worskla' and team_id in (Select team_id from startdate where start_date > '2018-01-01')
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
4	Group Key: startdate.team_id Read-only column
5	-> Seq Scan on startdate (cost=0.00..1791.10 rows=10 width=4) (actual time=0.106..13.209 rows=5 loops=1)
6	Filter: (start_date > '2018-01-01'::date)
7	Rows Removed by Filter: 100008
8	-> Index Scan using "Team_pkey" on team a (cost=0.42..8.44 rows=1 width=104) (actual time=0.008..0.008 rows=1 loops=3)
9	Index Cond: (team_id = startdate.team_id)
10	Filter: ((opponent)::text <> 'Worskla'::text)
11	-> Index Scan using "Stadium_pkey" on stadium (cost=0.15..0.17 rows=1 width=122) (actual time=0.003..0.004 rows=1 loops=3)
12	Index Cond: (stadium_id = a.stadium_id)
13	Filter: ((name)::text ~~ '%'::text)
14	Planning Time: 0.766 ms
15	Execution Time: 13.510 ms

```
1
Enter values following this sequence stadium_name, opponent, start_date:
stadium_name: Karpaty
opponent: Worskla
start_date: 2018-01-01

Time of request 20  ms
Show each row and it's columns values
team_id  team_name  coach     opponent    stadium_name
7          Broslaw    Lortus     Corole     Karpaty
```

```
1  --EXPLAIN ANALYZE
2  select team_id, name as team_name, coach, opponent, stadium_name from team  as a inner join
3    (select name as stadium_name, stadium_id from stadium where name LIKE 'Karpaty') as aa
4    on a.stadium_id=aa.stadium_id
5    where opponent != 'Worskla' and team_id in (Select team_id from startdate where start_date > '2018-01-01')
```

Data Output | Explain | Messages | Notifications

	team_id integer	team_name character varying (50)	coach character varying (50)	opponent character varying (50)	stadium_name character varying (50)
1	7	Broslaw	Lortus	Corole	Karpaty

```
1  EXPLAIN ANALYZE
2  select team_id, name as team_name, coach, opponent, stadium_name from team  as a inner join
3    (select name as stadium_name, stadium_id from stadium where name LIKE 'Karpaty') as aa
4    on a.stadium_id=aa.stadium_id
5    where opponent != 'Worskla' and team_id in (Select team_id from startdate where start_date > '2018-01-01')
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
5	-> Seq Scan on startdate (cost=0.00..1791.10 rows=10 width=4) (actual time=0.038..16.183 rows=5 loops=1)
6	Filter: (start_date > '2018-01-01'::date)
7	Rows Removed by Filter: 100008
8	-> Index Scan using "Team_pkey" on team a (cost=0.42..8.44 rows=1 width=104) (actual time=0.012..0.012 rows=1 loops=3)
9	Index Cond: (team_id = startdate.team_id)
10	Filter: ((opponent)::text <> 'Worskla'::text)
11	-> Index Scan using "Stadium_pkey" on stadium (cost=0.15..0.17 rows=1 width=122) (actual time=0.005..0.005 rows=0 loops=3)
12	Index Cond: (stadium_id = a.stadium_id)
13	Filter: ((name)::text ~~ 'Karpaty'::text)
14	Rows Removed by Filter: 1
15	Planning Time: 0.617 ms
16	Execution Time: 16.337 ms

```
2
Enter values following this sequence len_name, team_name, diff_score:
len_name: 8
team_name: Dolor
diff_score: 2

Time of request 218  ms
Show each row and it's columns values
start_date  own_score   opp_score
2019-12-12        2          2
2018-09-12        2          2
2000-01-18        1          1
2000-07-02        2          2
2000-07-13        3          3
2000-08-30        1          1
2000-12-26        3          3
2001-04-26        2          2
```

```sql
with t2 as (select team_id from team where length(name) < 8 and name not similar to 'Dehoj')
select start_date, own_score, opp_score from startdate  as a inner join
(select match_id, opp_score, own_score from match where opp_score = own_score or opp_score - own_score = 2) as aa
 on a.match_id=aa.match_id
 where team_id in (Select team_id from t2)
```

Data Output | Explain | Messages | Notifications

	start_date date	own_score integer	opp_score integer
1	2019-12-12	2	2
2	2018-09-12	2	2
3	2000-01-18	1	1
4	2000-07-02	2	2
5	2000-07-13	3	3
6	2000-08-30	1	1
7	2000-12-26	3	3
8	2001-04-26	2	2

```sql
EXPLAIN ANALYZE
with t2 as (select team_id from team where length(name) < 8 and name not similar to 'Dolor')
select start_date from startdate  as a inner join
(select match_id, opp_score, own_score from match where opp_score = own_score or opp_score - own_score = 2) as aa
 on a.match_id=aa.match_id
 where team_id in (Select team_id from t2)
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
4	-> Seq Scan on startdate a (cost=0.00..1541.08 rows=100008 width=12) (actual time=0.038..11.042 rows=100013 loops=1)
5	-> Hash (cost=1.91..1.91 rows=1 width=4) (actual time=0.044..0.045 rows=9 loops=1)
6	Buckets: 1024 Batches: 1 Memory Usage: 9kB
7	-> Seq Scan on match (cost=0.00..1.91 rows=1 width=4) (actual time=0.019..0.037 rows=9 loops=1)
8	Filter: ((opp_score = own_score) OR ((opp_score - own_score) = 2))
9	Rows Removed by Filter: 54
10	-> Index Scan using "Team_pkey" on team (cost=0.42..0.74 rows=1 width=4) (actual time=0.005..0.005 rows=0 loops=14425)
11	Index Cond: (team_id = a.team_id)
12	Filter: (((name)::text !~ '^(?:Dolor)$'::text) AND (length((name)::text) < 8))
13	Rows Removed by Filter: 1
14	Planning Time: 0.743 ms
15	Execution Time: 115.709 ms

```
2
Enter values following this sequence len_name, team_name, diff_score:
len_name: 100
team_name: Dynamo
diff_score: 1

Time of request 198  ms
Show each row and it's columns values
start_date  own_score   opp_score
2020-09-15        2          2
2020-12-12        2          1
2019-12-12        2          2
2000-02-01        1          0
2000-01-10        4          3
2000-01-10        4          3
2000-01-10        2          1
2000-01-10        4          4
2000-01-10        2          1
2000-01-10        2          1
```

```sql
--EXPLAIN ANALYZE
with t2 as (select team_id from team where length(name) < 100 and name not similar to 'Dynamo')
select start_date, own_score, opp_score from startdate  as a inner join
(select match_id, opp_score, own_score from match where opp_score = own_score or opp_score - own_score = 1) as aa
 on a.match_id=aa.match_id
 where team_id in (Select team_id from t2)
```

Data Output | Explain | Messages | Notifications

	start_date date	own_score integer	opp_score integer
1	2020-09-15	2	2
2	2020-12-12	1	2
3	2019-12-12	2	2
4	2000-02-01	0	1
5	2000-01-10	3	4
6	2000-01-10	3	4
7	2000-01-10	1	2
8	2000-01-10	4	4
9	2000-01-10	1	2
10	2000-01-10	1	2

```
EXPLAIN ANALYZE
with t2 as (select team_id from team where length(name) < 100 and name not similar to 'Dynamo')
select start_date from startdate  as a inner join
(select match_id, opp_score, own_score from match where opp_score = own_score or opp_score - own_score = 1) as aa
 on a.match_id=aa.match_id
 where team_id in (Select team_id from t2)
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
4	-> Seq Scan on startdate a (cost=0.00..1541.08 rows=100008 width=12) (actual time=0.039..15.130 rows=100013 loops=1)
5	-> Hash (cost=1.91..1.91 rows=1 width=4) (actual time=0.051..0.052 rows=11 loops=1)
6	Buckets: 1024 Batches: 1 Memory Usage: 9kB
7	-> Seq Scan on match (cost=0.00..1.91 rows=1 width=4) (actual time=0.023..0.042 rows=11 loops=1)
8	Filter: ((opp_score = own_score) OR ((opp_score - own_score) = 1))
9	Rows Removed by Filter: 52
10	-> Index Scan using "Team_pkey" on team (cost=0.42..0.74 rows=1 width=4) (actual time=0.006..0.006 rows=1 loops=17565)
11	Index Cond: (team_id = a.team_id)
12	Filter: (((name)::text !~ '^(?:Dynamo)$'::text) AND (length((name)::text) < 100))
13	Rows Removed by Filter: 0
14	Planning Time: 0.890 ms
15	Execution Time: 174.480 ms

```
3
Enter values following this sequence team_name, date_start, date_finish, team_id:
team_name: %
date_start: 2015-01-01
date_finish: 2020-07-11
team_id: 15

Time of request 7  ms
Show each row and it's columns values
team_id  sum_team   team_name   opponent_name
2          1          Dynamo     Hover
7          1          Broslaw    Corole
```

```
select tid as team_id,  sum(cnt) sum_team, cc.n as team_name, cc.o as opponent_name from (
select team_id tid, count(team_id) cnt, n, o from startdate inner join (
Select team_id, name as n, opponent as o from team where name LIKE '%') as t
using (team_id)
where start_date between '2015-01-01' and '2020-07-11'
    group by team_id, t.n, t.o
) cc
where tid < 15
group by tid, cc.n, cc.o
order by tid
```

Data Output | Explain | Messages | Notifications

	team_id integer	sum_team numeric	team_name character varying (50)	opponent_name character varying (50)
1	2	1	Dynamo	Hover
2	7	1	Broslaw	Corole

```sql
EXPLAIN ANALYZE
select tid as team_id,  sum(cnt) sum_team, cc.n as team_name, cc.o as opponent_name from (
select team_id tid, count(team_id) cnt, n, o from startdate inner join (
    Select team_id, name as n, opponent as o from team where name LIKE '%') as t
    using (team_id)
    where start_date between '2016-01-01' and '2020-07-11'
    group by team_id, t.n, t.o
) cc
where tid < 15
group by tid, cc.n, cc.o
order by tid
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
10	Index Cond: (team_id < 15)
11	Filter: ((start_date >= '2016-01-01'::date) AND (start_date <= '2020-07-11'::date))
12	Rows Removed by Filter: 13
13	-> Index Scan using "Team_pkey" on team (cost=0.42..8.44 rows=1 width=68) (actual time=0.012..0.012 rows=1 loops=2)
14	Index Cond: (team_id = startdate.team_id)
15	Filter: ((name)::text ~~ '%'::text)
16	Planning Time: 0.679 ms
17	Execution Time: 0.222 ms

```
3
Enter values following this sequence team_name, date_start, date_finish, team_id:
team_name: Dynamo
date_start: 2010-02-02
date_finish: 2021-01-01
team_id: 100000

Time of request 2  ms
Show each row and it's columns values
team_id  sum_team   team_name   opponent_name
2          2          Dynamo     Hover
```

```sql
--EXPLAIN ANALYZE
select tid as team_id,  sum(cnt) sum_team, cc.n as team_name, cc.o as opponent_name from (
select team_id tid, count(team_id) cnt, n, o from startdate inner join (
    Select team_id, name as n, opponent as o from team where name LIKE 'Dynamo') as t
    using (team_id)
    where start_date between '2016-01-01' and '2020-07-11'
    group by team_id, t.n, t.o
) cc
where tid < 15
group by tid, cc.n, cc.o
order by tid
```

Data Output | Explain | Messages | Notifications

	team_id integer	sum_team numeric	team_name character varying (50)	opponent_name character varying (50)
1	2	1	Dynamo	Hover

```sql
EXPLAIN ANALYZE
select tid as team_id,  sum(cnt) sum_team, cc.n as team_name, cc.o as opponent_name from (
select team_id tid, count(team_id) cnt, n, o from startdate inner join (
    Select team_id, name as n, opponent as o from team where name LIKE 'Dynamo') as t
    using (team_id)
    where start_date between '2016-01-01' and '2020-07-11'
    group by team_id, t.n, t.o
) cc
where tid < 15
group by tid, cc.n, cc.o
order by tid
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
11	Filter: ((start_date >= '2016-01-01'::date) AND (start_date <= '20...
12	Rows Removed by Filter: 13
13	-> Index Scan using "Team_pkey" on team (cost=0.42..8.44 rows=...
14	Index Cond: (team_id = startdate.team_id)
15	Filter: ((name)::text ~~ 'Dynamo'::text)
16	Rows Removed by Filter: 1
17	Planning Time: 0.683 ms
18	Execution Time: 0.169 ms

Завдання 4:

master | DataBase / Database and management tools(Lab 2) / basic_backend.py / Jump to | Go to file

PointProgram Add files via upload — Latest commit c01cf76 9 minutes ago — History

1 contributor

4 lines (3 sloc) | 95 Bytes — Raw | Blame

```python
from controller import Controller

if __name__ == "__main__":
    Controller().input_get()
```

master | DataBase / Database and management tools(Lab 2) / model.py / Jump to | Go to file

PointProgram Add files via upload — Latest commit c01cf76 10 minutes ago — History

1 contributor

269 lines (248 sloc) | 11.2 KB — Raw | Blame

```python
import psycopg2
import time

class Model:
    #connection to PostgreSQL server
    def __init__(self):
        try:
            self.connection = psycopg2.connect( user = "postgres",
                                        password = "postgres",
                                        host = "127.0.0.1",
                                        port = "5432",
                                        database = "Comand_Match" )
            self.cursor = self.connection.cursor()
        except (Exception, psycopg2.Error) as error:
```

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

    #show data from tables
    def showTable(self, sql_select_query, tab, record, bool):
        try:
            self.cursor = self.connection.cursor()
            if bool:
                beg = int(time.time() * 1000)
            self.cursor.execute(sql_select_query, record)
            if bool:
                end = int(time.time() * 1000) - beg
                print("Time of request", end, " ms")
            records = self.cursor.fetchall()
            #print_table(records, tab)
        except (Exception, psycopg2.Error) as error:
            print("Error while fetching data from PostgreSQL", error)
            self.connection.rollback()
        finally:
            if self.connection:
                self.cursor.close()
            return records

    #update table row
    def updateTable(self, records, sql_update_query):
        try:
            cursor = self.connection.cursor()
            cursor.executemany(sql_update_query, records)
            self.connection.commit()
            row_count = cursor.rowcount
            print(row_count, "Records Updated")
        except (Exception, psycopg2.Error) as error :
            print("Failed updating record of the table {}", error)
            self.connection.rollback()
```

```
        except (Exception, psycopg2.Error) as error :
            print("Failed updating record of the table {}", error)
            self.connection.rollback()
        finally:
            if self.connection:
                cursor.close()

    #add table row
    def addTable(self, records, sql_insert_query):
        try:
            cursor = self.connection.cursor()
            cursor.executemany(sql_insert_query, records)
            self.connection.commit()
            print(cursor.rowcount, "Record successfully inserted")
        except (Exception, psycopg2.Error) as error :
            print("Failed inserting record into table {}".format(error))
            self.connection.rollback()
        finally:
            if self.connection:
                cursor.close()

    #delete table row
    def delTable(self, records, sql_delete_query):
        try:
            cursor = self.connection.cursor()
            cursor.executemany(sql_delete_query, records)
            self.connection.commit()
            print(cursor.rowcount, "Record deleted")
        except (Exception, psycopg2.Error) as error :
            print("Failed deleting record into table {}".format(error))
            self.connection.rollback()
        finally:
```

```
        finally:
            if self.connection:
                cursor.close()

    def only_possible(self, val, num):
        try:
            rec = [(val, )]
            sql_origin = self.origin_type(num)
            curs = self.connection.cursor()
            curs.execute(sql_origin, rec)
            records = curs.fetchall()
            if records[0] == None:
                for row in records:
                    print("There is row with id: ", row[0])
        except (Exception, psycopg2.Error) as error:
            print("Failed, there are no records with such id: {}".format(error))
            self.if_exit()
        finally:
            if self.connection:
                curs.close()

    #close connection of Postgre table
    def close_connect(self):
        if self.connection:
            self.connection.close()
            print("PostgreSQL connection is closed")

    # choose command to insert data to specific table
    def insert_type(self, table_num):
        if table_num == '1':
```

```
    def insert_type(self, table_num):
        if table_num == '1':
            sql_insert_query = """ INSERT INTO match (match_id, opp_score, own_score)
            VALUES (%s, %s, %s)"""
        elif table_num == '2':
            sql_insert_query = """ INSERT INTO stadium (stadium_id, name, city)
            VALUES (%s, %s, %s)"""
        elif table_num == '3':
            sql_insert_query = """ INSERT INTO startdate (team_id, match_id, start_date)
            VALUES (%s, %s, %s)"""
        elif table_num == '4':
            sql_insert_query = """ INSERT INTO team (team_id, name, opponent, coach, stadium_id)
            VALUES (%s, %s, %s, %s, %s)"""
        return sql_insert_query

    # choose command to update data of specific table
    def update_type(self, table_num):
        if table_num == '1':
            sql_update_query = """UPDATE match set opp_score = %s, own_score = %s
            WHERE "match_id" = %s """
        elif table_num == '2':
            sql_update_query = """UPDATE stadium set name = %s, city = %s
            WHERE "stadium_id" = %s """
        elif table_num == '3':
            sql_update_query = """UPDATE startdate set start_date = %s
            WHERE team_id = %s AND match_id = %s"""
        elif table_num == '4':
            sql_update_query = """UPDATE team set name = %s, opponent = %s, coach = %s, stadium_id = %s
              WHERE team_id = %s """
        return sql_update_query

```

```
    # choose command to delete data of specific table
    def delete_type(self, table_num):
        if table_num == '1':
            sql_delete_query = """ DELETE FROM match WHERE match_id = %s"""
        elif table_num == '2':
            sql_delete_query = """ DELETE FROM stadium WHERE stadium_id = %s"""
        elif table_num == '3':
            sql_delete_query = """ DELETE FROM startdate WHERE team_id = %s AND match_id = %s"""
        elif table_num == '4':
            sql_delete_query = """ DELETE FROM team WHERE team_id = %s"""
        return sql_delete_query

    def select_type(self, table_num):
        if table_num == '1':
            sql_select_query = """ SELECT * FROM match ORDER BY match_id"""
        elif table_num == '2':
            sql_select_query = """ SELECT * FROM stadium ORDER BY stadium_id"""
        elif table_num == '3':
            sql_select_query = """ SELECT * FROM startdate ORDER BY team_id, match_id"""
        elif table_num == '4':
            sql_select_query = """ SELECT * FROM team ORDER BY team_id"""
        return sql_select_query

    def origin_type(self, table_num):
        if table_num == 1:
            sql_origin_val = """ SELECT stadium.stadium_id FROM stadium WHERE stadium_id = %s """
        elif table_num == 2:
            sql_origin_val = """ SELECT team.team_id FROM team WHERE team_id = %s """
        elif table_num == 3:
            sql_origin_val = """ SELECT match.match_id FROM match WHERE match_id = %s """
        elif table_num == 4:
```

```
        elif table_num == 4:
            sql_origin_val = """ SELECT count(a) FROM (
                            SELECT team_id, match_id FROM startdate
                            WHERE team_id = %s AND match_id = %s) as a """
        return sql_origin_val

    def random_type(self, table_num):
        if table_num == '1':
            sql_random_query = """ INSERT INTO match (opp_score, own_score)
            SELECT
                  trunc(random()*1000)::int,
                  trunc(random()*1000)::int
            FROM
                generate_series(1, %s)
            """
        elif table_num == '2':
            sql_random_query = """ INSERT INTO stadium (name, city)
            SELECT
                    chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
                    chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
                    chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int),
                    chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
                    chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int) ||
                    chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)
            FROM
                generate_series(1, %s)
            """
        elif table_num == '3':
            sql_random_query = """ INSERT INTO startdate (team_id, match_id, start_date)
            SELECT team_id, match_id, dat FROM
            (SELECT
            team_id, match_id,
              team_id, match_id,
              date '2000-01-10' + trunc(random()*10 * (date '2020-05-20' - date '2010-01-10'))::int as dat
              FROM team, match tablesample BERNOULLI(100)
              ORDER BY random()) k ,generate_series(1, 10000) LIMIT %s
              """
        elif table_num == '4':
            sql_random_query = """ INSERT INTO team (name, opponent, coach, stadium_id)
            SELECT nam, opp, coach, stadium_id FROM
            (SELECT md5((random()*1)::text) as nam,
                md5((random()*2)::text) as opp,
                md5((random()*3)::text) as coach,
                stadium_id
            FROM stadium tablesample BERNOULLI(100)
            ORDER BY random()) k, generate_series(1, 10000) LIMIT %s
            """
        return sql_random_query

    # def random_generator():

    def specific_type(self, table_num):
        if table_num == '1':
            sql_specific_query = """ SELECT team_id, name as team_name, coach, opponent, stadium_name
            FROM team  AS a INNER JOIN
            (SELECT name AS stadium_name, stadium_id FROM stadium WHERE name LIKE %s) AS aa
            ON a.stadium_id=aa.stadium_id
            WHERE opponent != %s AND team_id in (SELECT team_id FROM startdate WHERE start_date > %s)
            """
        elif table_num == '2':
            sql_specific_query = """ WITH t_n AS (SELECT team_id FROM team WHERE length(name) < %s
            AND name NOT SIMILAR TO %s)
                sql_specific_query = """ WITH t_n AS (SELECT team_id FROM team WHERE length(name) < %s
                AND name NOT SIMILAR TO %s)
                SELECT start_date, opp_score, own_score FROM startdate  AS a INNER JOIN
                (SELECT match_id, opp_score, own_score FROM
                match WHERE opp_score = own_score OR opp_score - own_score = %s) as aa
                ON a.match_id=aa.match_id
                WHERE team_id IN (SELECT team_id FROM t_n)
                """
            elif table_num == '3':
                sql_specific_query = """ SELECT tid AS team_id,  sum(cnt) sum_team, cc.n AS team_name,
                cc.o AS opponent_name FROM (
                SELECT team_id tid, count(team_id) cnt, n, o FROM startdate INNER JOIN
                (SELECT team_id, name AS n, opponent AS o FROM team WHERE name LIKE %s) AS t USING (team_id)
                WHERE start_date BETWEEN %s AND %s
                GROUP BY team_id, t.n, t.o) cc
                WHERE tid < %s
                GROUP BY tid, cc.n, cc.o
                ORDER BY tid
                """
            return sql_specific_query

        # check if int is valid
        def Repr_init(self, n):
            try:
                int(n)
                return True
            except ValueError:
                return False

        # check if void input
        def catch_void(self, inp):
            try:
                .
```

```
    def catch_void(self, inp):
        try:
            inp
            return "True"
        except SyntaxError:
            inp = None

    def check_default(self, times):
        if times == 'd':
            times = 100000
            return times
        else:
            return times

```

master · DataBase / Database and management tools(Lab 2) / **view.py** / Jump to · Go to file

PointProgram Add files via upload — Latest commit c01cf76 11 minutes ago — History

1 contributor

36 lines (34 sloc) | 1.98 KB — Raw | Blame

```
class View:
    def print_table(self, records, tab):
        if tab == '1':
            self.print_text("Match_id   opp_score   own_score ")
            for sql_row in records:
                print("{:<10}".format(sql_row[0]), "{:<10}".format(sql_row[1]), "{:<10}".format(sql_row[2]))
        elif tab == '2':
            self.print_text("Stadium_id    Name    City ")
            for sql_row in records:
                print("{:<10}".format(sql_row[0]), "{:<10}".format(sql_row[1]), "{:<10}".format(sql_row[2]))
        elif tab == '3':
            self.print_text("Team_id    Match_id       Start_date")
            for sql_row in records:
                print("{:<10}".format(sql_row[0]), "{:<10}".format(sql_row[1]), "   ", sql_row[2])
        elif tab == '4':
            self.print_text("Team_id    Name     Oppnent      Coach      Stadium_id")
            for sql_row in records:
                print("{:<10}".format(sql_row[0]), "{:<10}".format(sql_row[1]), "{:<10}".format(sql_row[2]), "{:<15}".format(sql_row[3]), "{:<15}".format(sql_row[4]))
        elif tab == '5':
            self.print_text("team_id   team_name   coach   opponent   stadium_name ")
            for sql_row in records:
                print("{:<10}".format(sql_row[0]), "{:<10}".format(sql_row[1]), "{:<10}".format(sql_row[2]), "{:<10}".format(sql_row[3]), "{:<10}".format(sql_row[4]))
        elif tab == '6':
            self.print_text("start_date    own_score     opp_score ")
            for sql_row in records:
                print(sql_row[0], "       {:<10}".format(sql_row[1]), "{:<10}".format(sql_row[2]))
        elif tab == '7':
            self.print_text("team_id    sum_team     team_name     opponent_name")
            for sql_row in records:
                print("{:<10}".format(sql_row[0]), "{:<10}".format(sql_row[1]), "{:<10}".format(sql_row[2]), "{:<10}".format(sql_row[3]))

    def display(self, txt):
        return input(txt)

    def print_text(self, txt):
        print(str(txt))
```

410 lines (389 sloc) | 17.5 KB · Raw · Blame

```python
from model import Model
from view import View
import datetime
class Controller:
    def __init__(self):
        self.model = Model()
        self.view = View()
    # enter new data for insertion for specific table
    def insert_query(self, opt):
        times = self.quantity_put()
        self.check_times(times)
        record_to_insert = ""
        if opt == '1':
            for i in range(int(times)):
                self.print_text("Enter values following this sequence: Match_id, opp_score, own_score")
                m_id = self.view.display("Match_id: ")
                self.non_null_check(m_id, True)
                self.Repr_int(m_id)
                opp = self.view.display("opp_score: ")
                self.non_null_check(opp, True)
                self.Repr_int(opp)
                own = self.view.display("own_score: ")
                self.non_null_check(own, True)
                self.Repr_int(own)
                record_to_insert = [(m_id, opp, own)]
        elif opt == '2':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence: Stadium_id, Name, City")
                s_id = self.view.display("Stadium_id: ")
                self.non_null_check(s_id, True)
                self.Repr_int(s_id)
                n = self.view.display("Name: ")
                self.non_null_check(n, True)
                self.Repr_char(n)
                c = self.view.display("City: ")
                self.non_null_check(n, True)
                self.Repr_char(n)
                record_to_insert = [(s_id, n, c)]
        elif opt == '3':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence: Team_id, Match_id, Start_date(yyyy-mm-dd)")
                t_id = self.view.display("Team_id: ")
                self.non_null_check(t_id, True)
                self.Repr_int(t_id)
                self.model.only_possible(t_id, 2)
                m_id = self.view.display("Match_id: ")
                self.non_null_check(m_id, True)
                self.Repr_int(m_id)
                self.model.only_possible(m_id, 3)
                s_date = self.view.display("Start_date: ")
                self.non_null_check(s_date, True)
                self.non_null_check(s_date, True)
                self.Repr_date(s_date)
                record_to_insert = [(t_id, m_id, s_date)]
        elif opt == '4':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence: Team_id, Name, Opponent, Coach, Stadium_id")
                team_id = self.display("Team_id: ")
                self.non_null_check(team_id, True)
                self.Repr_int(team_id)
                name = self.view.display("Name: ")
                self.non_null_check(name, True)
                self.Repr_char(name)
                opon = self.display("Opponent: ")
                self.non_null_check(opon, True)
                self.Repr_char(opon)
                coach = self.view.display("Coach: ")
                if self.non_null_check(coach, False):
                    self.Repr_char(coach)
                st_id = self.view.display("Stadium_id: ")
                self.non_null_check(st_id, True)
                self.Repr_int(st_id)
                self.model.only_possible(st_id, 1)
                record_to_insert = [(team_id, name, opon, coach, st_id)]
        return record_to_insert

    # enter new data to update the specific table
    def update_query(self, opt):
        times = self.view.display("What number of rows do you want to update? - ")
        self.check_times(times)
        record_to_update = ""
        if opt == '1':
```

```
        if opt == '1':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence opp_score, own_score, Match_id:")
                opp = self.view.display("opp_score: ")
                self.non_null_check(opp, True)
                self.Repr_int(opp)
                own = self.view.display("own_score: ")
                self.non_null_check(own, True)
                self.Repr_int(own)
                m_id = self.view.display("Match_id: ")
                self.non_null_check(m_id, True)
                self.Repr_int(m_id)
                self.model.only_possible(m_id, 3)
                record_to_update = [(opp, own, m_id)]
        elif opt == '2':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence Name, City, Stadium_id:")
                n = self.view.display("Name: ")
                self.non_null_check(n, True)
                self.Repr_char(n)
                c = self.view.display("City: ")
                self.non_null_check(c, True)
                self.Repr_char(c)
                s_id = self.view.display("Stadium_id: ")
                self.non_null_check(s_id, True)
                self.Repr_int(s_id)
                self.model.only_possible(s_id, 1)
                record_to_update = [(n, c, s_id)]
        elif opt == '3':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence Start_date(yyyy-mm-dd), Team_id, Match_id:")
                s_date = self.view.display("Start_date: ")
                self.Repr_date(s_date)
                t_id = self.view.display("Team_id: ")
                self.non_null_check(t_id, True)
                self.Repr_int(t_id)
                self.model.only_possible(t_id, 2)
                m_id = self.view.display("Match_id: ")
                self.non_null_check(m_id, True)
                self.Repr_int(m_id)
                self.model.only_possible(m_id, 3)
                record_to_update = [(s_date, t_id, m_id)]
        elif opt == '4':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence Name, Opponent, Coach, Stadium_id, Team_id:")
                name = self.view.display("Name: ")
                self.non_null_check(name, True)
                self.Repr_char(name)
                opon = self.view.display("Opponent: ")
                self.non_null_check(opon, True)
                self.Repr_char(opon)
                coach = self.view.display("Coach: ")
                if self.non_null_check(coach, False):
                    self.Repr_char(coach)
                st_id = self.view.display("Stadium_id: ")
                self.non_null_check(st_id, True)
                self.Repr_int(st_id)
                self.model.only_possible(st_id, 1)
                team_id = self.view.display("Team_id: ")
                self.non_null_check(team_id, True)
                self.Repr_int(team_id)
                self.Repr_int(team_id)
                self.model.only_possible(team_id, 2)
                record_to_update = [(name, opon, coach, st_id, team_id)]
        return record_to_update

    def specific_query(self, opt):
        if opt == '1':
            self.view.print_text("Enter values following this sequence stadium_name, opponent, start_date:")
            sn = self.view.display("stadium_name: ")
            if self.non_null_check(sn, False):
                self.Repr_char(sn)
            op = self.view.display("opponent: ")
            if self.non_null_check(op, False):
                self.Repr_char(op)
            sd = self.view.display("start_date: ")
            self.Repr_date(sd)
            record_to_specific = (sn, op, sd)
        elif opt == '2':
            self.view.print_text("Enter values following this sequence len_name, team_name, diff_score:")
            ng = self.view.display("len_name: ")
            self.non_null_check(ng, True)
            self.Repr_int(ng)
            lt = self.view.display("team_name: ")
            if self.non_null_check(lt, False):
                self.Repr_char(lt)
            tn = self.view.display("diff_score: ")
            self.non_null_check(tn, True)
            self.Repr_int(tn)
            record_to_specific = (ng, lt, tn)
        elif opt == '3':
            self.view.print_text("Enter values following this sequence team_name, date_start, date_finish, team_id:")
```

```
            tn = self.view.display("team_name: ")
            if self.non_null_check(tn, False):
                self.Repr_char(tn)
            ds = self.view.display("date_start: ")
            self.Repr_date(ds)
            df = self.view.display("date_finish: ")
            self.Repr_date(df)
            ti = self.view.display("team_id: ")
            self.non_null_check(ti, True)
            self.Repr_int(ti)
            record_to_specific = (tn, ds, df, ti)
        return record_to_specific

    # enter data for deletion of specific table
    def delete_query(self, opt):
        times = self.view.display("What number of row do you want to delete? - ")
        self.check_times(times)
        record_to_delete = ""
        if opt == '1':
            for i in range(int(times)):
                m_id = self.view.display("Enter value that marks Match_id:")
                self.non_null_check(m_id, True)
                self.Repr_int(m_id)
                self.model.only_possible(m_id, 3)
                record_to_delete = [(m_id,)]
        elif opt == '2':
            for i in range(int(times)):
                s_id = self.view.display("Enter value that marks Stadium_id:")
                self.non_null_check(s_id, True)
                self.Repr_int(s_id)
```

```
                self.Repr_int(s_id)
                self.model.only_possible(s_id, 1)
                record_to_delete = [(s_id,)]
        elif opt == '3':
            for i in range(int(times)):
                self.view.print_text("Enter values following this sequence team_id, match_id:")
                t_id = self.view.display("team_id: ")
                self.non_null_check(t_id, True)
                self.Repr_int(t_id)
                self.model.only_possible(t_id, 2)
                m_id = self.view.display("match_id: ")
                self.non_null_check(m_id, True)
                self.Repr_int(m_id)
                self.model.only_possible(m_id, 3)
                record_to_delete = [(t_id, m_id)]
        elif opt == '4':
            for i in range(int(times)):
                team_id = self.view.display("Enter value that mark Team_id:")
                self.non_null_check(team_id, True)
                self.Repr_int(team_id)
                self.model.only_possible(team_id, 2)
                record_to_delete = [(team_id,)]
        return record_to_delete

    # main menu function
    def input_get(self):
        opt = self.input_start()
        self.check_option(opt)
        if opt == "1":
            table = self.table_choose()
            if_rand = self.view.display("Do you want to insert rows manually or randomly? \nPress M if manually, R if randomly: ")
```

```
                self.model.delTable(record, sql_delete)
                if self.to_continue() is False:
                    break
            self.if_exit()
        elif opt == "4":
            table = self.spec_choose()
            tab = str(int(table) + 4)
            record = self.specific_query(table)
            sql_spec = self.model.specific_type(table)
            show = self.model.showTable(sql_spec, tab, record, True)
            self.view.print_table(show, tab)
            self.if_exit()
        elif opt == "5":
            table = self.table_choose()
            sql_select = self.model.select_type(table)
            show = self.model.showTable(sql_select, table, None, False)
            self.view.print_table(show, table)
            self.if_exit()

    # input option
    def input_start(self):
        x = input("""Choose option: \nPRESS: 1 to add... 2 to update...  3 to delete...  4 to specific_select... 5 to show_table...\n""")
        return x

    def quantity_put(self):
        return input("What number of row do you want to add? - ")

    # choose specific table
    def table_choose(self):
        table = input(
```

```
            self.to_random(if_rand)
            if if_rand == 'M':
                while True:
                    sql_insert = self.model.insert_type(table)
                    record = self.insert_query(table)
                    self.model.addTable(record, sql_insert)
                    if self.to_continue() is False:
                        break
            elif if_rand == 'R':
                times = self.view.display("What number of row do you want to add(type 'd' to set default)? - ")
                times = self.model.check_default(times)
                self.check_times(times)
                sql_random = self.model.random_type(table)
                self.model.addTable([(times,)], sql_random)
            self.if_exit()
        elif opt == "2":
            table = self.table_choose()
            while True:
                sql_update = self.model.update_type(table)
                record = self.update_query(table)
                self.model.updateTable(record, sql_update)
                if self.to_continue() is False:
                    break
            self.if_exit()

        elif opt == "3":
            table = self.table_choose()
            while True:
                sql_delete = self.model.delete_type(table)
                record = self.delete_query(table)
```

```
            "Choose table: \n press 1 - Match...   press 2 - Stadium...   press 3 - StartDate... press 4 - Team\n")
        self.check_table(table, True)
        return table

    def spec_choose(self):
        print("""1) Show all info about team(name, coach, opponent), which played game on specific 'stadium', but
                    not with that 'opponent', when matches took place later than that 'date'.
                    """)

        print("""2) Show date, opp_score, own_score from those teams that have a draw or the difference between
                    home and away teams in 'number of goal' and the length of home team is less then 'that number',
                    except of 'such team'.
                    """)

        print("""3) Show team_id, sum_team, team_name, opponent_name from those teams that have 'specific name',
                    where match took place between 'date_1' and 'date_2' and team_id is less than 'that number'
                    """)
        table = input(
            "Choose table: \nPress: 1 or 2 or 3\n")
        self.check_table(table, False)
        return table

    def to_random(self, opt):
        if opt != 'M' and opt != 'R':
            print("You entered wrong value")
            self.if_exit()

    def non_null_check(self, var_t, boool):
        if var_t is None or var_t == '' and boool:
            print("Error that column cannot contain NULL values!")
```

```
            self.if_exit()
        elif var_t is None or var_t == '' and boool is False:
            return False
        return True

    # check option validation
    def check_option(self, opti):
        if opti != '1' and opti != '2' and opti != '3' and opti != '4' and opti != '5' or self.model.catch_void(opti) != "True":
            print("You entered wrong character!")
            self.if_exit()

    # check table validation
    def check_table(self, tab, bl):
        if bl:
            if tab != '1' and tab != '2' and tab != '3' and tab != '4' or self.model.catch_void(tab) != "True":
                print("You entered wrong character!")
                self.if_exit()
        if bl is False:
            if tab != '1' and tab != '2' and tab != '3' or self.model.catch_void(tab) != "True":
                print("You entered wrong character!")
                self.if_exit()

    # ask if exit
    def if_exit(self):
        ext = input("Do you want to exit? Press M to go to the main menu, or E to exit: ")
        if ext == 'M':
            self.input_get()
        elif ext == 'E':
            self.model.close_connect()
            exit()
```

```
         else:
             print("You entered wrong character...")
             self.if_exit()

     def Repr_date(self, date):
         year = ""
         month = ""
         day = ""
         if len(date) != 10 or date[4] != '-' and date[7] != '-':
             print("You entered wrong date value, please try again!")
             self.if_exit()
         for i in (0, 1, 2, 3):
             year += date[i]
         for i in (5, 6):
             month += date[i]
         for i in (8, 9):
             day += date[i]
         try:
             d = datetime.date(int(year), int(month), int(day))
         except ValueError:
             print("You entered wrong date value, please try again!")
             self.if_exit()

     def Repr_int(self, n):
         try:
             int(n)
         except ValueError:
             print("You entered wrong instance, please try again!")
             self.if_exit()
```

```
    def Repr_int(self, n):
        try:
            int(n)
        except ValueError:
            print("You entered wrong instance, please try again!")
            self.if_exit()

    def Repr_char(self, ch):
        if len(ch) > 50:
            print("You entered wrong value only 50 symbols allowed, please try again!")
            self.if_exit()
        try:
            str(ch)
        except ValueError:
            print("You entered wrong value, please try again!")
            self.if_exit()

    # check if number for loop is valid
    def check_times(self, times):
        if self.model.Repr_init(times):
            return True
        else:
            print("You entered wrong instance, please try again!")
            self.if_exit()
            return False

    # ask if continue
    def to_continue(self):
        cont = input("Do you want to continue? Press Y if yes, N if no: ")
        if cont == 'Y':
            return True
        elif cont == 'N':
```