



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №2

з дисципліни “Бази даних”

тема “Створення додатку бази даних, орієнтованого на взаємодію з СУБД
PostgreSQL”

Виконав
студент II курсу
групи КП-93
Варіант 9
Зверєв Костянтин Васильович

Перевірів
“--” “вересня” 2020р.
викладач
Петрашенко Андрій Васильович

Мета роботи

Здобутти вмінь програмування прикладних додатків баз даних PostgreSQL.

Постановка 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).

Варіант: База даних Університет.

Посилання на Git репозиторій: <https://github.com/zver-came/lab2>

Посилання на bitbucket репозиторій:

<https://bitbucket.org/zverevfpm/databases/src/master/labs/lab2/>

Завдання 1

Приклад обробки помилок при введенні даних:

1. Find student
2. Add new student
3. Delete student
4. Update student
5. Get all student subjects
6. Get all student teachers
7. Add student teacher
8. Delete student teacher
9. Work with student phone numbers
10. Work with group menu
11. Work with teachers menu
12. Open main menu

Enter command: **2**

Enter student params:

Name: **Kostya**

Surname: **Zverev**

Group params

-> name

-> id

-> skip

Enter params: **name**

Enter group name: **KP**

Group params

-> name

-> id

-> skip

Enter params: **skip**

(2182, 'KP-80')

(3274, 'KP-43')

(3813, 'KP-98')

(3874, 'KP-93')

(4807, 'KP-35')

(5117, 'KP-61')

Enter id: **1231**

Creation of a new entity (student) is interrupted.

DETAIL: group_id -> (1231) is not present in table "groups"

При видаленні даних з бази даних (відповідних таблиць) помилок не було виявлено, оскільки при видаленні сутності за неіснуючим ідентифікатором помилка не виникає. Крім того, програма контролює зовнішні зв'язки (батьківські та підлеглі таблиці) тому при видаленні сутності з батьківської таблиці, видаляються записи і з підлеглих таблиць з відповідними зовнішніми ключами.

Приклад валідації даних при введенні:

```
1. Find student
2. Add new student
3. Delete student
4. Update student
5. Get all student subjects
6. Get all student teachers
7. Add student teacher
8. Delete student teacher
9. Work with student phone numbers
10. Work with group menu
11. Work with teachers menu
12. Open main menu
```

Enter command:2

Enter student params:

Name:

Name is empty

Name: Kostya

Surname:

Surname is empty

Surname: Zvere

Group params

-> name

-> id

-> skip

Enter params:name

Enter group name: KP

(2182, 'KP-80')

(3274, 'KP-43')

(3813, 'KP-98')

(3874, 'KP-93')

(4807, 'KP-35')

(5117, 'KP-61')

Enter id:

Don't enter id:

Enter id:sw2

Don't enter id: sw2

Enter id:2182

Student successfully added with id ->176523

Завдання 2

Ілюстрації зі згенерованими даними таблиць та відповідними SQL запитами:

```
1 select * from students
```

Результат План выполнения Сообщения

	student_id [PK] integer	name character varying (50)	surname character varying (50)	group_id integer
1	50516	TTA	TMOXCAFLU	1753
2	50517	CPSQXOSI	VJUE	2157
3	50518	BGXP	OPVFGSJ	1794
4	50519	FPDOWGUQQ	EMGHEOTSH	1709
5	50520	ILYUVX	OTETXIAI	2006
6	50521	LAH KC	AIONEQO	1868
7	50522	AHOR	FVYFI	1912
8	50523	DJGT Y	FLQHNWUXO	1874
9	50524	DAJAQX	KNHU	1813
10	50525	QRODYQQSM	SDMOPMA	2101
11	50526	RVBIOEAN	NBLGAO	1804
12	50527	IXARDSSAK	PAFDGFJE	1826
13	50528	UHAGQKHI	SALMAY	1787
14	50529	VBACYKCIM	WYKCHNNWQ	2149
15	50530	IKIHRJ	SJHL	1769
16	50531	BJDQ	PNTHLP	2119
17	50532	JYCRF	UWOYNLLV	2019
18	50533	INNIRU	UYMWV	2057

```

1 SELECT * FROM public.student_phone
2 ORDER BY phone_number ASC, student_id ASC

```

Результат План выполнения Сообщения

	phone_number [PK] character varying (50)	student_id [PK] integer
1	+3231212312	116000
2	+380432342	116000
3	+380900000164	105014
4	+380900002377	61217
5	+380900005438	99933
6	+380900005899	86484
7	+380900006065	86530
8	+380900006091	112189
9	+380900007075	150461
10	+380900007974	122451
11	+380900008102	147802
12	+380900010391	162281
13	+380900010696	149473
14	+380900011327	60353
15	+380900012515	125955
16	+380900014013	153300
17	+380900014653	169211
18	+380900015038	168791

Query Editor История запросов

```

1 with info as (insert into students (name, surname, group_id)
2                 select random_str(3+(random() * 7)::int), random_str(3 + (random() * 7)::int),random_group_id()
3                 from generate_series(1, 10000) returning student_id)
4
5 insert into student_phone(phone_number, student_id) select random_phone(), student_id from info

```

{random_phone()}

Общие Определение Код Параметры Безопасность SQL

```

1 select '+380' || (select trunc(900000000+random() * 99999999+1)::int)

```

{random_str(integer)}

Общие Определение Код Параметры Безопасность SQL

```

1 select array_to_string(array(select chr(trunc(65 + random() * 25)::int)
2 FROM generate_series(1,$1)), '')

```

{random_group_id()}

Общие Определение Код Параметры Безопасность SQL

```

1 select group_id from groups order by random() limit 1




```

```

1 SELECT * FROM public.teachers
2 ORDER BY teacher_id ASC

```

Результат План выполнения Сообщения

	 teacher_id [PK] integer 	name character varying (50) 	surname character varying (50) 	subject_id integer 
1	3604	NKKSSV	DVYDTTM	128
2	3605	JVQLDSJBL	SPLRW	171
3	3606	EBYWHVG	LEC	180
4	3607	WVKWFT	FRKBTk	126
5	3608	IXYQYCM	HVYXLHRO	154
6	3609	KESE	JDKXRBVL	161
7	3610	LUO	OVGWNEF	131
8	3611	LBNAJ	OTY	154
9	3612	VLCIW	JWS	135
10	3613	WVEGDP	HNDV	148
11	3614	THHXA	TSMFWF	179
12	3615	UKFGA	NKKYB	161
13	3616	ECN	IYANO	157
14	3617	WCUSCEX	VNCOCB	161
15	3618	IINFBD	IHKMKB	164
16	3619	LQBSHN	KXALU	129
17	3620	DBFFJ	PTQHLLF	157

```

1 SELECT * FROM public.teacher_email
2 ORDER BY email ASC, teacher_id ASC

```

Результат План выполнения Сообщения

	email [PK] character varying (50)	teacher_id [PK] integer
1	aasyeif@gmail.com	4639
2	acepgog@gmail.com	4348
3	adblxlg@gmail.com	5024
4	adevabd@gmail.com	4265
5	aedmxlw@gmail.com	4159
6	afwltcc@gmail.com	5124
7	afybtwj@gmail.com	5142
8	agpmqww@gmail.com	4124
9	ahgkbyu@gmail.com	4538
10	ahphdbl@gmail.com	5112
11	ahwigca@gmail.com	4256
12	aikbghg@gmail.com	4729
13	ajnelwz@gmail.com	4761
14	ajobiim@gmail.com	4085
15	amexsld@gmail.com	3982
16	amfbsix@gmail.com	4711
17	amunrkf@gmail.com	4576
18	anebipc@gmail.com	5140

Query Editor История запросов

```

1 with info as (insert into teachers (name,surname,subject_id)
2               select random_str(3+(random()*7)::int), random_str(3+(random()*7)::int),random_subject_id()
3               from generate_series(1,100000) returning teacher_id)
4
5 insert into teacher_email (email,teacher_id) select random_email(7), teacher_id from info

```

{random_email(integer)}

Общие Определение Код Параметры Безопасность SQL

```

1 select array_to_string(array(SELECT chr((97 + round(random() * 25)):: integer)
2                             from generate_series(1,$1)),')|| '@gmail.com'

```

{random_subject_id()}

Общие Определение Код Параметры Безопасность SQL

```

1 select subject_id from subjects order by random() limit 1

```



```

1 SELECT * FROM public.teacher_student
2 ORDER BY teacher_id ASC, student_id ASC

```

Результат План выполнения Сообщения

	teacher_id [PK] integer	student_id [PK] integer
1	3604	50992
2	3604	51151
3	3604	52284
4	3604	55088
5	3604	55743
6	3604	56092
7	3604	56320
8	3604	56710
9	3604	57638
10	3604	59489
11	3604	60008
12	3604	62008
13	3604	62859
14	3604	63997
15	3604	64119
16	3604	64254
17	3604	64451
18	3604	65402

Query Editor История запросов

```

1 INSERT INTO teacher_student(student_id, teacher_id)
2   SELECT link.student_id, link.teacher_id FROM
3   (select student_id, random_teacher_id() as teacher_id from students, generate_series(1,2)) as link
4   left join teacher_student as ts on
5   link.student_id=ts.student_id AND ts.teacher_id=link.teacher_id WHERE ts.student_id IS NULL |
6   GROUP BY (link.student_id, link.teacher_id)

```

{random_student_id()} ✕

Общие Определение Код Параметры Безопасность SQL

```

1 select student_id from students order by random() limit 1
2

```

{random_teacher_id()} ✕

Общие Определение Код Параметры Безопасность SQL

```

1 select teacher_id from teachers order by random() limit 1

```

Завдання 3

Знайти всіх вчителів студента з певними параметрами:
(name = '%W%', surname = '%T%', Email = '%f%', id = '%6%')

```
1. Find student
2. Add new student
3. Delete student
4. Update student
5. Get all student subjects
6. Get all student teachers
7. Add student teacher
8. Delete student teacher
9. Work with student phone numbers
10. Work with group menu
11. Work with teachers menu
12. Open main menu
-----
Enter command: 6
-----
1. find student
2. enter student id
3. exit
Enter command: 2
Enter id: 166000
Enter y to set teacher parameters: y
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params: name
Enter name: W
```

```
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params: surname
Enter surname: T
-----
```

```
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params: email
Enter email: f
-----
```

```
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params: id
Enter teacher id: 6
-----
```

```
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params: skip
-----
(4261, 'WFPEUYU', 'EIEDJBT', 135)
-----
request execution time: 0.0019876956939697266
-----
1. find student
2. enter student id
3. exit
Enter command: |
```

Query Editor

История запросов

```
1 select t.teacher_id, t.name,t.surname,t.subject_id from teachers as t, students as s, teacher_student as ts
2 natural join (select t.teacher_id,t.name,t.surname,te.email from teachers as t join teacher_email as te using(teacher_id)
3               where t.name like '%W%' and t.surname like '%T%' and te.email like '%f%'
4               and cast(t.teacher_id as varchar(20)) like '%6%') as l
5 where s.student_id = 166000 and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by t.teacher_id
6
```

Результат

План выполнения

Сообщения

	teacher_id [PK] integer	name character varying (50)	surname character varying (50)	subject_id integer	
1	4442	PWDY	GSVJ	168	

Знайти всіх студентів вчителя з певними параметрами:
(name = '%L%', surname = '%A%', id = '%6%')

```

1. Find teacher
2. Add new teacher
3. Delete teacher
4. Update teacher
5. Work with teacher email
6. Get all teacher students
7. Work with subject menu
8. Work with student menu
9. Open main menu
-----
Enter command: 4
-----
1. find teacher
2. enter teacher id
3. exit
Enter command: 2
Enter id: 4550
Enter y to set student parameters: y
Student params:
-> name
-> surname
-> id
-> phone
-> group
-> skip
Enter params: name
Enter name: L
Student params:
-> name
-> surname
-> id
-> phone
-> group
-> skip
Enter params: surname
Enter surname: A
Student params:
-> name
-> surname
-> id
-> phone
-> group
-> skip
Enter params: id
Enter student id: 6
Student params:

```

```

Student params:
-> name
-> surname
-> id
-> phone
-> group
-> skip
Enter params: skip
-----
(57469, 'NFNLTIGQL', 'RAKTQAH')
(61232, 'CUEBLCRI', 'HAWIBWK')
(64080, 'AEDUUARNL', 'CGUNYA')
(86978, 'WHPRKCLND', 'DSDJKALCE')
(134156, 'OLGWDGHHOM', 'FCFGARHN')
(136704, 'LUEN', 'BEDIAWWG')
(137610, 'KOQBJNLLGI', 'NMAEGITX')
(138963, 'MJKL', 'ANMF')
(176174, 'DEJLQMRGSJ', 'QXONACAM')
(176354, 'LCRLGX', 'AOWBPINOY')
-----
request execution time: 0.022937774658203125
-----
1. find teacher
2. enter teacher id
3. exit
Enter command: |

```

Query Editor

История запросов

```

1  select s.student_id, s.name,s.surname from teachers as t, students as s, teacher_student as ts
2  natural join (select s.student_id,s.name,s.surname from students as s where s.name like '%L%'
3                  and s.surname like '%A%' and cast(s.student_id as varchar(20)) like '%6%') as l
4  where t.teacher_id = 4550 and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by s.student_id
5

```

Результат

План выполнения

Сообщения

	student_id [PK] integer	name character varying (50)	surname character varying (50)	
1	57469	NFNLTIGQL	RAKTQAH	
2	61232	CUYEBCRI	HAWIBWK	
3	64080	AEDUJARNL	CGUNYA	
4	86978	WHPRKCLND	DSDJKALCE	
5	134156	OLGWDGHHOM	FCFGARHN	
6	136704	LUEN	BEDIAWWG	
7	137610	KOQBJNLLGI	NMAEGITX	
8	138963	MJKL	ANMF	
9	176174	DEJLQMRGSJ	QXONACAM	
10	176354	LCRLGX	AOWBPINOY	

Знайти всіх вчителів предмета з певними параметрами:
(name = '%H%', surname = '%P%', Email = '%f%', id = '%9%')

```

1. Add new subject
2. Delete subject by id
3. Update subject
4. Get all subject teachers
5. Open teacher menu
6. Open main menu
-----
Enter command:4
-----
Subject params
-> name
-> id
-> skip
Enter params:name
Enter subject name: F
-----
Subject params
-> name
-> id
-> skip
Enter params:skip
-----
(126, 'CGJFFOQ')
(128, 'BBLYRSF')
(132, 'ICOUFHK')
(138, 'UFOYKVJ')
(139, 'UFFCQLT')
(140, 'FQKGF EW')
(142, 'EFSUACS')
(146, 'EKVFFDW')
(156, 'FJFEHSA')
(164, 'OSRQOQF')

```

```

(178, 'FLFBHIF')
-----
Enter id:172
Enter y to set teacher parameters: y
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:id
Enter teacher id: 9
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:name
Enter name: H
-----
Teacher params
-> name
-> surname
-> id
-> email
-> skip
Enter params:surname
Enter surname: P
-----
Teacher params
-> name
-> surname
-> id
-> email
-> subject
-> skip
Enter params:skip
-----
(4969, 'DWHGHKX', 'USIKPBVCY')
-----
request execution time: 0.004985332489013672
-----

```

```

1
2 select t.teacher_id,t.name,t.surname from subjects as sub join teachers as t using(subject_id)
3 join (select t.teacher_id,t.name,t.surname from teachers as t where cast(t.teacher_id as varchar(20))
4 like '%9%' and t.name like '%H%' and t.surname like '%P%') as l using(teacher_id)
5 where sub.subject_id=172
6

```

Результат План выполнения Сообщения

	teacher_id [PK] integer	name character varying (50)	surname character varying (50)	
1	4969	DWNGHKX	USIKPBVCY	

Завдання 4

Model:

```

1 import psycopg2
2
3 class Model:
4     def __init__(self):
5         self.cursor=None
6         self.connection=None
7         try:
8             self.connection=psycopg2.connect(user="postgres",password="postgres",host="127.0.0.1",port="5432",database="database_lab1")
9             self.cursor=self.connection.cursor()
10        except(Exception, psycopg2.Error) as error:print("Error connection with PostgreSQL",error)
11
12    def __del__(self):
13        if self.connection:
14            self.cursor.close()
15            self.connection.close()
16            print("Connection closed")
17
18    def delete_item(self, item_id, table, item_point):
19        self.cursor.execute("DELETE FROM %s WHERE %s= %s" % (table, item_point, item_id))
20        self.connection.commit()
21
22    def select_items(self, table):
23        self.cursor.execute("select * FROM %s" % (table))
24        self.connection.commit()
25        return self.cursor.fetchall()
26
27    def select_item_by_id(self, item_id, table, item_point):
28        self.cursor.execute("select * FROM %s WHERE %s= %s" % (table, item_point, item_id))
29        self.connection.commit()
30        return self.cursor.fetchall()

```

```

32 #group
33 def add_group(self,new_group):
34     self.cursor.execute("INSERT INTO groups(name) VALUES ('%s') RETURNING group_id"% (new_group))
35     self.connection.commit()
36     return self.cursor.fetchall()[0][0]
37
38 def update_group(self,upp_group):
39     self.cursor.execute("update groups set name = '%s' where group_id=%s" % (upp_group['name'], upp_group['id']))
40     self.connection.commit()
41
42 #subject
43 def add_subject(self,name):
44     self.cursor.execute("INSERT INTO subjects (name) VALUES ('%s') returning subject_id "% (name))
45     self.connection.commit()
46     return self.cursor.fetchall()[0][0]
47
48 def update_subject(self, id, name):
49     self.cursor.execute("update subjects set name = '%s' where subject_id = %s" %(name,id))
50     self.connection.commit()
51
52 #student
53 def add_student(self,new_student):
54     self.cursor.execute("INSERT INTO students (name, surname, \"group_id\" ) VALUES ('%s','%s',%s) returning student_id "
55                        % (new_student['name'], new_student['surname'],new_student['group']))
56     self.connection.commit()
57     return self.cursor.fetchall()[0][0]
58
59 def update_student(self,upp_student):
60     self.cursor.execute("update students set name = '%s',surname = '%s','group_id\"=%s where student_id = %s" %
61                        (upp_student['name'], upp_student['surname'], upp_student['group'], upp_student['id']))
62     self.connection.commit()
63
64 #student phone
65 def add_student_phone(self, new_phone, item_id):
66     self.cursor.execute("insert into student_phone (phone_number, student_id) values ('%s',%s)"%(new_phone,item_id))
67     self.connection.commit()
68
69 def delete_some_student_phone(self, item_id,phone):
70     self.cursor.execute("delete from student_phone where student_id = %s and phone_number = '%s'" % (item_id,phone))
71     self.connection.commit()
72
73 def update_students_phones(self,student_id,old_phone,new_phone):
74     self.cursor.execute("update student_phone set phone_number = '%s' where student_id = %s and phone_number = '%s'"
75                        (new_phone, student_id,old_phone))
76     self.connection.commit()
77
78 #teacher
79 def add_teacher(self,new_teacher):
80     self.cursor.execute("INSERT INTO teachers(name, surname, subject_id) VALUES ('%s','%s',%s) RETURNING teacher_id"
81                        % (new_teacher['name'], new_teacher['surname'], new_teacher['subject']))
82     self.connection.commit()
83     return self.cursor.fetchall()[0][0]
84
85 def update_teacher(self,upp_teacher):
86     self.cursor.execute("update teachers set name = '%s', surname = '%s', subject_id = %s where teacher_id=%s" %
87                        (upp_teacher['name'], upp_teacher['surname'],upp_teacher['subject'], upp_teacher['id']))
88     self.connection.commit()
89
90 #teacher email
91 def add_teacher_email(self,new_email,item_id):
92     self.cursor.execute("insert into teacher_email (email, teacher_id) values ('%s',%s)"%(new_email,item_id))
93     self.connection.commit()
94
95 def delete_some_teacher_email(self, item_id,email):
96     self.cursor.execute("delete from teacher_email as t where t.teacher_id = %s and t.email = '%s'" % ( item_id,email))
97     self.connection.commit()

```

```

99     def update_teacher_email(self,teacher_id,old_email,new_email):
100         self.cursor.execute("update teacher_email set email = '%s' where teacher_id = %s and email = '%s'"%(new_email, teacher_id,old_email))
101         self.connection.commit()
102
103     #teacher student link
104     def get_teacher_student_link(self,teacher_id,student_id):
105         self.cursor.execute("select from teacher_student where teacher_id = %s and student_id = %s"%(teacher_id, student_id))
106         return self.cursor.fetchall()
107
108     def add_teacher_student_link(self, teacher_id,student_id):
109         self.cursor.execute("insert into teacher_student (teacher_id, student_id) values (%s,%s)"%(teacher_id, student_id))
110         self.connection.commit()
111
112     def delete_teacher_student_link(self, teacher_id,student_id):
113         self.cursor.execute("delete from teacher_student where teacher_id = %s and student_id =%s"%(teacher_id,student_id))
114         self.connection.commit()
115
116     #some function
117     def get_all_student_teachers(self,student_id,query):
118         self.cursor.execute("select t.teacher_id, t.name,t.surname,t.subject_id from teachers as t, students as s, teacher_student as ts %s"
119                             "where s.student_id = %s and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by t.teacher_id" % (query, student_id))
120         return self.cursor.fetchall()
121
122     def get_all_teacher_students(self,teacher_id,query):
123         self.cursor.execute("select s.student_id, s.name,s.surname from teachers as t, students as s, teacher_student as ts %s "
124                             "where t.teacher_id = %s and s.student_id=ts.student_id and t.teacher_id=ts.teacher_id order by s.student_id"%(query,teacher_id))
125         return self.cursor.fetchall()
126
127     def get_all_student_subjects(self,student_id,query,params):
128         self.cursor.execute("select sub.subject_id, sub.name from teachers as t,subjects as sub, students as st, teacher_student as ts "
129                             "%s where t.teacher_id=ts.teacher_id and st.student_id = ts.student_id and st.student_id=%s "
130                             "and t.subject_id=sub.subject_id %s group by sub.subject_id order by sub.subject_id" % (query, student_id,params))
131         return self.cursor.fetchall()
132
133     def get_all_subject_teachers(self,subject_id, query):
134         self.cursor.execute(
135             "select t.teacher_id,t.name,t.surname from subjects as sub join teachers as t using(subject_id) %s"
136             " where sub.subject_id=%s " % (query, subject_id))
137         return self.cursor.fetchall()
138
139     #gen
140     def gen_teachers(self,number):
141         self.cursor.execute("with info as (insert into teachers (name,surname,subject_id)"
142                             "select random_str(3+(random()*7)::int), random_str(3+(random()*7)::int),random_subject_id()"
143                             "from generate_series(1,%s) returning teacher_id)"
144                             "insert into teacher_email (email,teacher_id)"
145                             "select random_email(7), teacher_id from info" % (number))
146         self.connection.commit()
147
148     def gen_teacher_student_link(self):
149         self.cursor.execute("INSERT INTO teacher_student(student_id, teacher_id) "
150                             "SELECT link.student_id, link.teacher_id FROM "
151                             "(select student_id,random_teacher_id() as teacher_id from students,generate_series(1,2)) as link "
152                             "left join teacher_student as ts on "
153                             "link.student_id=ts.student_id AND ts.teacher_id=link.teacher_id WHERE ts.student_id IS NULL "
154                             "GROUP BY (link.student_id, link.teacher_id)")
155         self.connection.commit()
156
157     def gen_students(self,number):
158         self.cursor.execute("with info as (insert into students (name, surname, group_id)"
159                             "select random_str(3+(random() * 7)::int), random_str(3 + (random() * 7)::int),random_group_id()"
160                             "from generate_series(1, %s) returning student_id)"
161                             "insert into student_phone(phone_number, student_id)"
162                             "select random_phone(), student_id from info"%(number))
163         self.connection.commit()
164
165     def gen_group(self,number):
166         self.cursor.execute("insert into groups (name) (select chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)"
167                             "|| chr(45) ||trunc(random() * 99)::int from generate_series(1,%s))"%(number))
168         self.connection.commit()
169
170     def gen_subject(self,number):
171         self.cursor.execute("insert into subjects (name) select random_str(7) from generate_series(1,%s)" % (number))
172         self.connection.commit()
173
174     def find(self,query):
175         self.cursor.execute(query)
176         return self.cursor.fetchall()

```

View:

```

1 from model import Model
2 from time import time
3 database=Model()
4
5 class View:
6     def print_item(self,items):
7         print("-----")
8         for item in items:
9             print(item)
10        print("-----")
11
12    #student menu
13    def print_student(self,student_id):
14        student =database.select_item_by_id(student_id,"students","student_id")[0]
15        print("ID: %s\t\tFullName: %s %s\tGroup: %s"%(student[0],student[1],student[2],database.select_item_by_id(student[3],"groups","group_id")[0][1]))
16        self.print_item(database.select_item_by_id(student_id,"student_phone","student_id"))
17
18    def print_teacher(self, teacher_id):
19        student = database.select_item_by_id(teacher_id, "teachers", "teacher_id")[0]
20        print("ID: %s\t\tFullName: %s %s\tSubject: %s" % (
21            student[0], student[1], student[2], database.select_item_by_id(student[3], "subjects", "subject_id")[0][1]))
22        self.print_item(database.select_item_by_id(teacher_id, "teacher_email", "teacher_id"))
23
24    def add_new_student(self, new_student):print("Student successfully added with id ->%s"% database.add_student(new_student))
25
26    def delete_student_by_id(self,student):
27        database.delete_item(student, "student_phone", "student_id")
28        database.delete_item(student, "teacher_student", "student_id")
29        database.delete_item(student,"students","student_id")
30        print("Student with id = (%s) successfully deleted"%student)
31
32    def update_student(self, upp_student):
33        database.update_student(upp_student)
34        print("Student with id ->%s successfully updated" % upp_student['id'])
35
36    def get_all_student_subject(self, student_id,query):
37        start_time = time()
38        if len(query) != 0:subjects = database.get_all_student_subjects(student_id, "natural join (" + query + ") as l "," and l.subject_id=sub.subject_id ")
39        else:subjects = database.get_all_student_subjects(student_id, query,"")
40        final_time=time() - start_time
41        self.print_item(subjects)
42        print("request execution time: ",final_time)
43
44    def get_all_student_teacher(self,student_id,query):
45        start_time = time()
46        if len(query) != 0:teachers = database.get_all_student_teachers(student_id, "natural join (" + query + ") as l ")
47        else:teachers = database.get_all_student_teachers(student_id,query)
48        final_time=time() - start_time
49        self.print_item(teachers)
50        print("request execution time: ",final_time)
51
52    #teacher menu
53    def add_new_teacher(self, new_teacher):print("Teacher %s successfully added " % (database.add_teacher(new_teacher)))
54
55    def delete_teacher_by_id(self,teacher_id):
56        database.delete_item(teacher_id, "teacher_email", "teacher_id")
57        database.delete_item(teacher_id, "teacher_student", "teacher_id")
58        database.delete_item(teacher_id, "teachers", "teacher_id")
59        print("Teacher %s successfully deleted " % (teacher_id))
60
61    def update_tracher(self, upp_teacher):
62        database.update_teacher(upp_teacher)
63        print("Teacher %s successfully updated " % (upp_teacher['name']))

```



```

65 def get_all_teacher_students(self,teacher_id,query):
66     start_time = time()
67     if len(query) != 0: students = database.get_all_teacher_students(teacher_id, "natural join (" + query + ") as 1 ")
68     else: students = database.get_all_teacher_students(teacher_id, query)
69     final_time = time() - start_time
70     self.print_item(students)
71     print("request execution time: ", final_time)
72
73 # teacher email menu
74 def add_new_email(self,teacher_id,email):
75     database.add_teacher_email(email, teacher_id)
76     print("Email: %s successfully added" % email)
77
78 def delete_email(self,teacher_id,email):
79     database.delete_some_teacher_email(teacher_id, email)
80     print("Email: %s successfully deleted" % email)
81
82 def update_email(self,teacher_id,old_email, new_email):
83     database.update_teacher_email(teacher_id, old_email, new_email)
84     print("Email successfully updated (%s)->(%s) " % (old_email, new_email))
85
86 # teacher subject menu
87 def add_new_subject(self,number):print("Subject successfully added with id-> %s"%database.add_subject(number))
88
89 def delete_subject_by_id(self,subject_id):
90     teachers = database.select_item_by_id(subject_id,"teachers","subject_id")
91     for teacher in teachers:
92         database.delete_item(teacher[0],"teacher_email","teacher_id")
93         database.delete_item(teacher[0],"teacher_student","teacher_id")
94         database.delete_item(teacher[0],"teachers","teacher_id")
95     database.delete_item(subject_id,"subjects","subject_id")
96     print("Subject with id: %s successfully deleted" % subject_id)
97
98 def update_subject(self,subject_id, new_name):
99     database.update_subject(subject_id,new_name)
100     print("Subject successfully updated ->(%s) " % (new_name))

```

```

102 # student phone menu
103 def add_new_phone(self, phone, student_id):
104     database.add_student_phone(phone, student_id)
105     print("Phone successfully added")
106
107 def delete_phone(self, student_id, phone):
108     database.delete_some_student_phone(student_id, phone)
109     print("Phone %s successfully deleted" % phone)
110
111 def update_phone(self, student_id, old_number, new_number):
112     database.update_students_phones(student_id, old_number, new_number)
113     print("Phone successfully updated (%s)->(%s) " %(old_number, new_number))
114
115 #group menu
116 def add_new_group(self, new_group): print("Group successfully added with id -> %s" % database.add_group(new_group))
117
118 def delete_group_by_id(self, group_id):
119     students = database.select_item_by_id(group_id, "students", "group_id")
120     for student in students:
121         database.delete_item(student[0], "student_phone", "student_id")
122         database.delete_item(student[0], "teacher_student", "student_id")
123         database.delete_item(student[0], "students", "student_id")
124     database.delete_item(group_id, "groups", "group_id")
125     print("Group with id: %s successfully deleted" % group_id)
126
127 def update_group(self, group):
128     database.update_group(group)
129     print("Group (%s) successfully updated"%group['name'])
130
131 #st link
132 def add_new_student_teacher_link(self, teacher_id, student_id):
133     try: database.add_teacher_student_link(teacher_id, student_id)
134     except: print("Teacher (%s) Student (%s) link successfully added" % teacher_id, student_id)
135
136 def delete_student_teacher_link(self, teacher_id, student_id):
137     try: database.delete_teacher_student_link(teacher_id, student_id)
138     except: print("Teacher (%s) Student (%s) link successfully deleted" % teacher_id, student_id)
139
140 #gen menu
141 def gen_teacher(self, number):
142     start_time = time()
143     database.gen_subject(int(number / 30) + 1)
144     database.gen_teachers(number)
145     final_time = time() - start_time
146     print("request execution time: ", final_time, "\n%s teachers successfully added" % number)
147
148 def gen_student(self, number):
149     start_time = time()
150     database.gen_group(int(number / 30) + 1)
151     database.gen_students(number)
152     final_time = time() - start_time
153     print("request execution time: ", final_time, "\n%s student successfully added" % number)
154
155 def gen_link(self):
156     start_time = time()
157     database.gen_teacher_student_link()
158     final_time = time() - start_time
159     print("request execution time: ", final_time, "\nteacher-student links successfully added")
160
161 def get_all_subject_teachers(self, subject_id, query):
162     start_time = time()
163     if len(query) != 0: teachers = database.get_all_subject_teachers(subject_id, "join (" + query + ") as 1 using(teacher_id)")
164     else: teachers = database.get_all_subject_teachers(subject_id, query)
165     final_time = time() - start_time
166     self.print_item(teachers)
167     print("request execution time: ", final_time)

```

Controller:

```
1  from view import View
2  from item_search import Search
3  from model import Model
4  from time import time
5  database=Model()
6  search=Search()
7  view=View()
8
9  class Controller:
10     def __init__(self):
11         self.the_student={}
12         self.the_teacher={}
13
14     def check_item(self):
15         while 1:
16             id = input("Enter id:")
17             if id.isnumeric():
18                 if int(id)>0:
19                     return int(id)
20             else:print("Don't enter id: %s" % id)
21
22     def item_select_function(self, function, element,find):
23         while 1:
24             print("-----")
25             print("1. find %s\n2. enter %s id\n3. exit" % (element, element))
26             input_line = input("Enter command: ").strip()
27             if input_line.isnumeric():
28                 input_line = int(input_line)
29                 if (input_line == 1):
30                     input_line = find()
31                     function(input_line)
32                 elif (input_line == 2):self.cheack_id(function)
33                 elif (input_line == 3):break
34                 else:print("Try again")
35             else:print("Please don`t enter this %s id -> (%s)" % (element, input_line))
36
```

```

37     def cheack_id(self,function):
38         while 1:
39             id = input("Enter id:").strip()
40             if id.isnumeric():
41                 function(int(id))
42                 break
43             else:print("Don't enter id: %s" % id)
44
45     def cheack_human(self,item,type):
46         print("Enter %s params:"%type)
47         while 1:
48             item['name'] = input("Name: ").strip()
49             if len(item['name']) == 0: print("Name is empty")
50             else:break
51         while 1:
52             item['surname'] = input("Surname: ").strip()
53             if len(item['surname']) == 0: print("Surname is empty")
54             else:return item
55
56     # student menu
57     def add_new_student(self):
58         if database.select_items("groups"):
59             self.the_student=self.cheack_human(self.the_student,"student")
60             search.find_group()
61             groups = database.find(search.create_query())
62             view.print_item(groups)
63             self.the_student['group'] = False
64             while not self.the_student['group']:
65                 id = self.check_item()
66                 for group in groups:
67                     if (group[0] == id):
68                         self.the_student['group'] = id
69                         break
70             view.add_new_student(self.the_student)
71         else:print("Please add some group before adding student")
72
73     def find_student(self):
74         search.student_params()
75         start_time = time()
76         students = database.find(search.create_query())
77         finish_time = time() - start_time
78         view.print_item(students)
79         print("request execution time: %s" % finish_time,
80               "\n-----")
81         id = self.check_item()
82         for student in students:
83             if (student[0] == id):
84                 return id
85         print("Student with id = (%s) is not included in the list of found students" % id)
86         return 0
87

```

```

87
88     def print_student(self, student_id):
89         if database.select_item_by_id(student_id, "students", "student_id"): view.print_student(student_id)
90         else: print("Student with id: %s not found" % student_id)
91
92     def print_student_by_id(self): self.item_select_function(self.print_student, "student", self.find_student)
93
94     def delete_student(self, student_id):
95         if database.select_item_by_id(student_id, "students", "student_id"): view.delete_student_by_id(student_id)
96         else: print("Student with id: %s not found" % student_id)
97
98     def delete_student_by_id(self): self.item_select_function(self.delete_student, "student", self.find_student)
99
100    def update_student(self, student_id):
101        if database.select_item_by_id(student_id, "students", "student_id"):
102            student = database.select_item_by_id(student_id, "students", "student_id")
103            self.the_student['id'] = student_id
104            print("Enter params:")
105            while 1:
106                self.the_student['name'] = input("Name: ").strip()
107                if len(self.the_student['name']) == 0: self.the_student['name'] = student[0][1]
108                break
109            while 1:
110                self.the_student['surname'] = input("Surname: ").strip()
111                if len(self.the_student['surname']) == 0: self.the_student['surname'] = student[0][2]
112                break
113            search.find_group()
114            groups = database.find(search.create_query())
115            view.print_item(groups)
116            self.the_student['group'] = student[0][3]
117            id = input("Enter id: ").strip()
118            if len(id) != 0 and id.isnumeric():
119                for group in groups:
120                    if (group[0] == int(id)):
121                        self.the_student['group'] = int(id)
122                        return
123            view.update_student(self.the_student)
124        else: print("Student with id: %s not found" % student_id)
125
126    def update_student_by_id(self): self.item_select_function(self.update_student, "student", self.find_student)
127
128    def get_all_student_teacher(self, student_id):
129        if database.select_item_by_id(student_id, "students", "student_id"):
130            query = ""
131            input_line = input("Enter y to set teacher parameters: ")
132            if (input_line == 'y'):
133                search.teacher_params()
134                query = search.create_query()
135                view.get_all_student_teacher(student_id, query)
136        else: print("Student with id: %s not found" % student_id)

```

```

140 def get_all_student_subject(self, student_id):
141     if database.select_item_by_id(student_id, "students", "student_id"):
142         query = ""
143         input_line = input("Enter y to set subject parameters: ")
144         if (input_line == 'y'):
145             search.find_subject()
146             query = search.create_query()
147             view.get_all_student_subject(student_id, query)
148         else: print("Student with id: %s not found" % student_id)
149
150 def get_all_student_by_id_subject(self): self.item_select_function(self.get_all_student_subject, "student", self.find_student)
151
152 # teacher menu1
153 def add_new_teacher(self):
154     self.the_teacher = self.cheack_human(self.the_teacher, "teacher")
155     search.find_subject()
156     subjects=database.find(search.create_query())
157     view.print_item(subjects)
158     self.the_teacher['subject'] = False
159     while not self.the_teacher['subject']:
160         id = self.check_item()
161         for subject in subjects:
162             if (subject[0] == id):
163                 self.the_teacher['subject'] = id
164                 break
165     view.add_new_teacher(self.the_teacher)
166
167 def find_teacher(self):
168     search.teacher_params()
169     start_time=time()
170     teachers = database.find(search.create_query())
171     finish_time=time()-start_time
172     view.print_item(teachers)
173     print("request execution time: %s"%finish_time,"\n-----")
174     id = self.check_item()
175     for teacher in teachers:
176         if (teacher[0] == id):
177             return id
178     print("Teacher with id = (%s) is not included in the list of found teachers" % id)
179     return 0
180
181 def print_teacher(self,teacher_id):
182     if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):view.print_teacher(teacher_id)
183     else:print("Teacher with id: %s not found" % teacher_id)
184
185 def print_teacher_by_id(self):self.item_select_function(self.print_teacher, "teacher", self.find_teacher)
186
187 def delete_teacher_by_id(self):self.item_select_function(view.delete_teacher_by_id,"teacher",self.find_teacher)
188
189 def update_tracher(self, teacher_id):
190     if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
191         teacher = database.select_item_by_id(teacher_id,"teachers","teacher_id")
192         self.the_teacher['id'] = teacher_id
193         print("Enter teacher params:")
194         while 1:
195             self.the_teacher['name'] = input("Name: ").strip()
196             if len(self.the_teacher['name']) == 0: self.the_teacher['name'] = teacher[0][1]
197             break

```

```

198         while 1:
199             self.the_teacher['surname'] = input("Surname: ").strip()
200             if len(self.the_teacher['surname']) == 0: self.the_teacher['surname'] = teacher[0][2]
201             break
202         search.find_subject()
203         subjects = database.find(search.create_query())
204         view.print_item(subjects)
205         self.the_teacher['subject'] = teacher[0][3]
206         id = input("Enter id:").strip()
207         if len(id) != 0 and id.isnumeric():
208             for subject in subjects:
209                 if (subject[0] == int(id)):
210                     self.the_teacher['subject'] = int(id)
211                     return
212         view.update_tracher(self.the_teacher)
213
214     def get_all_teacher_students(self, teacher_id):
215         if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):
216             query=""
217             input_line=input("Enter y to set student parameters: ")
218             if(input_line=='y'):
219                 search.student_params()
220                 query=search.create_query()
221                 view.get_all_teacher_students(teacher_id,query)
222             else:print("Teacher with id-> %s not found"%teacher_id)
223
224     def get_all_subject_teachers(self):
225         search.find_subject()
226         view.print_item(database.find(search.create_query()))
227         subject_id=self.check_item()
228         if database.select_item_by_id(subject_id, "subjects", "subject_id"):
229             query = ""
230             input_line = input("Enter y to set teacher parameters: ")
231             if (input_line == 'y'):
232                 search.teacher_params()
233                 query = search.create_query()
234                 view.get_all_subject_teachers(subject_id, query)
235             else:print("Subject with id-> %s not found"%subject_id)
236
237     def get_all_teacher_by_id_students(self):self.item_select_function(self.get_all_teacher_students, "teacher", self.find_teacher)
238
239     # email menu
240     def add_new_email(self, teacher_id):
241         if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):
242             while 1:
243                 number = input("Enter new email: ")
244                 if len(number) != 0:
245                     view.add_new_email(teacher_id, number)
246                     break
247                 else:print("Name is empty")
248             else:print("Teacher with id: %s not found" % teacher_id)
249
250

```

```

252 def add_new_email_for_teacher(self):self.item_select_function(self.add_new_email, "teacher", self.find_teacher)
253
254 def delete_email(self,teacher_id):
255     if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
256         try:
257             number = input("Enter email address: ").strip()
258             view.delete_email(teacher_id, number)
259             except:print("you enter bad email")
260             else:print("Teacher with id: %s not found" % teacher_id)
261
262 def delete_email_for_teacher(self):self.item_select_function(self.delete_email, "teacher", self.find_teacher)
263
264 def update_email(self,teacher_id):
265     if database.select_item_by_id(teacher_id,"teachers","teacher_id"):
266         try:
267             number1 = input("Enter old email address: ").strip()
268             while len(number1) == 0:
269                 print("Email is empty")
270                 number1 = input("Enter old email address: ").strip()
271             number2 = input("Enter new email address: ").strip()
272             while len(number2) == 0:
273                 print("Email is empty")
274                 number2 = input("Enter new email address: ").strip()
275             view.update_email(teacher_id, number1, number2)
276             except:print("Email %s isn't exist"%number1)
277             else:print("Teacher with id: %s not found" % teacher_id)
278
279 def update_email_for_teacher(self): self.item_select_function(self.update_email, "teacher", self.find_teacher)
280
281 # subject menu
282 def add_new_subject(self):
283     while 1:
284         number = input("Enter subject name:")
285         if len(number) != 0:
286             view.add_new_subject(number)
287             break
288         else:print("Name is empty")
289
290 def delete_subject_by_id(self):
291     subject_id = self.check_item()
292     if database.select_item_by_id(subject_id,"subjects","subject_id"):view.delete_subject_by_id(subject_id)
293     else:print("Subject with id: %s not found" % subject_id)
294
295 def update_subject(self):
296     subject_id = self.check_item()
297     if database.select_item_by_id(subject_id,"subjects","subject_id"):
298         while 1:
299             number = input("Enter new subject name:")
300             if len(number) != 0:
301                 view.update_subject(subject_id, number)
302                 break
303             else:
304                 print("Name is empty")
305         else:print("Subject with id: %s not found" % subject_id)
306

```



```

307 # phone menu
308 def add_new_phone(self, student_id):
309     if database.select_item_by_id(student_id, "students", "student_id"):
310         while 1:
311             number = input("Enter phone number: ")
312             if len(number) != 0:
313                 view.add_new_phone(number, student_id)
314                 break
315             else: print("Phone is empty")
316         else: print("Student with id: %s not found" % student_id)
317
318 def add_new_phone_for_student(self): self.item_select_function(self.add_new_phone, "student", self.find_student)
319
320 def delete_phone(self, student_id):
321     if database.select_item_by_id(student_id, "students", "student_id"):
322         number = input("Enter phone number: ").strip()
323         try:
324             number = int(number)
325             view.delete_phone(student_id, number)
326         except: print("Don't enter this student phone -> %s" % number)
327     else: print("Student with id: %s not found" % student_id)
328
329 def delete_phone_for_student(self): self.item_select_function(self.delete_phone, "student", self.find_student)
330
331 def update_phone(self, student_id):
332     if database.select_item_by_id(student_id, "students", "student_id"):
333         try:
334             number = input("Enter phone number: ").strip()
335             while len(number) == 0:
336                 print("phone is empty")
337                 number = input("Enter phone number: ").strip()
338             new_number = input("Enter new phone: ").strip()
339             while len(new_number) == 0:
340                 print("phone is empty")
341                 new_number = input("Enter new phone: ").strip()
342             view.update_phone(student_id, number, new_number)
343         except: print("Phone %s isn't exist" % number)
344     else: print("Student with id: %s not found" % student_id)
345
346 def update_phone_for_student(self): self.item_select_function(self.update_phone, "student", self.find_student)
347
348 # group menu
349 def add_new_group(self):
350     while 1:
351         number = input("Enter group name:")
352         if len(number) != 0:
353             view.add_new_group(number)
354             break
355         else: print("Name is empty")
356
357 def delete_group_by_id(self):
358     group_id = self.check_item()
359     if database.select_item_by_id(group_id, "groups", "group_id"): view.delete_group_by_id(group_id)
360     else: print("Group with id-> (%s) don't found" % group_id)

```

```

362     def update_group(self):
363         group_id = self.check_item()
364         if database.select_item_by_id(group_id, "groups", "group_id"):
365             group = {};
366             group['id'] = group_id
367             number = input("Enter group name:").strip()
368             while 1:
369                 if len(number) != 0:
370                     group['name'] = number
371                     view.update_group(group)
372                     break
373                 else:print("Name is empty")
374             else:print("Group with id-> (%s) don`t found" % group_id)
375
376     #generation menu
377     def generation_teacher(self):
378         while 1:
379             number = input("Enter count generation teacher:")
380             if len(number) != 0 and number.isnumeric():
381                 view.gen_teacher(int(number))
382                 break
383             else:print("enter another teacher count")
384
385     def generation_student(self):
386         while 1:
387             number = input("Enter count generation student:")
388             if len(number) != 0 and number.isnumeric():
389                 view.gen_student(int(number))
390                 break
391             else:print("enter another teacher count")
392
393     def generation_links(self): view.gen_link()
394
395 #link menu
396 def add_new_student_teacher_link(self):
397     student_id = self.find_student()
398     if database.select_item_by_id(student_id, "students", "student_id"):
399         teacher_id = self.find_teacher()
400         if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):
401             if (not database.get_teacher_student_link(teacher_id, student_id)): view.add_new_student_teacher_link(teacher_id, student_id)
402             else:print("Link already exist")
403         else:print("Teacher with id-> (%s) not found" % teacher_id)
404     else:print("Student with id-> (%s) not found"%student_id)
405
406 def delete_student_teacher_link(self):
407     student_id = self.find_student()
408     if database.select_item_by_id(student_id, "students", "student_id"):
409         teacher_id = self.find_teacher()
410         if database.select_item_by_id(teacher_id, "teachers", "teacher_id"):
411             if (database.get_teacher_student_link(teacher_id, student_id)):
412                 view.delete_student_teacher_link(teacher_id, student_id)
413             else:print("Link don`t exist")
414         else:print("Teacher with id-> (%s) not found" % teacher_id)
415     else:print("Student with id-> (%s) not found"%student_id)

```