



Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ПОСТРЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Отчёт по домашней работе № 2

Выполнил:
студент группы ИУ5 – 23М

Крутов Т. Ю.

Преподаватель:

Виноградова М. В.

2020г.

Задание 1. Создание таблиц БД

Код создания таблицы «Usser» будет выглядеть следующим образом:

```
CREATE TABLE Usser
(
    userID SERIAL PRIMARY KEY,
    FirstName CHARACTER VARYING(40) NOT NULL,
    LastName CHARACTER VARYING(40) NOT NULL,
    Email CHARACTER VARYING(40) NOT NULL,
    Telephone varchar(15) [],
    Addr address,
    BirthDay DATE,
    dat DATE[],
    UNIQUE(Email)
);
```

Полный код создания вышеуказанных таблиц приведён на рисунке ниже, пояснения по ограничениям полей приведены ниже.

```
1 CREATE TYPE address AS (
2     town varchar(20),
3     street varchar (20),
4     house varchar (20)
5 );
6
7 CREATE TABLE Usser
8 (
9     userID SERIAL PRIMARY KEY,
10    FirstName CHARACTER VARYING(40) NOT NULL,
11    LastName CHARACTER VARYING(40) NOT NULL,
12    Email CHARACTER VARYING(40) NOT NULL,
13    Telephone varchar(15) [],
14    Addr address,
15    BirthDay DATE,
16    dat DATE[],
17    UNIQUE(Email)
18 );
19
20 CREATE TABLE Account (
21     accountID SERIAL PRIMARY KEY,
22     userID INTEGER,
23     Login CHARACTER VARYING (20) CHECK (Login != ''),
24     Pass CHARACTER VARYING (20) CHECK (Pass != ''),
25     FOREIGN KEY (userID) REFERENCES Usser (userID) ON UPDATE CASCADE ON DELETE SET NULL,
26     CONSTRAINT check_login UNIQUE (Login)
27 );
28
29 CREATE TABLE Orrder(
30     orderID SERIAL PRIMARY KEY,
31     accountID INTEGER,
32     FOREIGN KEY (accountID) REFERENCES Account (accountID) ON UPDATE CASCADE ON DELETE SET NULL,
33     Ammount_items INTEGER DEFAULT 1 CHECK (Ammount_items > 0 AND Ammount_items < 99),
34     CREATE TYPE OrderType AS ENUM ('Текст', 'Изображение')
35 );
```

Таблица Usser:

- первичный автоинкрементный ключ – userID;
- ограничение на уникальность задано для поля Email;
- поле пользовательского типа address (тип данных address включает три строки переменной длины не более 20 символов – town, street, house) – Addr;
- ограничение на нулевые значения заданы для полей FirstName, LastName, Email.

Таблица Account:

- первичный автоинкрементный ключ – accountID;
- ограничение на уникальность задано для поля Login;
- ограничения на принимаемые значения установлено для полей Login и Pass (строки логина и пароля не должны быть пустыми);
- внешний ключ с ограничениями на связь таблиц – userID (ссылается на таблицу Usser).

Таблица Orrder:

- первичный автоинкрементный ключ – orderID;
- поле перечислимого типа, принимающее только одно из двух возможных значений («Текст» или «Изображение») – OrderType (подробнее ознакомиться с типом можно в разделе «ENUM» методических указаний);
- поле со значением по умолчанию ('1') – Ammount_items;
- ограничение на принимаемые значения ('> 0 и < 99') установлено для поля Ammount_items;
- внешний ключ с ограничениями на связь таблиц – accountID (ссылается на таблицу Account).

Задание 2. Составные типы.

• Многомерные массивы

Следующие команды добавят в таблицу Account поле pass_history типа «текстовый двумерный массив» для хранения данных вида «дата установки пароля – пароль пользователя» и внесут данные об устанавливавшихся паролях пользователя с ID=6.

```
40 ALTER TABLE account ADD COLUMN pass_history text [] []
43 UPDATE account
44 SET pass_history = '{{"02.03.2018", "grumP7h"}, {"04.05.2018", "gse166G"}}'
45 WHERE accountid = 6
```

- **ENUM**

Ограничим выбор типов оцифровываемых объектов типами изображения и текста.

```
CREATE TYPE OrderType AS ENUM ('Текст', 'Изображение')
```

- **Структуры**

Добавим новую запись, содержащую составной тип, выведем все адреса:

```
15 INSERT INTO usser (firstname, lastname, addr, email)
16 VALUES ('Игорь', 'Робертс', ROW ('Ярославль', 'Свердлова', '45'), 'robertsIg@bmstu.ru')
17 SELECT addr FROM usser
```

- **Диапазоны**

```
8 UPDATE orrder SET exec_duration = '['2019-10-20 12:10:00',"2019-10-21 13:15:00")'
9 WHERE orderid = 6
10
11 SELECT * FROM orrder
12 WHERE orderid BETWEEN 6 AND 7;
```

- **Геометрические типы**

Добавим геометрический тип

```
62 ALTER TABLE orrder ADD COLUMN sizes box [] ;
65 UPDATE orrder
66 SET sizes = '{(0,0) (1920,1080) ; (1920,1080)(0,0) ; (2560,1440)(0,0)}'
67 WHERE orderid = 9
```

	orderid [PK] integer	accountid integer	ammount_items integer	ord_type ordertype	exec_duration tsrange	sizes box[]
1	4	5	1	Текст	["2019-08-23 10:45:00","2019-08-23 1...	[null]
2	8	5	3	Изображение	["2019-10-23 10:45:00","2019-10-23 1...	{{(749,749),(657,654);(754,978),(675,123)}}
3	9	5	3	Изображение	["2019-10-20 10:50:00","2019-10-22 1...	{{(1920,1080),(0,0);(1920,1080),(0,0);(2560,1440),(0,0)}}

- **XML**

Для демонстрации работы с XML создадим таблицу с переменной типа XML.

```
37 CREATE TABLE text_info (
38     textid SERIAL PRIMARY KEY,
39     information XML
40 );
```

```

41 INSERT INTO text_info (information) VALUES
42 ('
43     <BasicDetails>
44         <title>Sigh No More</title>
45         <type>poems</type>
46         <author>
47             <first_name>William</first_name>
48             <last_name>Shakespeare</last_name>
49         </author>
50     </BasicDetails>
51 ');
52 INSERT INTO text_info (information) VALUES
53 ('
54     <BasicDetails>
55         <title>Fahrenheit 451</title>
56         <type>novel</type>
57         <author>
58             <first_name>Ray</first_name>
59             <last_name>Bradbury</last_name>
60         </author>
61     </BasicDetails>
62 ');
63 SELECT
64     unnest(xpath('//title/text()', information::XML)) AS title
65     ,unnest(xpath('//type/text()', information::XML)) AS type
66     ,unnest(xpath('//first_name/text()', information::XML)) AS first_name
67     ,unnest(xpath('//last_name/text()', information::XML)) AS last_name
68 FROM text_info;

```

```

5 SELECT
6     (xpath( '//title/text()',information))[1]::text AS name,
7     (xpath( '//last_name/text()',information))[1]::text AS author
8 FROM text_info;

```

• JSON

```

INSERT INTO text_ (orderid, information)
VALUES (3, '{"title": "Sigh No More", "type": "poems", "author": { "first_name": "William", "last_name": "Shakespeare"}}');
INSERT INTO text_ (orderid, information)
VALUES (8, '{"title": "Fahrenheit 451", "type": "novel", "author": { "first_name": "Ray", "last_name": "Bradbury"}}');
INSERT INTO text_ (orderid, information)
VALUES (11, '{"title": "Famed Roman statue not ancient", "type": "article", "author": { "first_name": "", "last_name": ""}}');
INSERT INTO text_ (orderid, information)
VALUES (12, '{"title": "Magic Mushrooms May Permanently Alter Personality", "type": "article", "author": { "first_name": "", "last_name": ""}}');

```

```

20 SELECT orderid,
21     information->'title' AS title, information->'author'->'last_name' AS author
22 FROM text_

```

```

5 SELECT information->'title' AS title
6 FROM text_
7 WHERE information->'author'->'last_name' = ''

```

Задание 3. Секционирование таблиц

1. Создание главной таблицы, от которой будут наследоваться дочерние таблицы:

```

1 CREATE TABLE orders(
2     orderID SERIAL ,
3     accountID INTEGER,
4     FOREIGN KEY(accountID) REFERENCES Account (accountID) ON UPDATE CASCADE ON DELETE SET NULL,
5     ammount_items INTEGER DEFAULT 1 CHECK (Ammount_items > 0 AND Ammount_items < 99 ),
6     order_date date NOT NULL,
7     ord_type ordertype
8 ) ;

```

2. Создание нескольких дочерних таблиц.

```

9 CREATE TABLE order_even() INHERITS (orders);
10 CREATE TABLE order_odd() INHERITS (orders);

```

3. Добавление в таблицы неперекрывающихся ограничений.

```

10 Create table order_even (
11 CHECK (orderID % 2 = 0)
12 )INHERITS (orders);
13
14 Create table order_odd (
15 CHECK (orderID %2 !=0)
16 )INHERITS (orders);

```

4. Добавление индекса

```

20 CREATE INDEX order_even_inx ON order_even(orderID);
21 CREATE INDEX order_odd_inx ON order_odd(orderID);

```

5. Триггерная группа

```

25 CREATE OR REPLACE FUNCTION order_insert_trigger ()
26 RETURNS TRIGGER AS $$
27 BEGIN
28     IF (NEW.orderID % 2 = 0) THEN
29         INSERT INTO order_even VALUES (NEW.*);
30
31     ELSEIF (NEW.orderID % 2 != 0) THEN
32         INSERT INTO order_odd VALUES (NEW.*);
33
34     ELSE
35         RAISE EXCEPTION
36         'Error!!!';
37     END IF;
38     RETURN NULL;
39 END;
40 $$
41 LANGUAGE plpgsql;
42
43 CREATE TRIGGER order_insert_trigger
44 BEFORE INSERT ON orders
45 FOR EACH ROW EXECUTE FUNCTION order_insert_trigger();

```

```

20 CREATE OR REPLACE FUNCTION order_update_trigger ()
21 RETURNS TRIGGER AS $$
22 BEGIN
23     IF (OLD.orderID % 2 = 0) THEN
24         UPDATE order_even
25         SET VALUES = (NEW.*)
26         WHERE orderID = OLD.orderID;
27
28     ELSEIF (OLD.orderID % 2 != 0) THEN
29         UPDATE order_even
30         SET VALUES = (NEW.*)
31         WHERE orderID = OLD.orderID;
32
33     ELSE
34         RAISE EXCEPTION
35         'Error???';
36     END IF;
37     RETURN NULL;
38 END;
39 $$
40 LANGUAGE plpgsql;
41
42 CREATE TRIGGER order_update_trigger
43 BEFORE INSERT ON orders
44 FOR EACH ROW EXECUTE FUNCTION order_update_trigger();

```

Добавление данных:

```

47 insert into orders (accountid, ammount_items, order_date, ord_type)
48 values ('4', '2', '12.03.2020', 'Изображение');
49
50 insert into orders (accountid, ammount_items, order_date, ord_type)
51 values ('5', '2', '12.03.2020', 'Изображение');

```

Задание 4. Полнотекстовый поиск

- Определим свой собственный словарь

```

1 CREATE TEXT SEARCH DICTIONARY public.simple_dict (
2     TEMPLATE = pg_catalog.simple,
3     STOPWORDS = my_own_stop_word_dict
4 );

```

```

7 SELECT ts_lexize('public.simple_dict', 'wow');

```

Data Output	Expli
ts_lexize text[]	
1	{}

- Словарь синонимов

Для добавления собственного словаря синонимов необходимо выполнить команду:

```

3 CREATE TEXT SEARCH DICTIONARY my_own_synonym (
4     TEMPLATE = synonym,
5     SYNONYMS = my_own_synonyms
6 );

```

Поиск:

```
5 SELECT ts_lexize('my_own_synonym','refrigerator');
```

Data Output		Explain
	ts_lexize text[]	
1	{fridge}	

```
4 SELECT 'Smb advised him to get the edu in Paris to become a doc.'::tsvector  
5 @@ to_tsquery('tst_config','doctor');
```

Data Output		Explain
	?column? boolean	
1	true	

- Тезаурус

Для создания нового словаря-тезауруса используется шаблон thesaurus

```
1 CREATE TEXT SEARCH DICTIONARY thesaurus_it (  
2     TEMPLATE = thesaurus,  
3     DictFile = my_own_thesaurus,  
4     Dictionary = english_stem  
5 );
```

Для использования тезауруса «thesaurus_it» его нужно связать с нужной конфигурацией.

```
7 ALTER TEXT SEARCH CONFIGURATION english  
8     ALTER MAPPING FOR asciiword, asciihword, hword_asciipart  
9     WITH thesaurus_it, english_stem;
```

Например, пусть содержимое нового тезауруса «thesaurus_it.ths»:

```
content management systems : cms  
corporate information system : cis  
customer relationship management : crm  
virtual private network : vpn  
application service provider : asp
```

Поиск

```
7 SELECT plainto_tsquery('Content management systems are popular site management tools.');
```

Data Output		Explain	Messages	Not
	plainto_tsquery tsquery			
1	'cms' & 'popular' & 'site' & 'manag' & 'tool'			

```
9 SELECT to_tsvector('Virtual private network is now a common method to stay protected on the network.');
```


Data Output Explain Messages Notifications

	to_tsvector
	tsvector
1	'common':5 'method':6 'network':12 'protect':9 'stay':8 'vpn':1

- Индексы для текстового поиска

```

2 ALTER TABLE text_
3   ADD COLUMN textsearchable_index_col tsvector
4   GENERATED ALWAYS AS (to_tsvector('english', coalesce(body, ''))) STORED;

```

	textid [PK] integer	orderid integer	information jsonb	body text	textsearchable_index_col tsvector
1	1	8	{ "type": "novel", "title": "Fahrenheit 451", "author": { "last...	Ray Bradbury Fahrenheit 451 This on...	'11':9865 '1555':11028 '16':11027 '1790':9469 '451':4,...
2	2	3	{ "type": "poems", "title": "Sigh No More", "author": { "last...	Sigh no more, ladies, sigh no more,	'blith':36 'bonni':38 'constant':23 'convert':39 'deceiv':1...
3	3	11	{ "type": "article", "title": "Famed Roman statue not anci...	A statue symbolising the mythical or...	'1200s':26 '13th':143 '15th':80 '18th':164 '1960':308 '1...
4	4	12	{ "type": "article", "title": "Magic Mushrooms May Perma...	Just one strong dose of hallucinoge...	'14':56,613 '1960s':245 '25':105 '29':629 '30':107,732 '...
5	5	9	{ "type": "article", "title": "The Worlds Largest Indoor Far...	A former Sony Corporation semicon...	'000':63,95 '10':62,260 '100':268 '15':119 '17':109 '18':...

```

9 CREATE INDEX textsearch_idx ON text_ USING GIN (textsearchable_index_col);

```

```

12 SELECT orderid, information->'type' AS type, information->'author'->'last_name' AS author
13 FROM text_
14 WHERE textsearchable_index_col @@ to_tsquery('temperature')
15 ORDER BY orderid DESC
16 LIMIT 5;

```

Data Output Explain Messages N

	orderid integer	type jsonb	author jsonb
1	9	"article"	""
2	8	"novel"	"Bradbury"

- Ранжирование и подсветка результата

```

2 SELECT orderid, information->'type' AS type, information->'author'->'last_name' AS author,
3       ts_rank_cd(to_tsvector('english', body), query) AS rank
4 FROM text_, to_tsquery('english', 'sigh<->no<->more') query
5 WHERE query @@ to_tsvector('english', body)
6 ORDER BY rank DESC
7 LIMIT 3;

```

Data Output Explain Messages Notifications

	orderid integer	type jsonb	author jsonb	rank real
1	3	"poems"	"Shakespeare"	0.3
2	8	"novel"	"Bradbury"	0.1

```

2 SELECT ts_headline( 'english',
3                     body,
4                     to_tsquery('temperature'),
5                     'StartSel = <, StopSel = >, MinWords=10')
6 FROM text_;

```

Data Output	Explain	Messages	Notifications
ts_headline text			
1	A statue symbolising the mythical origins and power of Rome		
2	<temperature> at which book-paper catches fire and burns. Part		
3	Sigh no more, ladies, sigh no more,		
4	Just one strong dose of hallucinogenic mushrooms can alter a		
5	<temperature>, humidity and irrigation, the farm can also cut its water		

Задание 5. Хранимые функции

■ Динамический запрос

```

2 CREATE OR REPLACE FUNCTION select_dynamic(some_table text, some_date date, some_column text, some_element anyelement,
3                                           OUT select_result anyelement)
4 RETURNS anyelement AS
5 $$
6 BEGIN
7     EXECUTE format('SELECT %I FROM %I WHERE %I = $1 ORDER BY %I',
8                   some_element, some_table, some_column, some_element)
9     INTO select_result
10    USING some_date;
11 END;
12 $$
13 LANGUAGE plpgsql;

```

Такую функцию можно вызвать, например, чтобы найти e-mail пользователя с нужной датой рождения.

```

16 SELECT select_dynamic('usser', '2000-09-04', 'birthday', 'email'::varchar);

```

Data Output	Explain	M
select_dynamic character varying		
1	mtfo@mail.ru	

■ Циклы и условия

Рассмотрим хранимую процедуру, содержащую цикл по элементам массива. Такая функция будет подсчитывать общий размер файлов текста в заказе в зависимости от введенного идентификатора заказа.

```

9 CREATE FUNCTION size_of_order (n integer) RETURNS integer AS
10 $$
11 DECLARE
12     size_of_ord INTEGER := 0;
13     counter INTEGER := 0;
14     col integer := 0;
15     arr integer[] ;
16 BEGIN
17     SELECT INTO col (SELECT array_length(text_sizes, 1) FROM orrder where orderid = n);
18     SELECT INTO arr (select text_sizes FROM orrder where orderid = n);
19 LOOP
20     EXIT WHEN counter = col;
21     counter := counter + 1;
22     size_of_ord := size_of_ord + arr[counter];
23 END LOOP;
24 RETURN size_of_ord;
25 END;
26 $$ LANGUAGE plpgsql;
27
28 Select size_of_order(5) as q;

```

▪ Работа с файловой системой

Рассмотрим применение хранимой процедуры для чтения файла, хранящегося в локальной директории. Считываемый файл будем заносить в таблицу.

```

15 CREATE OR REPLACE FUNCTION CSV_INSERT (param int) RETURNS integer
16 AS $$
17 BEGIN
18     IF (param = 1) THEN
19         COPY image (orderid, imgsize)
20         FROM 'C:\downloads\2\data.csv' DELIMITER ',' CSV HEADER;
21         RETURN 1;
22     ELSEIF (param = 2) THEN
23         COPY (SELECT * FROM image) To 'C:\downloads\export.csv' With CSV DELIMITER ',';
24         RETURN 1;
25     ELSE
26         RAISE EXCEPTION
27         'ERROR!!!';
28         RETURN 0;
29     END IF ;
30 END;
31 $$
32 LANGUAGE plpgsql;
33
34 SELECT CSV_INSERT(2);

```

Список литературы

1. Документация PostgreSQL [электронный ресурс]. – Режим доступа: <https://postgrespro.ru/docs/postgresql/>, свободный.
2. Методические указания по выполнению домашнего задания.