

(МГТУ им. Н.Э. Баумана)

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

Вариант 5

(И.О. Фамилия)

Цель:

Сформировать у студента понимание методов упрощения работы аналитика с БД.

Задачи:

- Получить теоретические знания о функциях.
- Подробнее узнать о языке PL/pgSQL.
- Узнать основу синтаксиса языка.
- Ознакомиться с операторами, управляющими конструкциями.
- Получить знания о курсорах и обработки исключений.
- Научится использовать вышеописанные навыки для написания собственной функции.
- Анализ эффективности запросов.
- Узнать о индексах и их влияния на оптимизацию.
- Изучить функции для полнотекстового поиска.

Практическая часть

Часть 1

Составить SQL-скрипты для создания нескольких функций, упрощающих манипуляции с данными.

Меняет страну на Россию у всех Ярославов:

```
create or replace function to_Russia()
returns BOOLEAN as
$$
begin
    update participants set address = address || '{"country": "Russia"}'
    where name like 'Ярослав%';
    return true;
end;
$$
LANGUAGE 'plpgsql';

select * from to_Russia();
```

Результат:

	code	name	phone	address
1	3 749	Ярослава Волков	+7-(968)-411-81-83	{"city": "Johnsonberg", "street": "Emily Stravenue", "country": "Russia", "postcode": "0
2	11 201	Ярослав Каминов	+7-(909)-336-01-47	{"city": "Port Theresastad", "street": "Evans Run", "country": "Russia", "postcode": "84
3	18 185	Ярослав Бажова	+7-(999)-517-51-13	{"city": "Johnnychester", "street": "Alicia Drive", "country": "Russia", "postcode": "741
4	31 119	Ярослав Модестов	+7-(968)-894-84-20	{"city": "Lutzburgh", "street": "Adams Glen", "country": "Russia", "postcode": "22394"
5	36 111	Ярослава Котов	+7-(925)-243-72-89	{"city": "Wrightchester", "street": "Kathy Crest", "country": "Russia", "postcode": "307
6	37 667	Ярослава Овдокимов	+7-(909)-627-59-42	{"city": "Knappberg", "street": "Adkins Inlet", "country": "Russia", "postcode": "22942"
7	61 886	Ярослава Богомякова	+7-(999)-510-48-20	{"city": "Justinview", "street": "Gordon Neck", "country": "Russia", "postcode": "19085"

Продemonстрировать полученные знания о возможностях языка PL/pgSQL.

Циклы, ветвления и переменные:

```
create or replace function biggest_prime_number(N integer)
RETURNS INTEGER AS $$
DECLARE -- блок с переменными
R integer = 1;
del integer = 2;
cnt integer = 0;
s integer;
begin
while R < N loop
    del = 2;
    cnt = 0;
    while del < R loop
        if (R % del = 0)
        then
            cnt = cnt + 1;
        end if;
        del = del + 1;
    end loop;
    if (cnt = 0)
    then
        s = R;
    end if;
    R = R + 1;
end loop;
return s;
end;
$$
LANGUAGE 'plpgsql';
```

```
select * from biggest_prime_number(100);
```

Результат 1 ×

select * from biggest_prime_number(100) | Введите SQL выражение чтобы отфильт

123 biggest_prime_number

97

Работа с курсором:

```

create or replace function participation_date(_from date, _to date)
RETURNS refcursor AS
$$
DECLARE
    curs1 refcursor := 'curs1';
begin
    open curs1 for select * from participation p
    where p.p_date >= _from and p.p_date <= _to;
    return curs1;
end;
$$
LANGUAGE 'plpgsql';

begin;
select * from participation_date('2020-01-01', '2021-01-01');
fetch all from curs1;

```

participation 1 ×

fetch all from curs1 Введите SQL выражение чтобы отфильтровать результаты

	123 code	123 p_part	123 p_coop	p_date	123 p_contribution	p_roles
▶ 1	383 761	37 772 788	9	2020-08-22	78 032	▶ {'participant'}
▶ 2	384 023	48 688 736	10	2020-04-16	91 050	▶ {'manager'}
▶ 3	384 114	48 038 412	6	2020-03-08	49 546	▶ {'accountant', 'manager', 'executor'}
▶ 4	384 147	92 918 148	2	2020-01-11	39 281	▶ {'chief'}

Работа с исключениями:

```

create or replace function owner_func(_own_id integer)
RETURNS boolean AS $$
DECLARE -- блок с переменными
    flag boolean := true;
begin
BEGIN
    -- код, в котором может возникнуть исключение
    select * from owners o
    where o.ow_id = _own_id;
EXCEPTION when others
then
    RAISE NOTICE 'ERROR CODE: %. MESSAGE TEXT: %', SQLSTATE, SQLERRM;
    flag := false;
end;
return flag;
end;
$$
LANGUAGE 'plpgsql';

select * from owner_func(1000000);

```

Вывод ×

ERROR CODE: 42601. MESSAGE TEXT: в запросе нет назначения для данных результата

Часть 2

Запрос к одной таблице, содержащий фильтрацию по нескольким полям

```
select p.code, p.p_part, p.p_date, p.p_contribution
from participation p
where p.p_date > '2022-01-01' and
p.p_contribution > 20000;
```

Результат с параметром explain analyze:

ABC QUERY PLAN	
1	Gather (cost=1000.00..22528.00 rows=15625 width=16) (actual time=0.439..216.824 rows=16714 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Seq Scan on participation p (cost=0.00..19965.50 rows=6510 width=16) (actual time=0.210..161.927 rows=5571 loops=3)
5	Filter: ((p_date > '2022-01-01'::date) AND (p_contribution > 20000))
6	Rows Removed by Filter: 344429
7	Planning Time: 0.091 ms
8	Execution Time: 217.449 ms

Создание индекса

```
create index participation_search on participation(p_date, p_contribution);
```

Результат индекса с параметром explain analyze:

ABC QUERY PLAN	
Bitmap Heap Scan on participation p (cost=376.31..14681.47 rows=15625 width=16) (actual time=3.199..59.523 rows=16714 loops=1)	
Recheck Cond: ((p_date > '2022-01-01'::date) AND (p_contribution > 20000))	
Heap Blocks: exact=9590	
-> Bitmap Index Scan on participation_search (cost=0.00..372.41 rows=15625 width=0) (actual time=2.055..2.055 rows=16714 loops=1)	
Index Cond: ((p_date > '2022-01-01'::date) AND (p_contribution > 20000))	
Planning Time: 1.547 ms	
Execution Time: 60.166 ms	

Время выполнения:

Без индекса: Execution Time: 217.449 ms

С индексом: Execution Time: 60.166 ms

Время выполнения уменьшилось почти в 4 раза

Запрос к нескольким связанным таблицам, содержащий фильтрацию по нескольким полям.

```

select c.cp_name, p.code, p.p_date, p.p_contribution
from participation p, cooperatives c
where p.p_coop = c.cp_id and
c.cp_name = 'PenPen' and
p.p_date > '2022-01-01' and
p.p_contribution > 30000;

```

Результат с параметром explain analyze:

Gather (cost=1019.80..21007.34 rows=70 width=44) (actual time=0.674..98.965 rows=1419 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Hash Join (cost=19.80..20000.34 rows=29 width=44) (actual time=1.130..41.700 rows=473 loops=3)
Hash Cond: (p.p_coop = c.cp_id)
-> Parallel Seq Scan on participation p (cost=0.00..19965.50 rows=5700 width=16) (actual time=0.122..40.480 rows=4894 loops=3)
Filter: ((p_date > '2022-01-01'::date) AND (p_contribution > 30000))
Rows Removed by Filter: 345106
-> Hash (cost=19.75..19.75 rows=4 width=36) (actual time=0.222..0.222 rows=1 loops=3)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Seq Scan on cooperatives c (cost=0.00..19.75 rows=4 width=36) (actual time=0.210..0.212 rows=1 loops=3)
Filter: (cp_name = 'PenPen'::text)
Rows Removed by Filter: 9
Planning Time: 0.271 ms
Execution Time: 99.071 ms

Создание индексов

```
create index participation_search on participation(p_coop, p_date, p_contribution);
```

```
create index cooperative_search on cooperatives(cp_id, cp_name);
```

Результат индекса с параметром explain analyze:

abc QUERY PLAN
Nested Loop (cost=50.77..4080.10 rows=1368 width=44) (actual time=0.361..1.492 rows=1419 loops=1)
-> Seq Scan on cooperatives c (cost=0.00..1.13 rows=1 width=36) (actual time=0.007..0.010 rows=1 loops=1)
Filter: (cp_name = 'PenPen'::text)
Rows Removed by Filter: 9
-> Bitmap Heap Scan on participation p (cost=50.77..4065.30 rows=1368 width=16) (actual time=0.350..1.345 rows=1419 loops=1)
Recheck Cond: ((p_coop = c.cp_id) AND (p_date > '2022-01-01'::date) AND (p_contribution > 30000))
Heap Blocks: exact=1340
-> Bitmap Index Scan on participation_search (cost=0.00..50.43 rows=1368 width=0) (actual time=0.227..0.227 rows=1419 loops=1)
Index Cond: ((p_coop = c.cp_id) AND (p_date > '2022-01-01'::date) AND (p_contribution > 30000))
Planning Time: 2.128 ms
Execution Time: 1.554 ms

Время выполнения:

Execution Time: 99.071 ms - без индекса

Execution Time: 1.554 ms - с индексом

С индексом быстрее почти в 64 раза!

Продемонстрировать полезность индексов для организации полнотекстового поиска, фильтрации с использованием массива и json-формата.

Для работы индексов с полями, содержащими json-формат используем поле address таблицы participants.

```
explain analyze select * from participants p
where p.address::jsonb->>'city' = 'East April'
and p.address::jsonb->>'country' = 'Micronesia';
```

Результат 1 ×

in analyze select * from participants p wh Введите SQL выражение чтобы отфильтровать результаты

ABC QUERY PLAN

Gather (cost=1000.00..3163848.10 rows=2501 width=155) (actual time=0.342..24752.864 rows=37 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Parallel Seq Scan on participants p (cost=0.00..3162598.00 rows=1042 width=155) (actual time=1813.958..24695.256 rows=12 loops=3)
Filter: (((address ->> 'city'::text) = 'East April'::text) AND ((address ->> 'country'::text) = 'Micronesia'::text))
Rows Removed by Filter: 33349988
Planning Time: 0.068 ms
Execution Time: 24752.888 ms

Создаем индексы:

```
create index city on participants using GIN ((address->'city') jsonb_path_ops);
create index country on participants using GIN ((address->'country') jsonb_path_ops);
```

Сравнение: Execution Time: 24752.888 ms

С индексом- Execution Time: 24669.539 ms

Проверим работу полнотекстового поиска с индексом и без него.

```
select p.p_roles[1]
from participation p
where to_tsvector(p.p_roles[1]) @@ plainto_tsquery('chief');
```

Создание индекса:

```
create index roles on participation(p_roles);
```

Без индекса - Execution Time: 1770.549 ms

С индексом - Execution Time: 1670.831 ms

Для таблицы объемом больше 100 млн. записей произвести оптимизацию, позволяющую быстро удалять старые данные, ускорить вставку и чтение данных.

Чтение данных:

```
➤ explain analyze select * from participation p  
  where p.code = 777;
```

Время: Execution Time: 96.926 ms

Удаление данных:

```
➤ explain analyze delete from participation p  
  where p.code = 777;
```

Время: Execution Time: 113.946 ms

Обновление данных:

```
explain analyze update participation p  
  set p.contribution = 33333  
  where code = 777;
```

Время: Execution Time: 108.993 ms

Для ускорения работы удаления, ускорения вставки и чтения данных добавим индексы в таблицу:

```
create index faster on participation(code, p.contribution);
```

Чтение: Execution Time: 0.073 ms

Удаление: Execution Time: 0.059 ms

Обновление: Execution Time: 0.059 ms

Вывод

В ходе выполнения данной лабораторной работы были получены и продемонстрированы знания работы функций в языке SQL.

Продemonстрированы навыки работы с циклами, ветвлениям, переменными, исключениями, курсорами в языке PL/pgSQL. Проанализирована эффективность запросов, изучены и созданы индексы, оптимизирующие работу с таблицами