

Университет ИТМО

Факультет инфокоммуникационных технологий

Дисциплина:

«ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ БАЗ ДАННЫХ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Запросы на выборку и модификацию данных, представления и индексы в PostgreSQL»

Вариант 19

Специальность:

09.03.03 Мобильные и сетевые технологии

Выполнила:

Студентка группы К3240

Вахрушева К.А.

Проверила:

Говорова М.М.

Дата: «__» ____ 2022 г.

Оценка _____

Санкт-Петербург

2022г.

Цель работы: овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, pgadmin 4.

Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Ход работы:

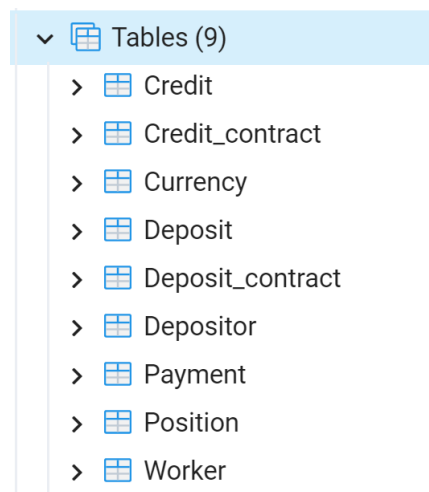


Рисунок 1 - схема базы данных

Запросы

1. Найти вкладчика, имеющего на текущий день несколько вкладов.

```
select count(d.depositor_id), d.depositor_id, d."name" from "default"."Deposit" d
join "default"."Depositor" dep on dep.deposit_id = d.deposit_id
group by d."depositor_id", d."name" having count(*) > 1;
```

count	depositor_id	name
2	1	{Виктор Соловьев}

2. Найти вкладчика, имеющего вклады во всех видах валюты на текущий день.

```
select dep."name", dep.deposit_id, count(dep.name), c."name" from "default"."Deposit" d
join "default"."Depositor" dep on dep.deposit_id = d.deposit_id
join "default"."Currency" c on c.currency_id = d.currency_id
group by dep.name, dep.deposit_id, d.currency_id, c."name" order by dep."name" limit (select count(*) currency_id from "default"."Currency")
```

name	deposit_id	count	name
{Анна Григорьева}	2	1	{Евро}
{Анна Григорьева}	2	1	{Р}
{Анна Григорьева}	2	1	{Д}

3. Вывести данные вкладчика, имеющего максимальный вклад в долларах.

```
select dep."name", dep.deposit_id, c."name", d.interest from "default"."Deposit" d
join "default"."Depositor" dep on dep.deposit_id = d.depositor_id
join "default"."Currency" c on c.currency_id = d.currency_id
group by dep.name, dep.deposit_id, d.currency_id, c."name", d.interest having c."name" = '{$}' order by interest desc limit 1;
```

depositor 1 Depositor(+)... x

select dep."name", dep.deposit_id, c."name", d.interest Введите SQL выражение чтобы отфильтровать результаты

name	deposit_id	name	interest
{Виктор Соловьев}	1	{}	50 000

4. Какой из вкладов пользовался наибольшей популярностью за истекший год.

```
select count(dc.deposit_id), dc.deposit_id from "default"."Deposit_contract" dc
group by dc.deposit_id order by count(dc.deposit_id) ;
```

depositor 1	Deposit_contract 2	
Введите SQL выражение чтобы отфильтровать результаты		
count	deposit_id	
2	1	

5. Кто из сотрудников заключил максимальное число договоров по кредитам за последний месяц.

```
select c.worker_id, count(worker_id) from "default"."Credit_contract" c
where (c.credit_date >= '01-04-2022') and (c.credit_date < '01-05-2022')
group by c.worker_id order by count(worker_id) desc limit 1;
```

credit_contract 1 ×

select c.worker_id, count(worker_id) from "default"." Введите SQL выражение чтобы отфильтровать

	123 worker_id	123 count
1	1	3

6. Вывести список вкладчиков, у которых срок вклада истекает с завтрашнего дня и суммы начислений, которые могут быть ими востребованы.

```
select d.interest, d."name" from "default"."Deposit" d
join "default"."Deposit_contract" cont on cont.deposit_id = d.deposit_id
where cont.return_date >= current_timestamp+interval '1 day';
```

deposit 1 ×

select d.interest, d."name" from "default"."Deposit" Введите SQL выражение чтобы отфильтровать

	123 interest	name
1	20 000	{Виктор Соловьев}
2	20 000	{Виктор Соловьев}

7. Вывести список сотрудников, заключивших договоры по вкладам на максимальную сумму за последний месяц.

```
select cont.return_sum , cont.worker_id, w.first_name , w.middle_name, w.last_name from "default"."Deposit" d
join "default"."Deposit_contract" cont on cont.deposit_id = d.deposit_id
join "default"."Worker" w on w.worker_id = cont.worker_id
where cont.deposit_date >= current_timestamp - interval '30 days' and cont.deposit_date <= current_timestamp order by d.interest
```

osit_contract(+) 1 ×

ct.return_sum, cont.worker_id, w.first_name

return_sum	worker_id	first_name	middle_name	last_name
350 000	1	{Алла}	{Федоровна}	{Зубенко}
350 000	2	{Елена}	{Владимировна}	{Зендина}

Создать представление:

1. Содержащее сведения обо всех сотрудниках банка и заключенных ими договорах по кредитам за прошедший месяц.

```
select w.worker_id, w.first_name, w.middle_name, w.last_name, w.phone_number, w.passport_type, w.passport_num, w.birth_day, w.address, w.position_id,
cc.credit_id, cc.credit_date, cc.repay_date, cc.credit_sum, cc.repay_sum, cc.comment, cc.worker_id, cc.depositor_id, cc.credit_id
from "default"."Worker" w
join "default"."Credit_contract" cc on cc.worker_id = w.worker_id
where cc.credit_date >= current_timestamp - interval '30 days' and cc.credit_date <= current_timestamp;
```

orker(+) 1 ×

lect w.worker_id, w.first_name, w.middle_name, w

worker_id	first_name	middle_name	last_name	phone_number	passport_type	passport_num	birth_day	address	position_id	credit_id
1	{Алла}	{Федоровна}	{Зубенко}	{+79063842299}	{RU}	{123478801}	2000-03-19	{Вяземский 5}	{NULL}	
2	{Елена}	{Владимировна}	{Зендина}	{+79063843399}	{RU}	{74563705}	1995-01-19	{Адмирала Кнь	{NULL}	

2. Найти клиентов банка, имеющих задолженности по кредитам.

```
select * from "default"."Credit_contract" cc
join "default"."Depositor" d on d.deposit_id = cc.depositor_id
where cc.repay_date < current_timestamp ;
```

edit_contract(+) 1 ×

lect * from "default"."Credit_contract" cc join "def"

credit_id	credit_date	repay_date	credit_sum	repay_sum	comment	worker_id	depositor_id	credit_id	depositor_id
5	2022-03-12	2022-04-12	100 000	10 000	NULL	1	2	4	2

Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.

1. Вставить комментарий «Просрочен» по кредитам, которые не выплачивались больше месяца.

```
update "default"."Credit_contract" cc
set "comment" = (select cs."Status" from "default"."Credit_status" cs where id = 2)
where cc.repay_date < current_timestamp - interval '30 days';
```

Статистика 1 ×

update "default"."Credit_cont" Введите SQL выражение чтобы отфильтровать резул

ne	Value
dated Rows	1
ery	update "default"."Credit_contract" cc set "comment" = (select cs."Status" from "default"."Credit_status" cs where id = 2) where cc.repay_date < current_timestamp - interval '30 days'
sh time	Fri May 27 13:00:24 MSK 2022

2. Добавить оплату по кредиту для клиента с id = 1.

```
insert into "default"."Payment" (payment_id, "sum", pay_date, credit_cont_id)
values (1, 10000, current_timestamp, (
select c.credit_cont_id from "default"."Credit_contract" c where c.depositor_id = 1
));
```

Статистика 1 ×

sert into "default"."Payment" Введите SQL выражение чтобы отфильтровать резул

e	Value
dated Rows	1
y	insert into "default"."Payment" (payment_id, "sum", pay_date, credit_cont_id) values (1, 10000, current_timestamp, (select c.credit_cont_id from "default"."Credit_contract" c where c.depositor_id = 1))
n time	Fri May 27 13:30:23 MSK 2022

3. Удалить те кредиты, которые отмечены, как «Закрыт»

```
delete from "default"."Credit_contract" where
"comment" = (select "Status" from "default"."Credit_status" cs where "Status" = '{"Закрыт"}');
```

Статистика 1 ×

delete from "default"."Credit_ Введите SQL выражение чтобы отфильтровать результаты

ne	Value
dated Rows	0
ery	delete from "default"."Credit_contract" where "comment" = (select "Status" from "default"."Credit_status" cs where "Status" = '{"Закрыт"}')
sh time	Fri May 27 13:46:42 MSK 2022

Изучить графическое представление запросов и просмотреть историю запросов

Bank/postgres@PostgreSQL 13 ▾

Query Editor Query History

Show queries generated internally by pgAdmin?

☒ Yes

Today - 27.05.2022

<code>SELECT * FROM "default"."Credit_status"...</code>	13:45:12
<code>SELECT * FROM "default"."Credit_contrac...</code>	13:44:54
<code>BEGIN;</code>	12:56:29
<code>COMMIT;</code>	12:56:29
<code>UPDATE "default"."Credit_status" SET ...</code>	12:56:18

Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

План выполнения для запроса №1:

```
explain select dep."name", dep.deposit_id, c."name", d.interest from "default"."Deposit" d
join "default"."Depositor" dep on dep.deposit_id = d.depositor_id
join "default"."Currency" c on c.currency_id = d.currency_id
group by dep.name, dep.deposit_id, d.currency_id, c."name", d.interest having c."name" = '{$}' order by
```

результат 1 ×

explain select dep."name", dep.deposit_id, c."name", d.interest from "default"."Deposit" d join "default"."Depositor" dep on dep.deposit_id = d.depositor_id join "default"."Currency" c on c.currency_id = d.currency_id group by dep.name, dep.deposit_id, d.currency_id, c."name", d.interest having c."name" = '{\$}' order by

QUERY PLAN

1	Limit (cost=47.77..47.78 rows=1 width=76)
2	-> Group (cost=47.77..47.82 rows=4 width=76)
3	Group Key: d.interest, dep.deposit_id, d.currency_id, c.name
4	-> Sort (cost=47.77..47.78 rows=4 width=76)
5	Sort Key: d.interest DESC, dep.deposit_id, d.currency_id
6	-> Nested Loop (cost=26.07..47.73 rows=4 width=76)
7	-> Hash Join (cost=25.95..47.07 rows=4 width=44)
8	Hash Cond: (d.currency_id = c.currency_id)
9	-> Seq Scan on "Deposit" d (cost=0.00..18.80 rows=880 width=12)
10	-> Hash (cost=25.88..25.88 rows=6 width=36)
11	-> Seq Scan on "Currency" c (cost=0.00..25.88 rows=6 width=36)
12	Filter: (name = '{\$')::character varying[])
13	-> Index Scan using "Depositor_pkey" on "Depositor" dep (cost=0.13..0.16 rows=1 width=36)
14	Index Cond: (deposit_id = d.depositor_id)

Создание простого индекса для таблицы Currency:

<code>create index simple_index on "default"."Currency" (name);</code>		
Статистика 1		
create index simple_index on "def" Введите SQL выражение чтобы отфильтровать результат		
name	Value	
Updated Rows	0	
Query	create index simple_index on "default"."Currency" (name)	
Finish time	Fri May 27 15:19:48 MSK 2022	

Результат:

RBC QUERY PLAN
Limit (cost=29.94..29.94 rows=1 width=76)
-> Sort (cost=29.94..30.44 rows=200 width=76)
Sort Key: d.interest DESC
-> HashAggregate (cost=26.94..28.94 rows=200 width=76)
Group Key: d.interest, dep.deposit_id, d.currency_id, c.name
-> Hash Join (cost=2.07..24.01 rows=293 width=76)
Hash Cond: (d.depositor_id = dep.deposit_id)
-> Hash Join (cost=1.05..22.20 rows=293 width=44)
Hash Cond: (d.currency_id = c.currency_id)
-> Seq Scan on "Deposit" d (cost=0.00..18.80 rows=880 width=12)
-> Hash (cost=1.04..1.04 rows=1 width=36)
-> Seq Scan on "Currency" c (cost=0.00..1.04 rows=1 width=36)
Filter: (name = '\${}':character varying[])
-> Hash (cost=1.01..1.01 rows=1 width=36)
-> Seq Scan on "Depositor" dep (cost=0.00..1.01 rows=1 width=36)

Стоимость получения строки сократилась в полтора раза, операция seq scan в колонке Currency оценена не в 25.88, а в 1.04. Связано это с тем, что мы проиндексировали атрибут, и по нему стало проще искать нужное значение.

Попробуем проиндексировать таблицу Deposit с помощью составного индекса:


```
create index complex_index on "default"."Deposit" (depositor_id, name, currency_id);
```

статистика 1 ×

create index complex_index on "d"  Введите SQL выражение чтобы отфильтровать результаты  

	Value	
ед Rows	0	
	create index complex_index on "default"."Deposit" (depositor_id, name, currency_id)	
time	Fri May 27 15:27:49 MSK 2022	

План запроса:

ABC QUERY PLAN		
1	Limit (cost=3.17..3.18 rows=1 width=76)	
2	-> Group (cost=3.17..3.19 rows=2 width=76)	
3	Group Key: d.interest, dep.deposit_id, d.currency_id, c.name	
4	-> Sort (cost=3.17..3.17 rows=2 width=76)	
5	Sort Key: d.interest DESC, dep.deposit_id, d.currency_id	
6	-> Nested Loop (cost=1.05..3.16 rows=2 width=76)	
7	Join Filter: (d.depositor_id = dep.deposit_id)	
8	-> Seq Scan on "Depositor" dep (cost=0.00..1.01 rows=1 width=36)	
9	-> Hash Join (cost=1.05..2.12 rows=2 width=44)	
10	Hash Cond: (d.currency_id = c.currency_id)	
11	-> Seq Scan on "Deposit" d (cost=0.00..1.05 rows=5 width=12)	
12	-> Hash (cost=1.04..1.04 rows=1 width=36)	
13	-> Seq Scan on "Currency" c (cost=0.00..1.04 rows=1 width=36)	
14	Filter: (name = '{\$'}::character varying[])	

За счет того, что в индексе учтены некоторые атрибуты, используемые функцией group by, стоимость работы с этой функцией сократилась с 26.94 до 3.17.

Попробуем сделать то же самое с другим запросом, а именно:

```

explain select w.worker_id, w.first_name, w.middle_name, w.last_name, w.phone_number, w.passport_type, w.passport_num, w.birth_day,
cc.credit_cont_id, cc.credit_date, cc.repay_date, cc.credit_sum, cc.repay_sum, cc."comment", cc.worker_id, cc.depositor_id, cc.cre
from "default"."Worker" w
join "default"."Credit_contract" cc on cc.worker_id = w.worker_id
where cc.credit_date >= current_timestamp - interval '30 days' and cc.credit_date <= current_timestamp;

```

Результат 1 Результат 2 ×

plain select w.worker_id, w.first_name, w.l Введите SQL выражение чтобы отфильтровать результаты

ABC	QUERY PLAN
	Nested Loop (cost=0.00..30.86 rows=4 width=300)
	Join Filter: (w.worker_id = cc.worker_id)
	-> Seq Scan on "Worker" w (cost=0.00..1.01 rows=1 width=236)
	-> Seq Scan on "Credit_contract" cc (cost=0.00..29.80 rows=4 width=64)
	Filter: ((credit_date <= CURRENT_TIMESTAMP) AND (credit_date >= (CURRENT_TIMESTAMP - '30 days'::interval)))

Индексы:

```

create index simple_index2 on "default"."Credit_contract" (credit_date);

```

Результат 1 Результат 2 ×

plain select w.worker_id, w.first_name, w.l Введите SQL выражение чтобы отфильтровать результаты

ABC	QUERY PLAN
	Nested Loop (cost=0.00..2.14 rows=1 width=300)
	Join Filter: (w.worker_id = cc.worker_id)
	-> Seq Scan on "Worker" w (cost=0.00..1.01 rows=1 width=236)
	-> Seq Scan on "Credit_contract" cc (cost=0.00..1.11 rows=1 width=64)
	Filter: ((credit_date <= CURRENT_TIMESTAMP) AND (credit_date >= (CURRENT_TIMESTAMP - '30 days'::interval)))

С добавлением простого индекса по дате мы оптимизируем выполнение запроса в 15 раз благодаря упрощению поиска по условию where.

```

create index complex_index2 on "default"."Worker" (worker_id, first_name);

```

Результат 1 Результат 2 ×

plain select w.worker_id, w.first_name, w.l Введите SQL выражение чтобы отфильтровать результаты

ABC	QUERY PLAN
	Nested Loop (cost=0.00..2.16 rows=1 width=300)
	Join Filter: (w.worker_id = cc.worker_id)
	-> Seq Scan on "Credit_contract" cc (cost=0.00..1.11 rows=1 width=64)
	Filter: ((credit_date <= CURRENT_TIMESTAMP) AND (credit_date >= (CURRENT_TIMESTAMP - '30 days'::interval)))
	-> Seq Scan on "Worker" w (cost=0.00..1.02 rows=2 width=236)

С добавлением составного индекса мы сталкиваемся с избыточностью: запрос даже становится дороже, потому что созданный индекс не несет для нас пользы в плане оптимизации.

Вывод: мной были созданы представления и запросы на выборку данных к базе данных PostgreSQL, использованы подзапросы при модификации данных и индексы.