

Практическое задание по теме “Транзакции, переменные, представления”

Оглавление

1	Практическое задание по теме “Оптимизация запросов”	2
1.1	Создайте таблицу logs типа Archive. Пусть при каждом создании записи в таблицах users, catalogs и products в таблицу logs помещается время и дата создания записи, название таблицы, идентификатор первичного ключа и содержимое поля name.	2
1.2	(по желанию) Создайте SQL-запрос, который помещает в таблицу users миллион записей.	4
2	Практическое задание по теме “NoSQL”	5
2.1	В базе данных Redis выберите коллекцию для подсчета посещений с определенных IP-адресов.....	5
2.2	При помощи базы данных Redis решите задачу поиска имени пользователя по электронному адресу и наоборот, поиск электронного адреса пользователя по его имени.	8
2.3	Организуйте хранение категорий и товарных позиций учебной базы данных shop в СУБД MongoDB.	9

1 Практическое задание по теме “Оптимизация запросов”

- 1.1 Создайте таблицу logs типа Archive. Пусть при каждом создании записи в таблицах users, catalogs и products в таблицу logs помещается время и дата создания записи, название таблицы, идентификатор первичного ключа и содержимое поля name.

```
USE shop;

DROP TABLE IF EXISTS users;
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) COMMENT 'Имя покупателя',
    birthday_at DATE COMMENT 'Дата рождения',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
    CURRENT_TIMESTAMP
) COMMENT = 'Покупатели';

INSERT INTO users (name, birthday_at) VALUES
    ('Геннадий', '1990-10-05'),
    ('Наталья', '1984-11-12'),
    ('Александр', '1985-05-20'),
    ('Сергей', '1988-02-14'),

DROP TABLE IF EXISTS logs;
CREATE TABLE logs (
    id SERIAL,
    record_created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    tab_name varchar(255) NOT NULL,
    k_id INT UNSIGNED NOT NULL,
    name varchar(255) COMMENT 'Название раздела'
) COMMENT = 'Log Record' ENGINE=Archive;

SHOW TRIGGERS;
SELECT * FROM logs;

DROP TRIGGER IF EXISTS insert_log_products_after_insert;
DELIMITER //
CREATE TRIGGER insert_log_products_after_insert AFTER INSERT ON
products
FOR EACH ROW
BEGIN
INSERT INTO logs( id, record_created_at, tab_name, k_id, name )
VALUES (null, NOW(), "products", new.id, new.name) ;
END//

DELIMITER ;

SELECT * FROM products;
```

```

INSERT INTO products
    (name, description, price, catalog_id)
VALUES
    ('Intel Core i8-8300', 'Проц для персональных компьютеров',
27890.00, 1);

```

```

SELECT * FROM logs;

```

```

DROP TRIGGER IF EXISTS insert_log_catalogs_after_insert;
DELIMITER //
CREATE TRIGGER insert_log_catalogs_after_insert AFTER INSERT ON
catalogs
FOR EACH ROW
BEGIN
INSERT INTO logs ( id, record_created_at, tab_name, k_id, name )
    VALUES (null, NOW(), 'catalogs', new.id, new.name) ;
END//
DELIMITER ;

```

```

DROP TRIGGER IF EXISTS insert_log_users_after_insert;
DELIMITER //
CREATE TRIGGER insert_log_users_after_insert AFTER INSERT ON users
FOR EACH ROW
BEGIN
INSERT INTO logs ( id, record_created_at, tab_name, k_id, name )
    VALUES (null, NOW(), 'users', new.id, new.name) ;
END//
DELIMITER ;

```

```

INSERT INTO users (name, birthday_at) VALUES ('Мефодий', '1880-10-
05');
INSERT INTO users (name, birthday_at) VALUES ('Ануфрий', '1870-10-
05');
SELECT * FROM users;

```

```

DELETE FROM logs WHERE id=1;
SHOW TRIGGERS;

```

	Trigger	Event	Table	Statement	Timing	Created
▶	catalogs_count	INSERT	catalogs	BEGIN SELECT COUNT(*) INTO @total FROM ca...	AFTER	2022-09-16 17:44:33.02
	insert_log_catalogs_after_insert	INSERT	catalogs	BEGIN INSERT INTO logs (id, record_created_a...	AFTER	2022-09-22 13:16:40.87
	check_catalog_id_insert	INSERT	products	BEGIN DECLARE cat_id INT; SELECT id INTO cat...	BEFORE	2022-09-16 18:01:16.35
	validate_name_description_insert	INSERT	products	BEGIN IF NEW.name IS NULL AND NEW.descrip...	BEFORE	2022-09-19 14:55:06.49
	insert_log_products_after_insert	INSERT	products	BEGIN INSERT INTO logs(id, record_created_a...	AFTER	2022-09-22 13:05:40.94
	check_catalog_id_update	UPDATE	products	BEGIN DECLARE cat_id INT; SELECT id INTO cat...	BEFORE	2022-09-16 18:17:01.29
	insert_log_users_after_insert	INSERT	users	BEGIN INSERT INTO logs (id, record_created_a...	AFTER	2022-09-22 13:07:16.11

```
SELECT * FROM logs;
```

	id	record_created_at	tab_name	k_id	name
▶	1	2022-09-22 13:12:47	products	18	Intel Core i8-8300
	2	2022-09-22 13:12:55	users	26	Мефодий
	3	2022-09-22 13:13:03	users	27	Ануфрий
•	NULL	NULL	NULL	NULL	NULL

1.2 (по желанию) Создайте SQL-запрос, который помещает в таблицу users миллион записей.

```
USE shop;
DROP PROCEDURE IF EXISTS insert_names;

DELIMITER //
CREATE PROCEDURE insert_names(IN n INT)
BEGIN
  DECLARE i INT;
  SET i = 0;
  WHILE i < n DO
    INSERT users ( id, name ) VALUES (NULL, CONCAT('user_',i));
    SET i = i + 1;
  END WHILE;
END//
CALL insert_names(10000);

SELECT * FROM users;
```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cel					
	id	name	birthday_at	created_at	updated_at
	25	Ануфрий	1870-10-05	2022-09-22 13:07:53	2022-09-22 13:07:53
	26	Мефодий	1880-10-05	2022-09-22 13:12:55	2022-09-22 13:12:55
	27	Ануфрий	1870-10-05	2022-09-22 13:13:03	2022-09-22 13:13:03
	183060	user_0	NULL	2022-09-22 13:53:04	2022-09-22 13:53:04
	183061	user_1	NULL	2022-09-22 13:53:04	2022-09-22 13:53:04
	183062	user_2	NULL	2022-09-22 13:53:04	2022-09-22 13:53:04
	183063	user_3	NULL	2022-09-22 13:53:04	2022-09-22 13:53:04
	183064	user_4	NULL	2022-09-22 13:53:04	2022-09-22 13:53:04

```
SELECT COUNT(*) FROM users;
```

	COUNT(*)
▶	10013

```
DELETE FROM users where id BETWEEN 28 AND 200000;
```

2 Практическое задание по теме “NoSQL”

- 2.1 В базе данных Redis подберите коллекцию для подсчета посещений с определенных IP-адресов.

Взято отсюда:

https://github.com/bostspb/mysql/blob/master/Scripts/lesson11/task10_01.sql

<https://medium.com/swlh/install-redis-inside-a-ubuntu-vm-d5022d42d8cc>

-- Подготовка

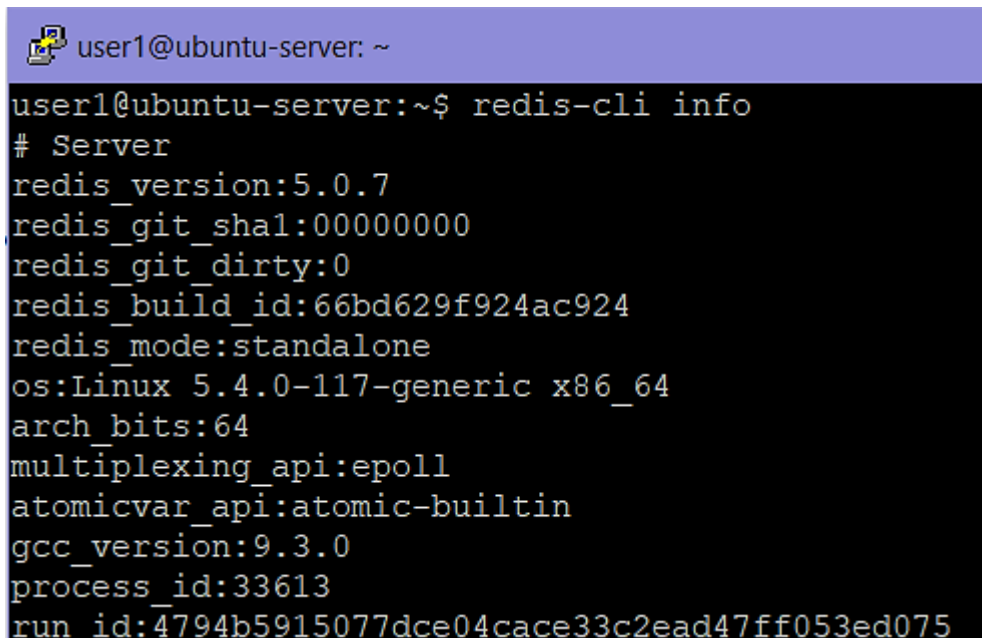
--

Устанавливаем Redis

```
sudo apt update -y
sudo apt install redis-server -y
sudo sed -i 's/^supervised no/supervised systemd/' /etc/redis/redis.conf
sudo systemctl restart redis.service
sudo systemctl status redis
```

Устанавливаем консольный клиент redis-cli для работы с Redis в Windows

-- Проверка redis-cli info

A terminal window with a purple title bar showing the command 'redis-cli info' and its output. The output lists various Redis server details such as version, build ID, mode, and OS.

```
user1@ubuntu-server: ~
user1@ubuntu-server:~$ redis-cli info
# Server
redis_version:5.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:66bd629f924ac924
redis_mode:standalone
os:Linux 5.4.0-117-generic x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:9.3.0
process_id:33613
run_id:4794b5915077dce04cace33c2ead47ff053ed075
```

-- проверка redis-cli info stats

```
user1@ubuntu-server:~$ redis-cli info stats
# Stats
total_connections_received:8
total_commands_processed:13
instantaneous_ops_per_sec:0
total_net_input_bytes:495
total_net_output_bytes:42207
instantaneous_input_kbps:0.00
instantaneous_output_kbps:0.00
rejected_connections:0
sync_full:0
sync_partial_ok:0
sync_partial_err:0
expired_keys:0
expired_stale_perc:0.00
expired_time_cap_reached_count:0
```

-- проверка redis-cli info server

```
user1@ubuntu-server:~$ redis-cli info server
# Server
redis_version:5.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:66bd629f924ac924
redis_mode:standalone
os:Linux 5.4.0-117-generic x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:9.3.0
process_id:33613
run_id:4794b5915077dce04cace33c2ead47ff053ed075
tcp_port:6379
uptime_in_seconds:11826
uptime_in_days:0
hz:10
configured_hz:10
lru_clock:2920255
executable:/usr/bin/redis-server
config_file:/etc/redis/redis.conf
user1@ubuntu-server:~$
```

--

-- Решение задачи

--

Для решения задачи будем использовать коллекцию типа множество, где имя множества будет IP-адрес, а значение – идентификатор одного посещения.

Записываем тестовые данные:

```
sadd "ip:192.168.1.1" "visit:3215111"
```

```
sadd "ip:192.168.1.2" "visit:3214222"
```

```
sadd "ip:192.168.1.1" "visit:3215333"
```

```
sadd "ip:192.168.1.3" "visit:3215444"
```

```
sadd "ip:192.168.1.1" "visit:3215555"
```

Подсчитываем количество посещений с IP-адреса "192.168.1.1":

```
scard "ip:192.168.1.1"
```

```
user1@ubuntu-server:~$ redis-cli
127.0.0.1:6379> sadd "ip:192.168.1.1" "visit:3215111"
(integer) 1
127.0.0.1:6379> sadd "ip:192.168.1.2" "visit:3214222"
(integer) 1
127.0.0.1:6379> sadd "ip:192.168.1.1" "visit:3215333"
(integer) 1
127.0.0.1:6379> sadd "ip:192.168.1.3" "visit:3215444"
(integer) 1
127.0.0.1:6379> sadd "ip:192.168.1.1" "visit:3215555"
(integer) 1
127.0.0.1:6379> scard "ip:192.168.1.1"
(integer) 3
127.0.0.1:6379> █
```

Ответ: с IP address 192.168.1.1 было 3 посещения

- 2.2 При помощи базы данных Redis решите задачу поиска имени пользователя по электронному адресу и наоборот, поиск электронного адреса пользователя по его имени.

Записываем пары данных в БД mail

```
127.0.0.1:6379> HSET mail ecrvmal@yandex.ru "Vlad_Mal"  
(integer) 1
```

```
127.0.0.1:6379> HSET mail Vlad_Mal "ecrvmal@yandex.ru"  
(integer) 1
```

```
127.0.0.1:6379> HSET mail mamamira@rambler.ru "Mira_Mal"  
(integer) 1
```

```
127.0.0.1:6379> HSET mail Mira_Mal "mamamira@rambler.ru"  
(integer) 1
```

-- Проверка

```
127.0.0.1:6379> HGET mail mamamira@rambler.ru  
"Mira_Mal"
```

```
127.0.0.1:6379> HGET mail Vlad_Mal  
"ecrvmal@yandex.ru"
```

```
127.0.0.1:6379> HGET mail ecrvmal@yandex.ru  
"Vlad_Mal"
```


2.3 Организуйте хранение категорий и товарных позиций учебной базы данных shop в СУБД MongoDB.

т.к. MongoDB на компьютере нет, то взято отсюда:

https://github.com/Alex-Smil/MySQL_IC_HomeWorks/commit/b4ba2c4eb6ee81bbde16a5cf573bcef4a04e53fe

```
// *** Табл. products ***
use products
db.products.insert(
  {"name": "Intel Core i3-8100",
   "description": "Процессор для настольных ПК",
   "price": "8000.00",
   "catalog_id": "Процессоры",
   "created_at": new Date(),
   "updated_at": new Date()})

db.products.insertMany(
  [
    {"name": "AMD FX-8320", "description": "Процессор для настольных ПК",
     "price": "4000.00", "catalog_id": "Процессоры",
     "created_at": new Date(), "updated_at": new Date()},
    {"name": "AMD FX-8320E", "description": "Процессор для настольных
     ПК", "price": "4500.00", "catalog_id": "Процессоры", "created_at": new
     Date(), "updated_at": new Date()}]

db.products.find().pretty()
db.products.find({name: "AMD FX-8320"}).pretty()

// *** Табл. catalogs ***
use catalogs
db.catalogs.insertMany([
  {"name": "Процессоры"},
  {"name": "Мат.платы"},
  {"name": "Видеокарты"}])
```