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.

DROP TABLE log;

CREATE TABLE log (

create\_at DATETIME DEFAULT NOW(),

name\_table VARCHAR(100),

table\_id INT,

table\_name VARCHAR(255)

) ENGINE=ARCHIVE;

DELIMITER //

DROP TRIGGER IF EXISTS log\_note\_catalogs;//

CREATE TRIGGER log\_note\_catalogs AFTER INSERT ON catalogs  
FOR EACH ROW  
BEGIN  
INSERT INTO log (name\_table, table\_id, table\_name) SELECT 'catalogs', NEW.id, catalogs.name FROM catalogs ORDER BY catalogs.id DESC LIMIT 1;  
END//

DROP TRIGGER IF EXISTS log\_note\_users;//

CREATE TRIGGER log\_note\_users AFTER INSERT ON users  
FOR EACH ROW  
BEGIN  
INSERT INTO log (name\_table, table\_id, table\_name) SELECT 'users', NEW.id, users.name FROM users ORDER BY users.id DESC LIMIT 1;  
END//

DELIMITER //

DROP TRIGGER IF EXISTS log\_note\_products;//

CREATE TRIGGER log\_note\_products AFTER INSERT ON products  
FOR EACH ROW  
BEGIN  
INSERT INTO log (name\_table, table\_id, table\_name) SELECT ' products', NEW.id, products.name FROM products ORDER BY products.id DESC LIMIT 1;  
END//

\*\*\*правильно

CREATE TABLE Logs (

id INT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

created\_at datetime DEFAULT CURRENT\_TIMESTAMP,

table\_name varchar(50) NOT NULL,

row\_id INT UNSIGNED NOT NULL,

row\_name varchar(255)

) ENGINE = Archive;

CREATE TRIGGER products\_insert AFTER INSERT ON products

FOR EACH ROW

BEGIN

INSERT INTO Logs VALUES (NULL, DEFAULT, "products", NEW.id, NEW.name);

END;

CREATE TRIGGER users\_insert AFTER INSERT ON users

FOR EACH ROW

BEGIN

INSERT INTO Logs VALUES (NULL, DEFAULT, "users", NEW.id, NEW.name);

END;

CREATE TRIGGER catalogs\_insert AFTER INSERT ON catalogs

FOR EACH ROW

BEGIN

INSERT INTO Logs VALUES (NULL, DEFAULT, "catalogs", NEW.id, NEW.name);

END;

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

DELIMITER //

DROP FUNCTION IF EXISTS notes//

CREATE FUNCTION noutes(num INT)

RETURNS VARCHAR(255) DETERMINISTIC

BEGIN

DECLARE i INT DEFAULT 0;

WHILE i < num DO

RETURN 'INSERT INTO users VALUES (DEFAULT, 'DMITRIY', '1974-12-24', DEFAULT, DEFAULT);//';

SET num = num + 1;

END WHILE;

END//

SELECT notes(2)//

\*\*\*правильно

CREATE TABLE samples (

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 samples (name, birthday\_at) VALUES

('Геннадий', '1990-10-05'),

('Наталья', '1984-11-12'),

('Александр', '1985-05-20'),

('Сергей', '1988-02-14'),

('Иван', '1998-01-12'),

('Мария', '1992-08-29'),

('Аркадий', '1994-03-17'),

('Ольга', '1981-07-10'),

('Владимир', '1988-06-12'),

('Екатерина', '1992-09-20');

SELECT

COUNT(\*)

FROM

samples AS fst,

samples AS snd,

samples AS thd,

samples AS fth,

samples AS fif,

samples AS sth;

SELECT COUNT(\*) FROM users;

SELECT \* FROM users LIMIT 10;

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

такие данные можно хранить в хеш-таблице

HMSET ip\_address address '123.123,123' count 5

\*\*\*правильно

HINCRBY addresses '127.0.0.1' 1

HGETALL addresses

HINCRBY addresses '127.0.0.2' 1

HGETALL addresses

HGET addresses '127.0.0.1'

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

такие данные удобнее хранить в коллекции хеш

HSET address\_book kate '123@mail.ru'

HGET address\_book kate

HVALS address\_book

\*\*\*правильно

При помощи базы данных Redis решите задачу поиска имени пользователя по электронному

-- адресу и наоборот, поиск электронного адреса пользователя по его имени.

HSET emails 'igor' 'igorsimdyanov@gmail.com'

HSET emails 'sergey' 'sergey@gmail.com'

HSET emails 'olga' 'olga@mail.ru'

HGET emails 'igor'

HSET users 'igorsimdyanov@gmail.com' 'igor'

HSET users 'sergey@gmail.com' 'sergey'

HSET users 'olga@mail.ru' 'olga'

HGET users 'olga@mail.ru'

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

db.shop.insert({processor: 'Intel Core I3-8100', price: 7890})

db.shop.insert({processor: 'Intel Core I5-7400', price: 12700'})

db.shop.insert({processor: 'AMD FX-8320E', price: 4780'})

show dbs

use shop

db.createCollection('catalogs')

db.createCollection('products')

db.catalogs.insert({name: 'Процессоры'})

db.catalogs.insert({name: 'Мат.платы'})

db.catalogs.insert({name: 'Видеокарты'})

db.products.insert(

{

name: 'Intel Core i3-8100',

description: 'Процессор для настольных персональных компьютеров, основанных на платформе Intel.',

price: 7890.00,

catalog\_id: new ObjectId("5b56c73f88f700498cbdc56b")

}

);

db.products.insert(

{

name: 'Intel Core i5-7400',

description: 'Процессор для настольных персональных компьютеров, основанных на платформе Intel.',

price: 12700.00,

catalog\_id: new ObjectId("5b56c73f88f700498cbdc56b")

}

);

db.products.insert(

{

name: 'ASUS ROG MAXIMUS X HERO',

description: 'Материнская плата ASUS ROG MAXIMUS X HERO, Z370, Socket 1151-V2, DDR4, ATX',

price: 19310.00,

catalog\_id: new ObjectId("5b56c74788f700498cbdc56c")

}

);

db.products.find()

db.products.find({catalog\_id: ObjectId("5b56c73f88f700498cbdc56bdb")})