VỀ CƠ SỞ DỮ LIỆU REDIS SO SÁNH VỚI MYSQL

NHÓM 10
Nguyễn Thùy Linh
Nguyễn Nhật Long
Tăng Đức Thịnh

TỔNG QUAN

redis là một cơ sở dữ liệu **NoSQL** mã nguồn mở

- Lưu trữ dữ liệu dưới dạng **key-value**
- Lưu dữ liệu ở vào **bộ nhớ trong** (RAM)

TỔNG QUAN

- **redis** không chỉ đơn giản lưu **key-value** dưới dạng ***string*** mà còn hỗ trợ nhiều kiểu dữ liệu khác như ***list, hash, set, sorted set***...
- Lợi thế về **hiệu năng** nhờ lưu trữ ở **bộ nhớ trong**
- **atomic operations, transaction**
- Hỗ trợ kết nối với nhiều **ngôn ngữ lập trình** khác nhau

MỤC ĐÍCH VÀ ỨNG DỤNG

redis phù hợp với các ứng dụng cần có

- Thời gian phản hồi **thời gian thực**
- Linh hoạt trong việc mở rộng **quy mô dữ liệu** liên tục
- Đáp ứng khả năng thay đổi **mô hình dữ liệu** (thêm, xóa các trường, đối tượng thường xuyên)
- Chạy trên **nhiều máy chủ** một cách dễ dàng

DỮ LIỆU

redis lưu dữ liệu ở dạng **key-value**

- **key:** định danh một cấu trúc dữ liệu được lưu trữ
- **value:** một hoặc tập các giá trị của key

SET name "Peter"

key value

SADD myset "Hello" "World"

key value

redis

DỮ LIỆU

KIỂU	MÔ TẢ	VÍ DỤ
String	Một dãy các kí tự bao gồm cả **binary** và **non-binary**, tối đa **512M** SET, GET: O(1)	• SET name “Peter” • GET name

redis

DỮ LIỆU

KIỂU	MÔ TẢ	VÍ DỤ
List	Cài đặt theo kiểu **linked list** PUSH, POP: O(1) Truy cập theo index: O(N)	• RPUSH author "Conan" "David" "Ant" • LRANGE author 0 -1

redis

ỨNG DỤNG

Lưu trữ các tweet **mới nhất** của ***Twitter***

- **LPUSH id** để thêm các giá trị id của tweet mới vào trong ***List***
- Khi muốn lấy ra ***10 tweet mới nhất***, sử dụng lệnh **LRANGE 0 9**

Mô hình này tương tự với cấu trúc dữ liệu **Stack**

DỮ LIỆU

KIỂU	MÔ TẢ	VÍ DỤ
Hash	Biểu diễn object có **nhiều trường**, thay cho việc lưu trữ dưới dạng **key object:field** HSET, HGET: O(1)	HMSET user:1 firstname "Roger" lastname "Federer" birth "1987-10-20"

ỨNG DỤNG

Quản lý **session**

- **HMSET** session:alp0j382fmsalf4 *user_id* 12 *role* “admin”
- **EXPIRE** session:alp0j382fmsalf4 3600

Ở đây redis lưu một session ***có thời hạn*** 3600 giây

Việc lưu trữ dưới dạng **key-value** sẽ đơn giản, hiệu năng tốt hơn so với lưu trữ session trong **bảng** như **MySQL**

DỮ LIỆU

KIỂU	MÔ TẢ	VÍ DỤ
Set	Một tập các **string phân biệt *không có thứ tự***, hỗ trợ tìm một phần tử trong set; **hợp, giao** các set giống như trong toán học SADD, SISMEMBER: O(1) SMEMBERS: O(N) SUNION: O(N) (N là tổng số phần tử)	• SADD beauty “Lipstick” “Eyeliner” • SMEMBERS beauty • SISMEMBER beauty “Lipstick”

redis

ỨNG DỤNG

Gắn **tag** cho object: VD cần gắn tag cho các món hàng trên trang Thương mại Điện tử (gắn tag ***beauty, sale*** cho mặt hàng ***Lipstick***)

- **HMSET** product:67 name “Lipstick” price 12
- **SADD** beauty 67
- **SADD** sale 67

Lấy ra các món hàng theo tag *beauty* và *sale*:

SINTER beauty sale (Độ phức tạp **O(N)** với N là số lượng mặt hàng tìm được)

DỮ LIỆU

KIỂU	MÔ TẢ	VÍ DỤ
Sorted Set	Khá giống với set, là một tập các string **khác nhau** nhưng lại **có thứ tự** được đánh số bởi một giá trị gọi là **score** Thêm, xóa, tìm kiếm: O(logN) ZRANGE, ZRANGEBYSCORE: O(logN + M) (M là số bản ghi)	• ZADD mathscore 10 "An" • ZADD mathscore 9 "Binh“ • ZADD mathscore 10 "Chau“ • ZREVRANGE mathscore 0 -1 withscores

redis

ỨNG DỤNG

Cập nhật **bảng xếp hạng**: Điểm của người chơi được **cập nhật liên tục**, cần lấy ra **top 10** người chơi có điểm cao nhất và thứ hạng bản thân trong **thời gian thực**.

- Khi có điểm số mới xuất hiện: **ZADD** score 37 “Peter”
- Truy vấn top 10: **ZREVRANGE** score 0 9
- Truy vấn thứ hạng của user: **ZRANK** score “Peter”

Độ phức tạp **O(logN)** cho tất cả các thao tác trên

TRUY VẤN, GIAO TÁC ĐỒNG THỜI

Sử dụng **transaction**, các thao tác trong transaction được thực hiện **tuần tự**, các thao tác bên ngoài **không thể nào** chen ngang trong quá trình này.

- **MULTI**
- **GET** key
- **INCR** key
- **EXEC**

TRUY VẤN, GIAO TÁC ĐỒNG THỜI

Câu lệnh **WATCH**: kiểm tra một key có bị **thay đổi** hay không, nếu xảy ra sự thay đổi, **transaction** sẽ bị **hủy bỏ**

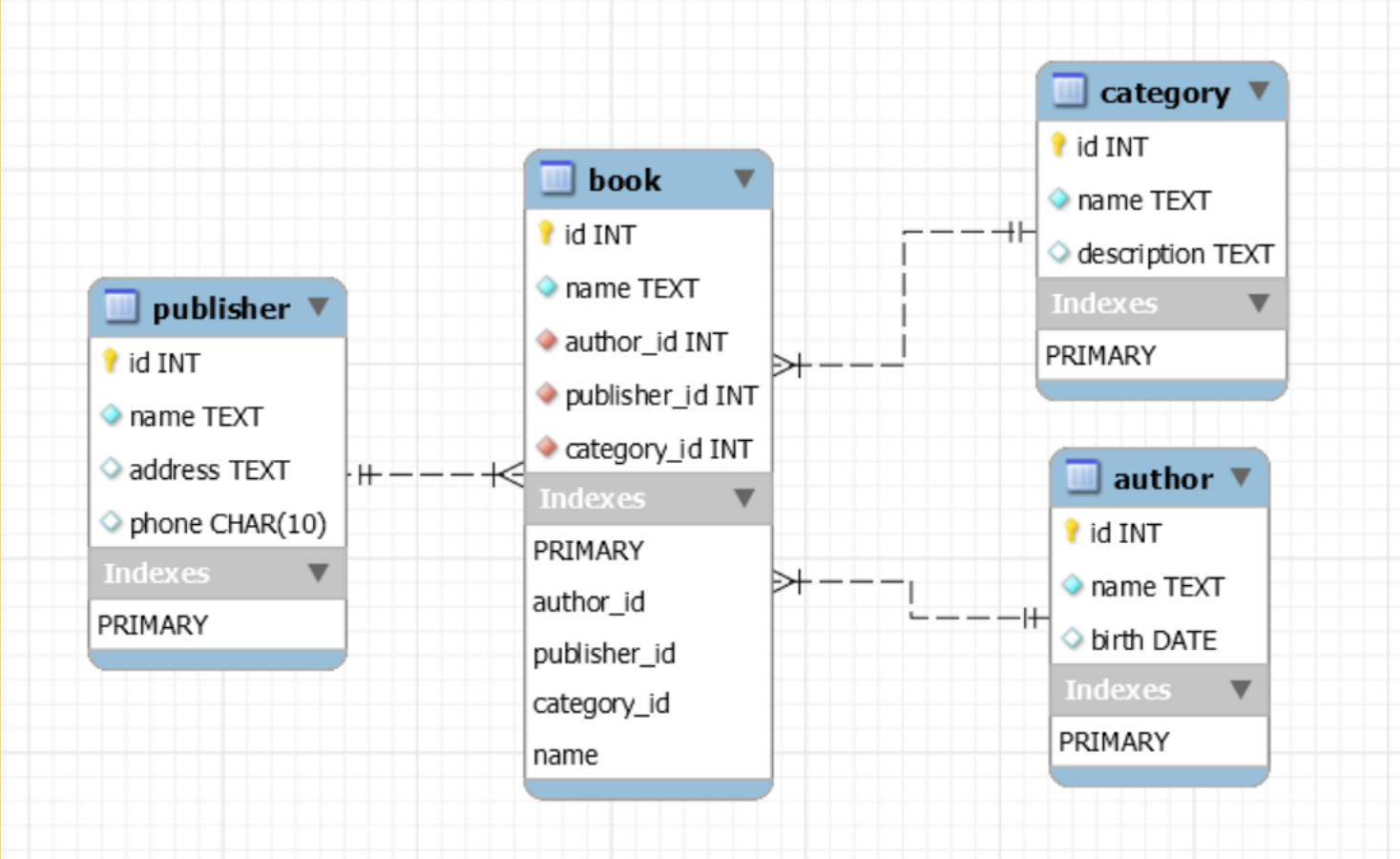
- **WATCH** key
- val = **GET** key
- val = transform(val)
- **MULTI**
- **SET** key val
- **EXEC**

TRUY VẤN, GIAO TÁC ĐỒNG THỜI

Khi có lỗi xảy ra ở các thao tác trong **transaction** thì có 2 trường hợp:

- Lỗi trước câu lệnh **EXEC** (lỗi cú pháp...): transaction bị **hủy bỏ**
- Lỗi sau câu lệnh **EXEC** (lỗi thực thi...): các câu lệnh không bị lỗi **vẫn được thực hiện**

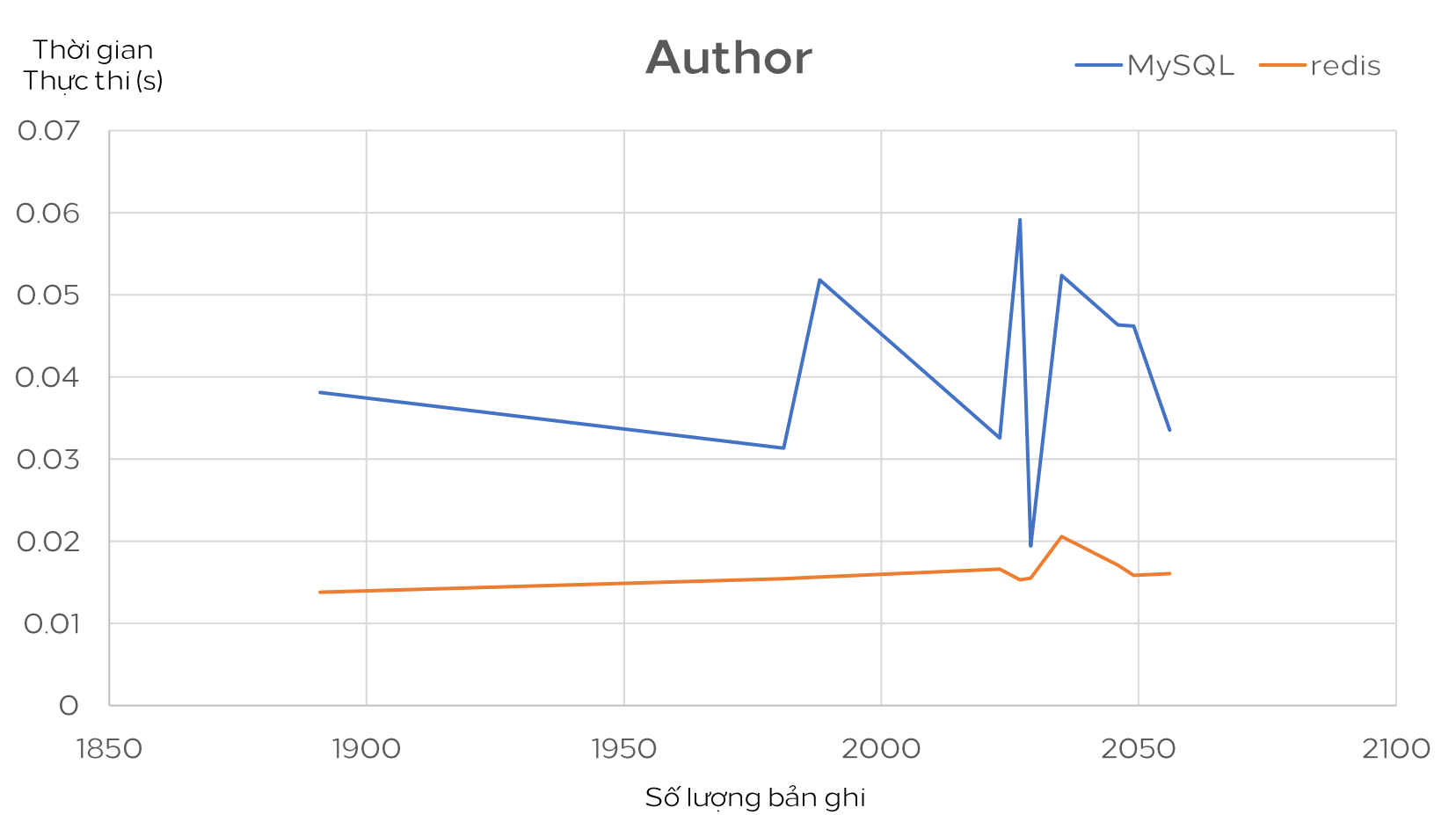
SO SÁNH VỚI MySQL



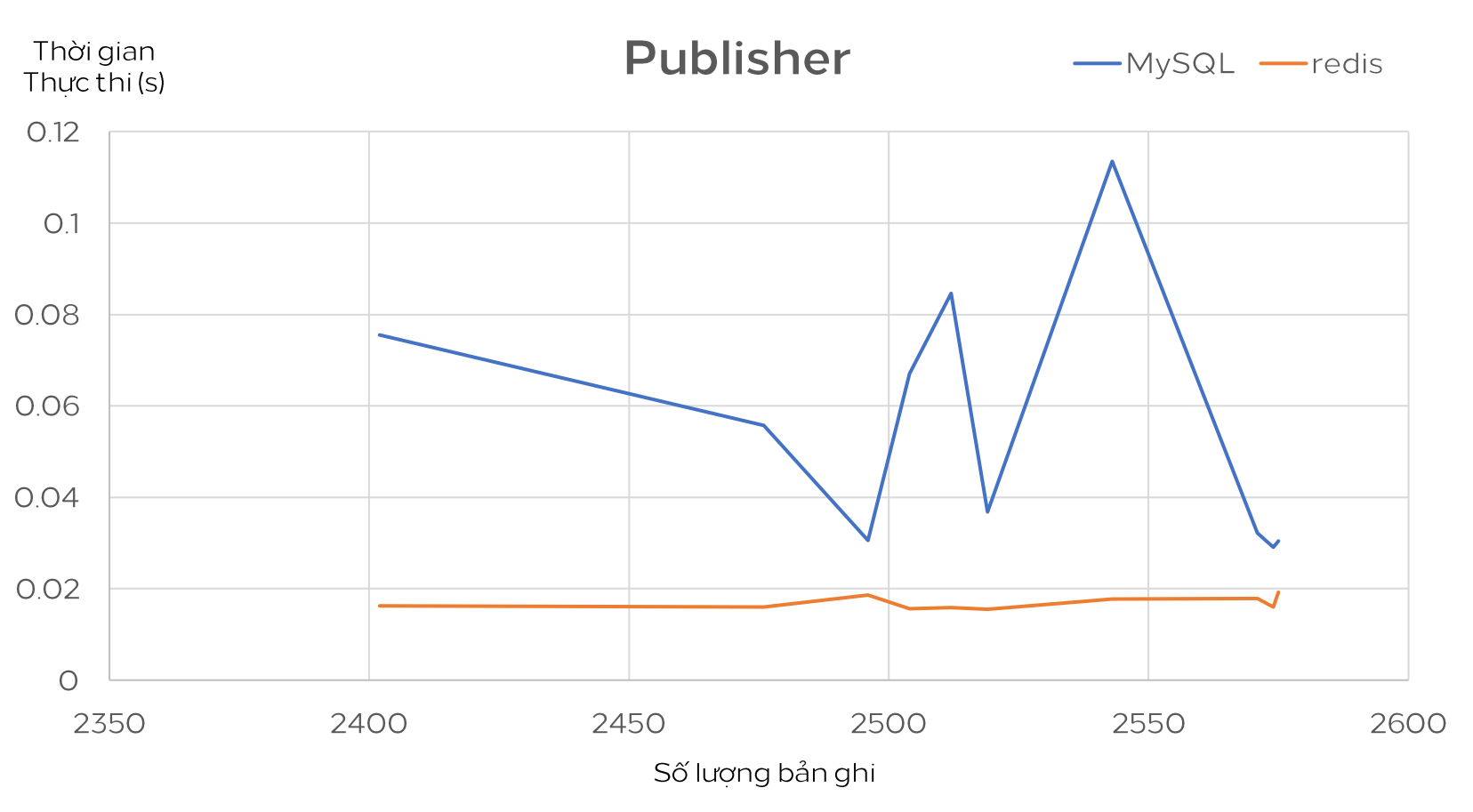
MySQL redis

SO SÁNH VỚI MySQL

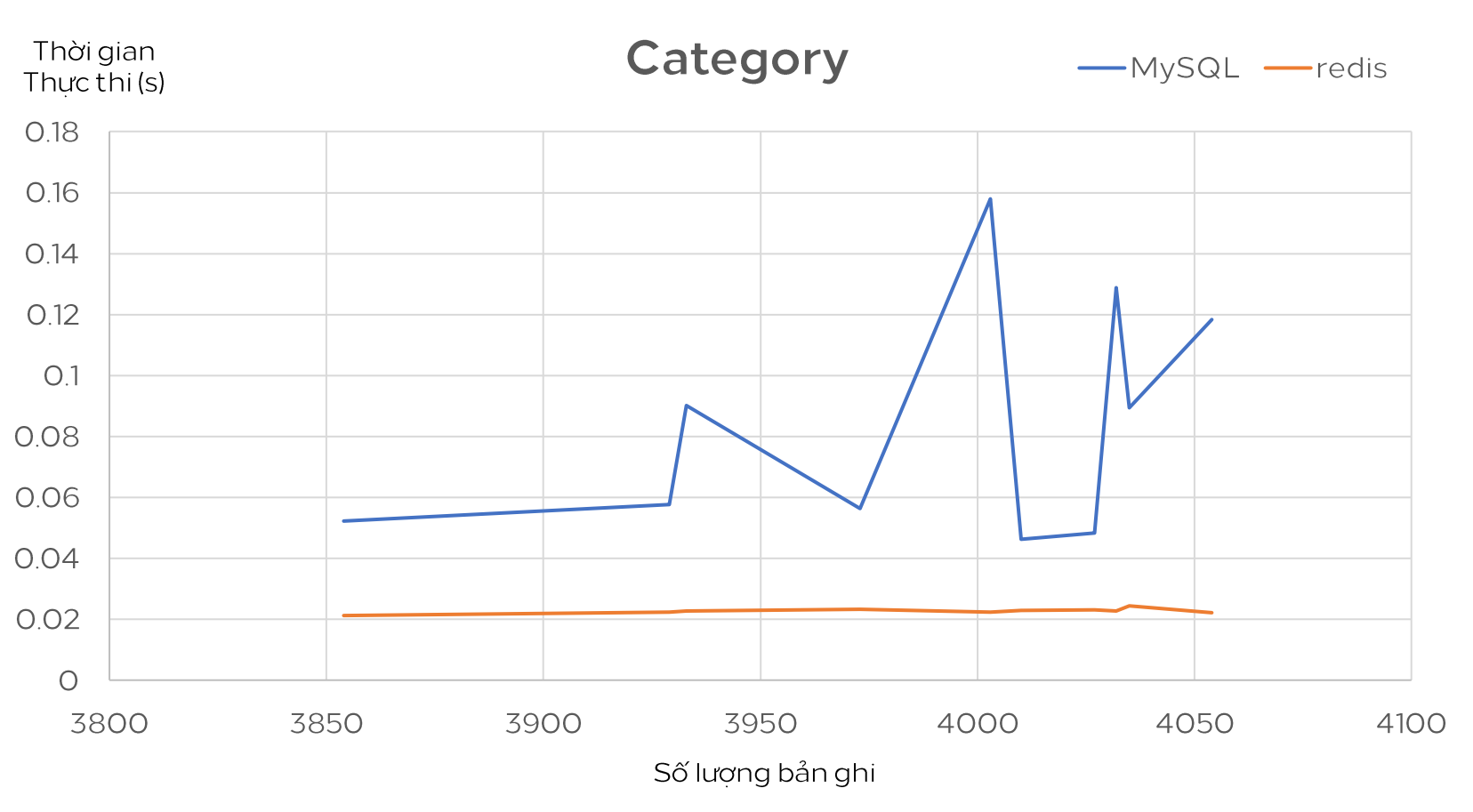
SO SÁNH VỚI MySQL



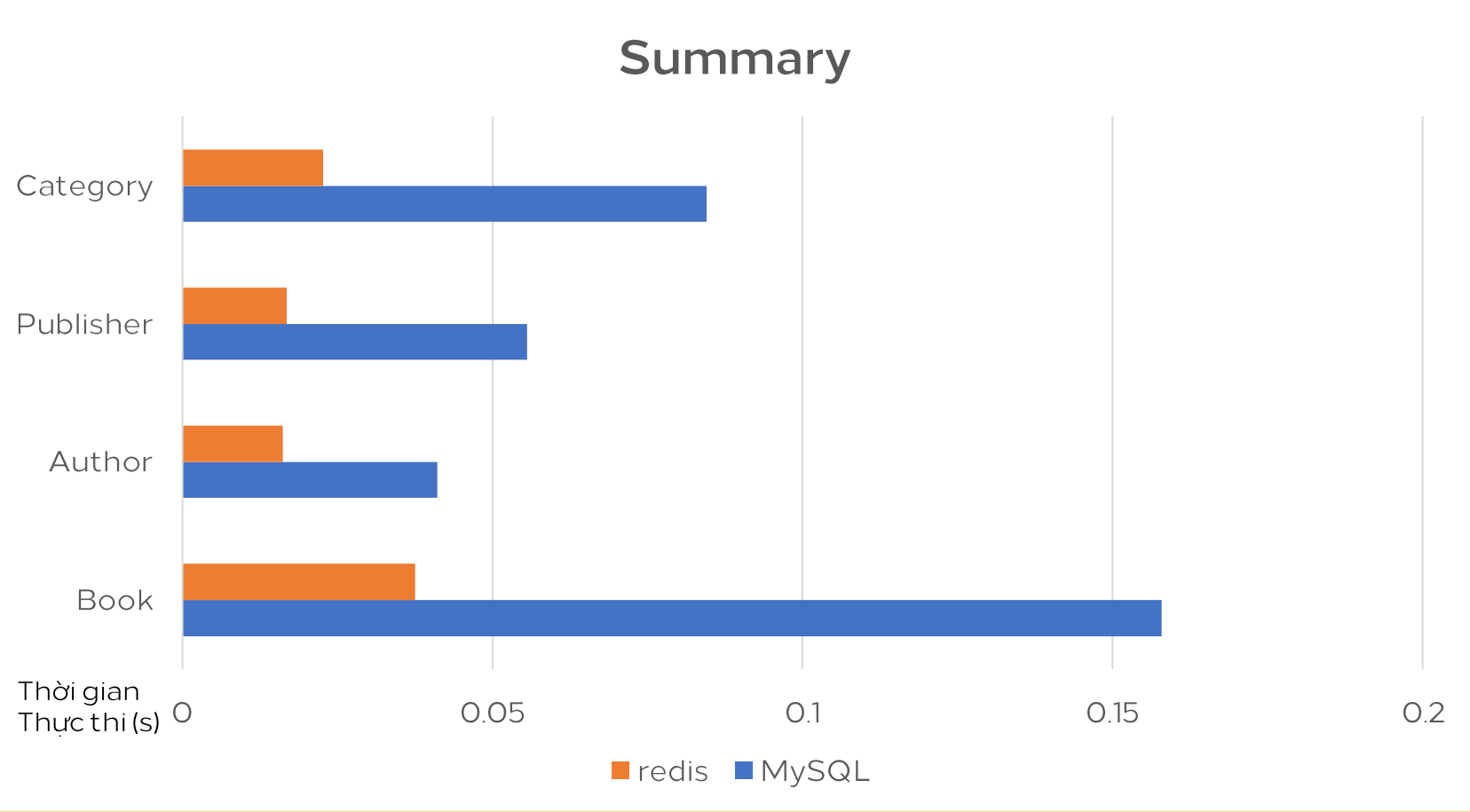
SO SÁNH VỚI MySQL



SO SÁNH VỚI MySQL



SO SÁNH VỚI MySQL



CẢM ƠN CÔ VÀ CÁC BẠN ĐÃ LẮNG NGHE