ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Trần Hữu Đức Mạnh-1902016
Nguyễn Văn Thắng-19020440

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ORIENTDB VÀ

PHÂN TÍCH THIẾT KẾ HỆ THỐNG WEBSITE CHATTING

Ngành: Công nghệ thông tin

Cán bộ hướng dẫn: TS.Nguyễn Thị Hậu

Tóm tắt

Tóm tắt: Hiện nay, mạng xã hội và các hoạt động trực tuyến ngày càng phát triển dẫn đến dữ liệu được lưu ngày càng nhiều và phức tạp việc lưu trữ dữ liệu ngày càng khó khăn và là phần rất quan trọng của các hệ thống, việc chọn lựa hệ quản trị phù hợp với hệ thống mang lại nhiều lợi ích cho hệ thống. Trong đó OrientDB là một hệ quản trị NoSQL phổ biến và rất được tin dùng bởi khả năng xử lý nhanh chóng và hiệu quả. Vì thế, trong báo cáo này em xin được giới thiệu với cô và các bạn về hệ quản trị cơ sở dữ liệu OrientDB . Nội dung chính của báo cáo này tập trung vào cách thức hoạt động, các đặc điểm nổi bật, minh họa cài đặt và sử dụng OrientDB và so sánh nó với các hệ quản trị cùng dòng khác. Cùng với đó em sẽ trình bày về phân tích và thiết kế hệ thống của bài tập lớn cuối kỳ

MỤC LỤC

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ORIENTDB VÀ 1

PHÂN TÍCH THIẾT KẾ HỆ THỐNG WEBSITE CHATTING 1

Tóm tắt 2

MỤC LỤC 2

Danh sách bảng 4

Danh sách ảnh 5

CHƯƠNG I: GIỚI THIỆU CHUNG 6

1.1 Giới thiệu về OrientDb 6

1.2 Cài đặt 6

CHƯƠNG II: TÌM HIỂU ORIENTDB 14

2.1 Mô hình dữ liệu (Data Model) 14

2.2 Cú pháp truy vấn 14

2.3 Kiến trúc 16

2.4 Scalability 17

2.5 Use Case: Time Series 18

CHƯƠNG III: SO SÁNH ORIENTDB 19

3.1 Với MongoDB, Arrango và Neo4j 19

3.2 Lí do cho sự thất thế 20

CHƯƠNG IV: KIẾN TRÚC CHAT APP 20

4.1 Tổng quan hệ thống 21

4.1.1 Tầng giao diện 21

4.1.2 Tầng máy chủ 22

4.1.3 Tầng CSDL 22

4.2 Các công nghệ dự kiến sử dụng 22

4.2.1. Front-End 22

4.2.2 Back-End 22

4.2.3 CSDL: MongoDB 22

4.3 Các chức năng dự kiến 23

4.3.1 Các yêu cầu chức năng 23

4.3.2 Các yêu cầu phi chức năng 23

4.4 Thiết kế CSDL 23

4.4.1 Lược đồ cơ sở dữ liệu 23

4.4.2 Thông tin user Document 25

4.4.3 Thông tin Conversations Document 25

4.4.4 Thông tin bảng Message Document 26

THAM KHẢO 27

Danh sách bảng

Danh sách ảnh

CHƯƠNG I: GIỚI THIỆU CHUNG

1.1 Giới thiệu về OrientDb

- OrientDB là một cơ sở dữ liệu NoSQL đa mô hình với sự hỗ trợ cho dữ liệu đồ thị và tài liệu, điểm đáng chú ý của OrientDB đó chính là nó được phát triển bằng ngôn ngữ lập trình Java nên có thể được cài đặt trên bất kỳ hệ điều hành nào các bạn muốn, đây là một lợi thế lớn so với các cơ sở dữ liệu NoSQL khác. Cơ sở dữ liệu OrientDB cũng mang đầy đủ các đặc tính của một cơ sở dữ liệu NoSQL, tương tự như MongoDB.
- OrientDB hỗ trợ cơ sở dữ liệu đồ thị cho phép các bạn lưu trữ các thực thể và quan hệ giữa các thực thể. Các đối tượng này còn được gọi là các nút, trong đó có các thuộc tính, mỗi nút là một thể hiện của một đối tượng trong ứng dụng. Và tất nhiên, một cơ sở dữ liệu NoSQL thì không thể thiếu dữ liệu tài liệu, một thành phần quản lý dữ liệu rất quan trọng.
- Điểm mạnh của OrientDB đó chính là tốc độ thực thi, trả về kết quả các truy vấn rất nhanh, giúp tăng tốc độ ứng dụng của các bạn. Nếu các bạn đang thiết kế một ứng dụng với nhiều thao tác lấy dữ liệu hơn so với ghi dữ liệu thì OrientDB là một sự lựa chọn tốt để cải thiện hiệu năng ứng dụng.

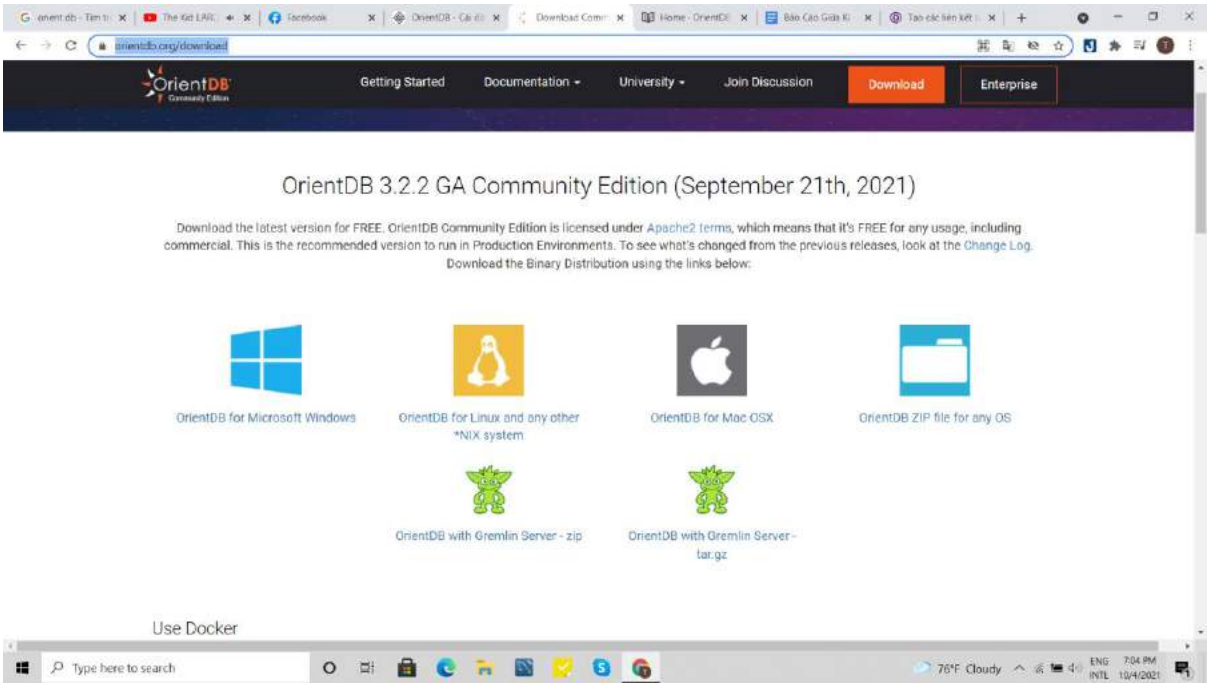
1.2 Cài đặt

- Điều kiện tiên quyết: Cả phiên bản Community và Enterprise đều có thể chạy trên bất kỳ Hệ điều hành nào triển khai Máy ảo Java (JVM). OrientDB yêu cầu Java với phiên bản 1.7 trở lên.

- **Bước 1**:

  Vào trang chủ của OrientDb, ảnh chụp màn hình sau đây cho thấy trang tải xuống của OrientDB. Bạn có thể tải xuống tệp nén hoặc tarred bằng cách nhấp vào biểu tượng

hệ điều hành phù hợp.



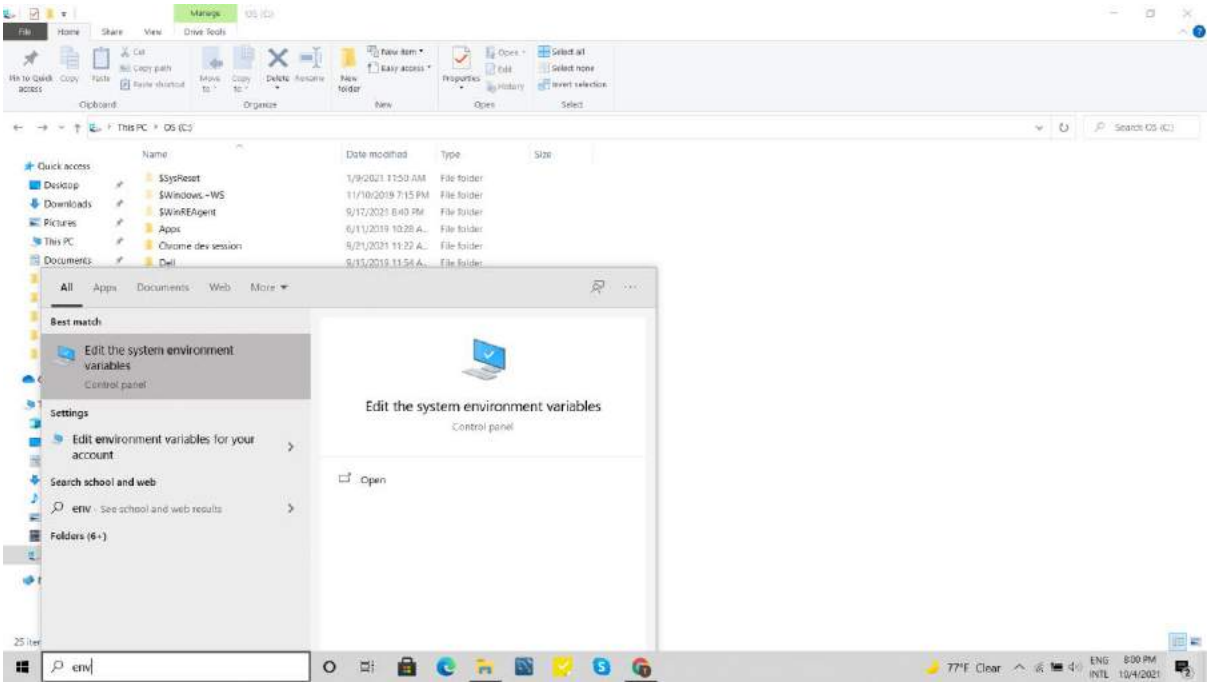
Hình 1.1. Trang chủ OrientDB

(link dẫn đến trang chủ https://orientdb.org/download).

❖ **Bước 2**: Giải nén và cài đặt

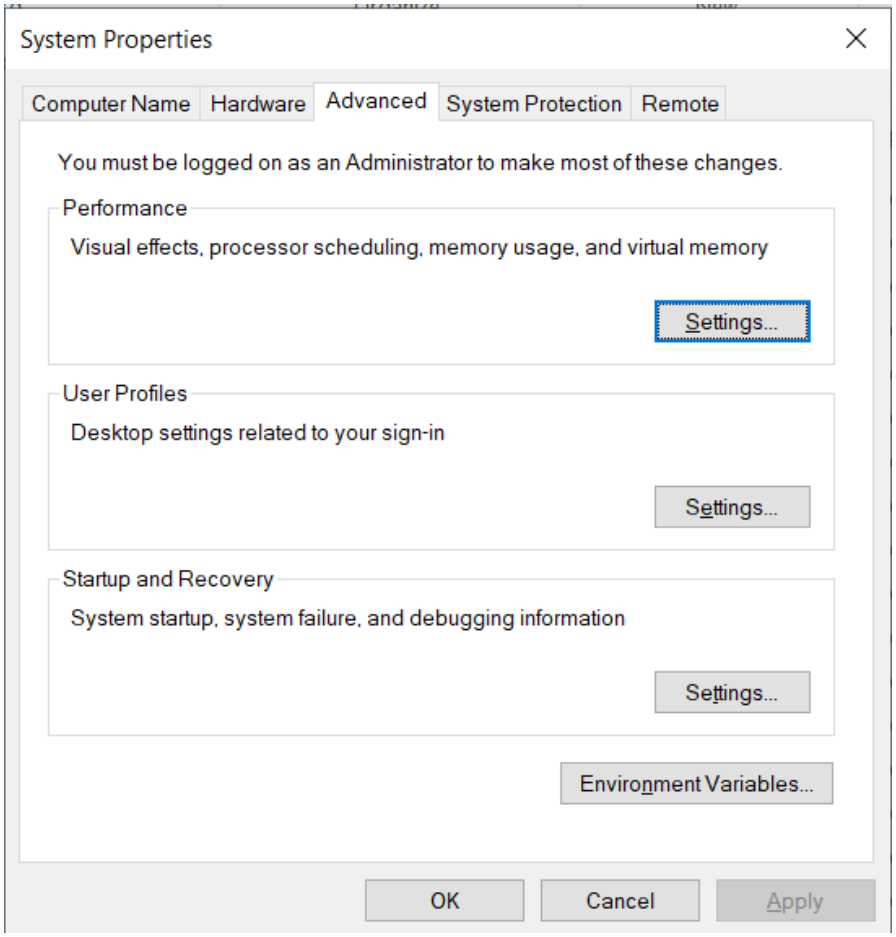
Sau khi giải nén thì di chuyển thư mục vừa giải nén sang ổ C

Tạo hai biến môi trường ORIENTDB_HOME và biến PATH theo cách sau:



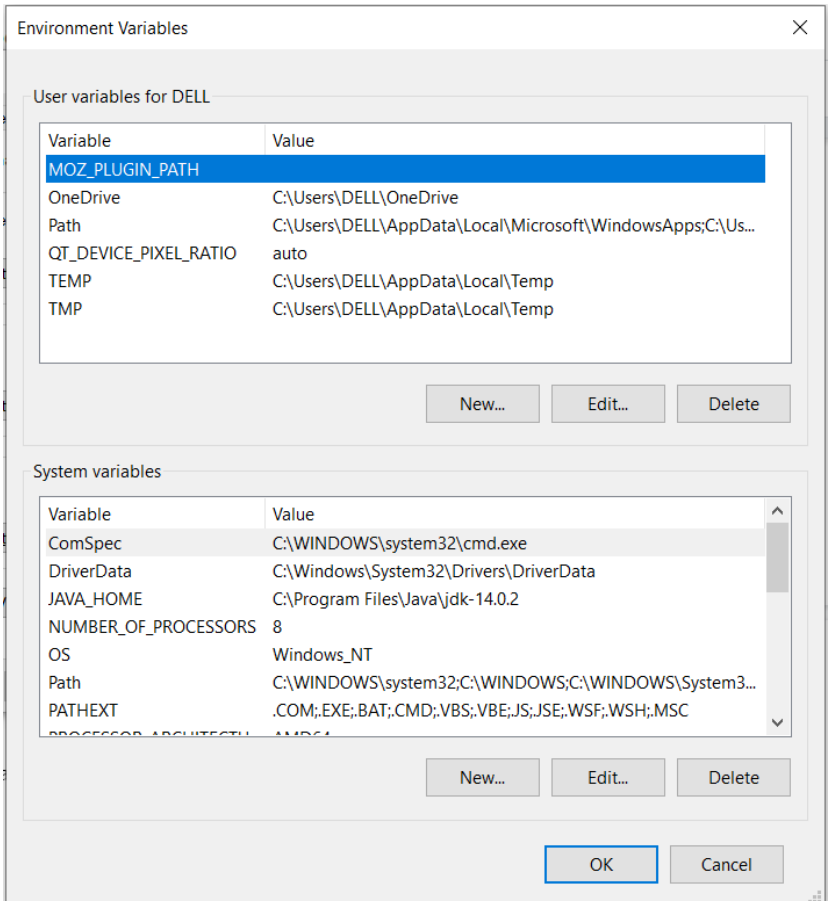
Hình 1.2. Mở cửa sổ biến môi trường

Đầu tiên gõ “env”ở thanh công cụ cạnh biểu tượng window dưới góc phải màn hình rồi nhấn ENTER.
Sau đó 1 của sổ như hình dưới sẽ hiện ra nhấn vào “Environment Variables”:



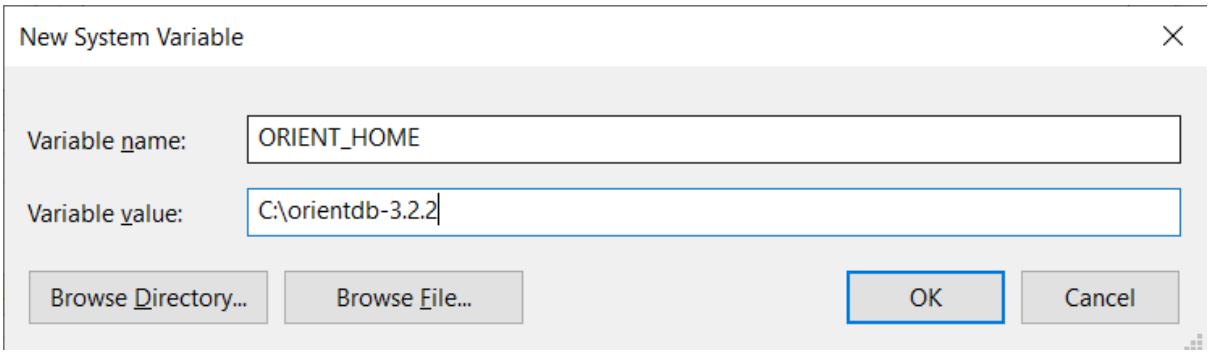
Hình 1.3. Cửa sổ biến môi trường

Sau đó 1 cửa sổ mới có hình dưới đây sẽ hiện ra và ta ấn “New” tại phần “System variables”:



Hình 1.4. Cửa sổ chi tiết của các biến môi trường

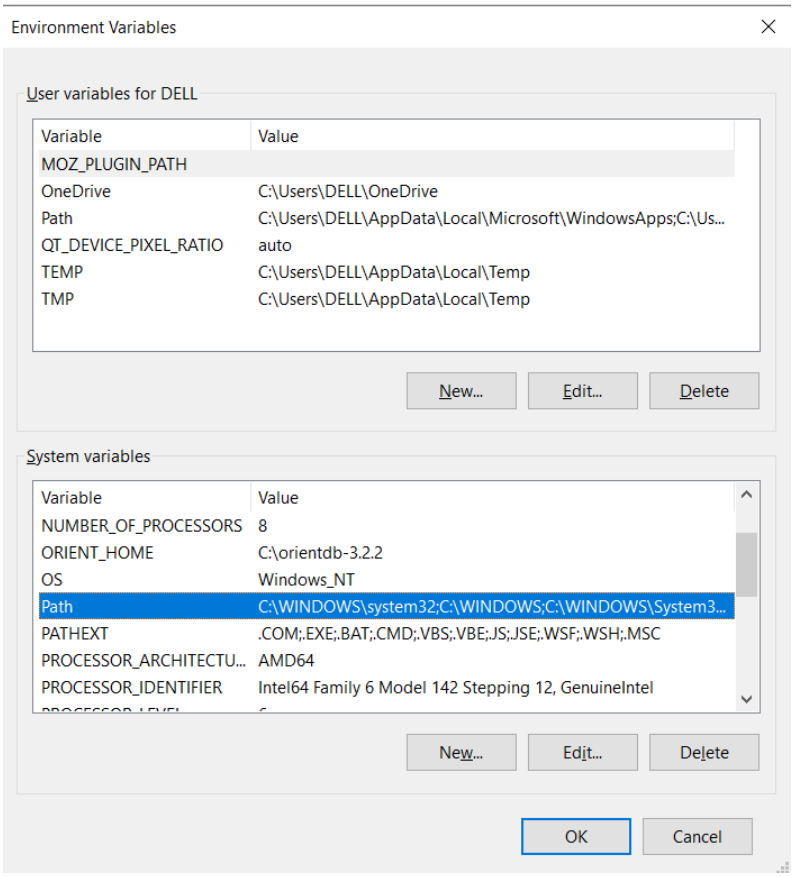
Sau đó ta được hình dưới đây:



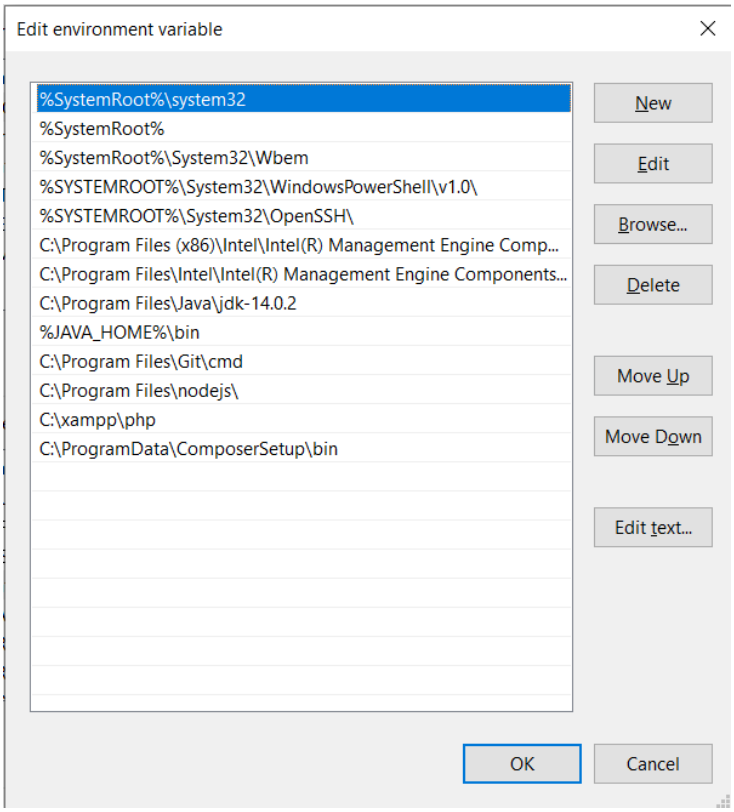
Hình 1.5. Cấu hình biến môi trường ORIENT_HOME

Tại ô "Variable name" ta điền như hình trên, ô "Variable value" ta điền đường dẫn tới thư mục mà chúng ta vừa giải nén nhấn "OK".

Tiếp theo ta tiếp tục tìm kiếm biến môi trường tại "System variable" như hình dưới

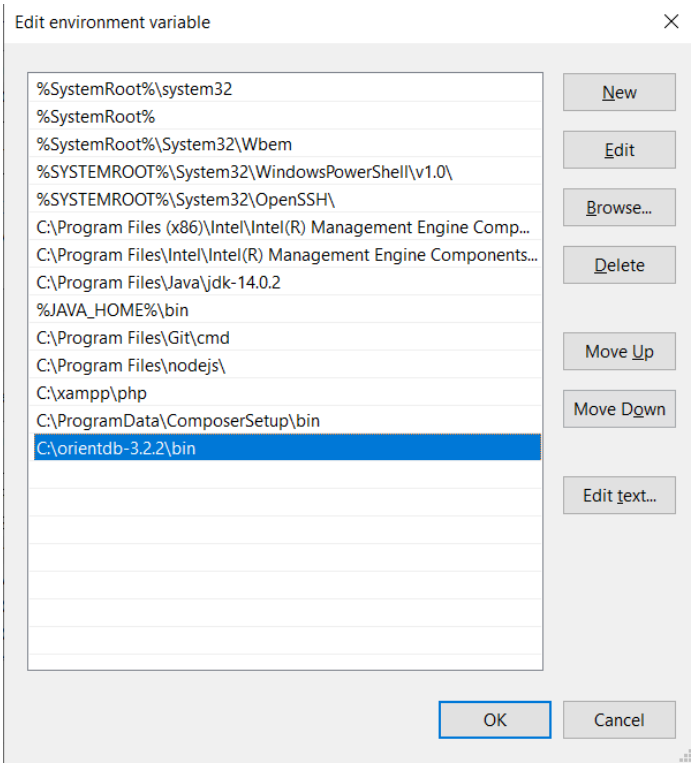


Hình 1.6. Hình ảnh sau khi cài biến môi trường ORIENT_HOME thành công ”Double click” vào “Variable” là “Path” như hình trên thì sẽ xuất hiện của sổ như hình dưới đây:



Hình 1.7. Cửa sổ của biến môi trường với “variable name” là “Path”

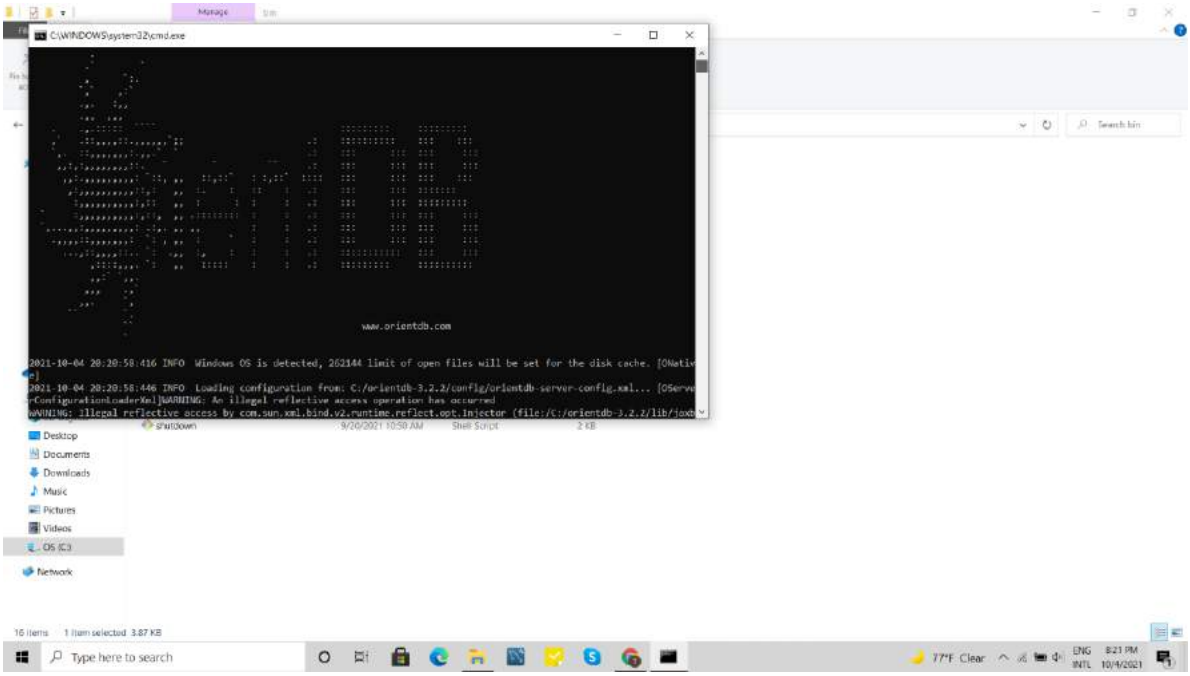
Chọn “New” và nhập đường dẫn tới thư mục “bin” của thư mục ta vừa giải nén như hình dưới đây và nhấn “OK”



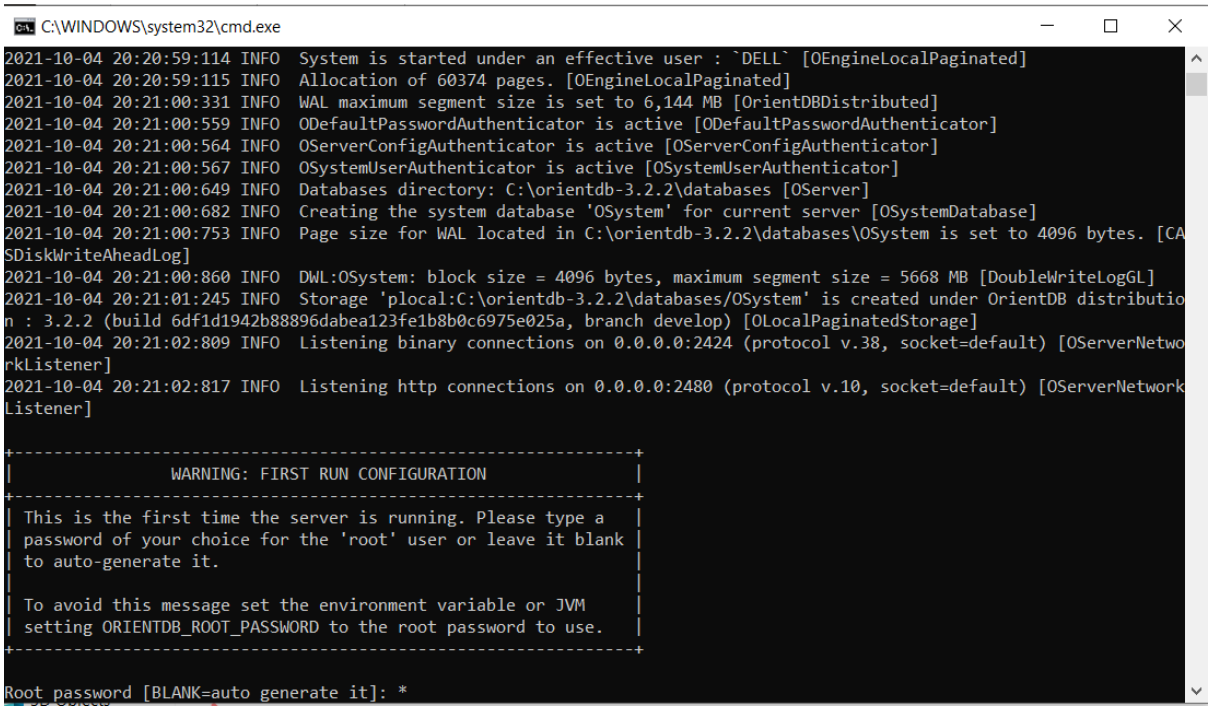
Hình 1.8. Cấu hình thêm đường dẫn tới thư mục “bin” trong “Path”

- **Bước 3**: Chạy file và kết thúc

Bạn vào thư mục “bin” trong thư mục mà bạn vừa tải về hay cài đặt tìm file “server.bat” rồi chạy ta được hình như dưới đây:

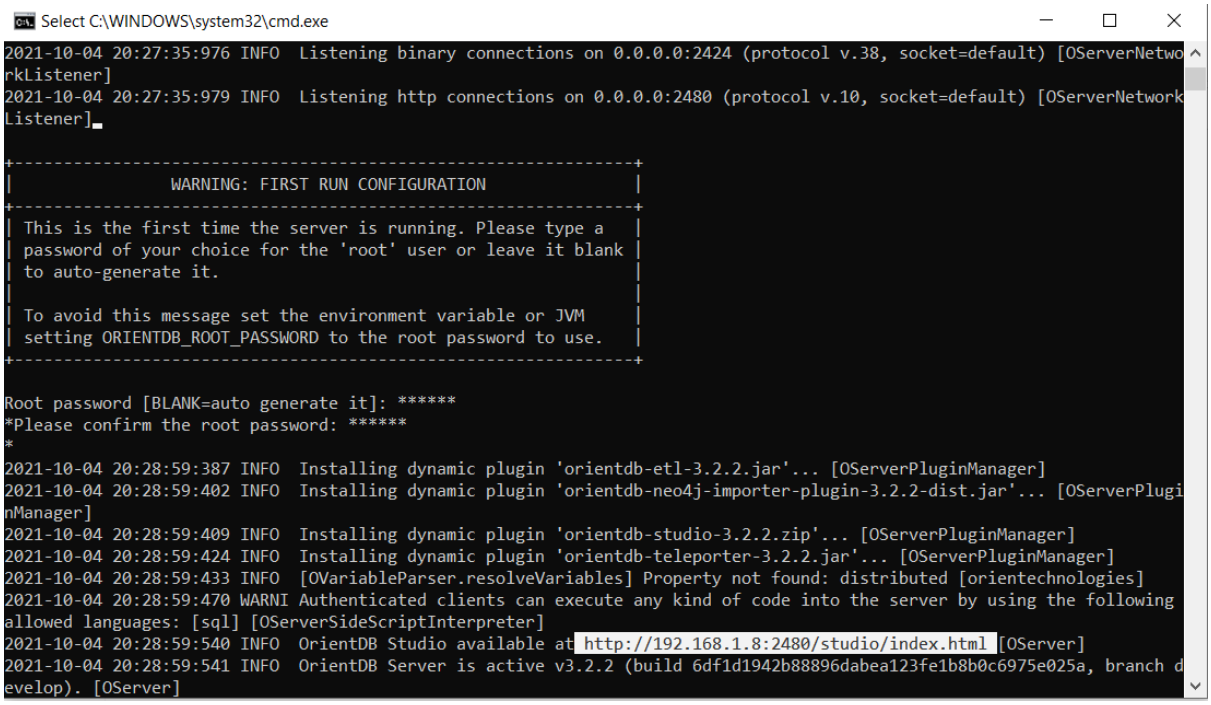


Hình 1.9. Màn hình khi chạy file server.bath



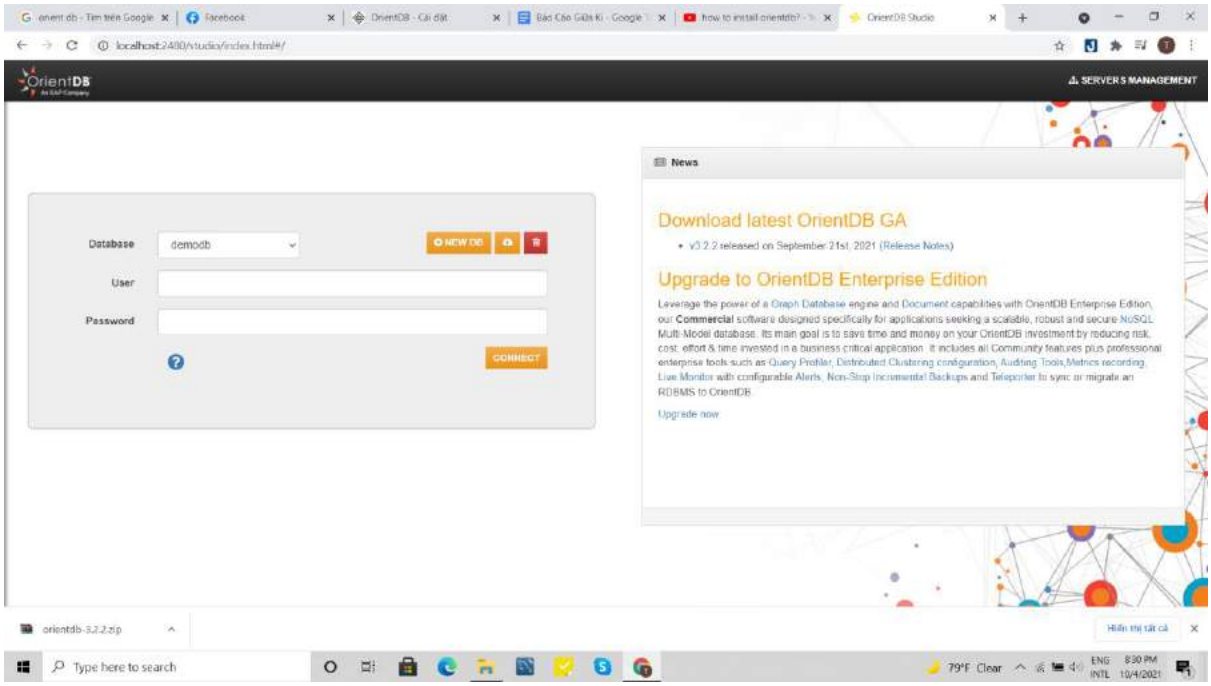
Hình 1.10. Màn hình nhập passWord cho DB root trong quá trình chạy file server.bat

Tiếp tục nhập "password" sau đó sẽ có 1 đường link dẫn đến màn hình console cho cơ sở dũ liệu của bạn như hình dưới đây:



Hình 1.11. Màn hình khi file server.bat chạy thành công và server bắt đầu run

Sau khi truy cập link ta được hình như dưới đây:



Hình 1.12. Giao diện của Orient Client trên nền web

CHƯƠNG II: TÌM HIỂU ORIENTDB

2.1 Mô hình dữ liệu (Data Model)

- Record
  Đơn vị nhỏ nhất được sử dụng và lưu trữ trong OrientDB.
  Record được phân biệt với nhau bằng RID (record id), có format
  `#<cluster>:<position>`
  Record có 4 hình thái, ứng với 4 loại model phổ biến trong NoSQL:
  - Document

Relational Model	Document Model	OrientDB Document Model
Table	Collection	Class or Cluster
Row	Document	Document
Column	Key/value pair	Document field
Relationship	not available	Link

  - BLOB (binary large object)
  - Vertex
  - Edge
- Class
  Ở trên, em có đề cập đến Class. Ý tưởng ở đây được lấy từ OOP, khi ta có thể tạo ra các Schema có liên kết với nhau.
- Cluster
  Trong 1 class, có nhiều tình huống mà ta muốn gom nhóm dữ liệu lại để xử lí cho những bài toán cụ thể. Ví dụ, với class Person, ta có :
  - Cluster “Cache" để lưu những record được truy cập nhiều nhất
  - Cluster “Today" để lưu những record được tạo ra hôm nay
  - …

  1 cluster có thể là Persistent Cluster (lưu vào ổ đĩa) hoặc In-Mermory Cluster (chỉ lưu trên ram). Theo em, ý tưởng này tương tự như tạo VIEW và MATERIALIZED VIEW trong RDBMS

2.2 Cú pháp truy vấn

- Thay vì tạo ra 1 ngôn ngữ truy vấn khác, OrientDB sử dụng câu lệnh SQL phổ biến, với 1 số tính năng bổ sung hỗ trợ cho các khái niệm phức tạp trong CSDL đồ thị, như Cây
- CREATE
  Ví dụ, ta muốn thêm 1 record Jay Miner thuộc class Employee
  OrientDB support 3 loại syntax để thực hiện việc đấy
  - ANSI-93

```
orientdb> INSERT INTO    Employee(name, surname, gender)
        VALUES('Jay', 'Miner', 'M')
```

➢ ANSI-92

```
orientdb>
INSERT INTO Employee SET name='Jay', surname='Miner', gender='M'
```

➢ JSON

```
orientdb> INSERT INTO Employee CONTENT {name : 'Jay', surname : 'Miner',
        gender : 'M'}
```

Nhờ sự đa dạng trên, ta có thể chọn syntax phù hợp với hoàn cảnh của mình

❖ READ

Ta có thể lấy ra toàn bộ record thuộc class OUser

```
orientdb> SELECT FROM OUser
```

Hoặc chỉ từ 1 cluster mong muốn

```
orientdb> SELECT FROM CLUSTER:Ouser
```

Hoặc thậm chí chỉ 1 / vài record, từ RID

```
orientdb> SELECT FROM #10:3
orientdb> SELECT FROM [#10:1, #10:30, #10:5]
```

Trong OrientDB, ta sẽ phải chỉ đích danh tên index nếu muốn sử dụng chúng

```
orientdb> SELECT VALUE FROM INDEX:dictionary WHERE key='Jay'
```

Các keyword hỗ trợ cho việc tìm kiếm vẫn như SQL thông thường (WHERE, GROUP BY, LIMIT, SKIP …)

```
orientdb> SELECT SUM(salary) FROM Employee WHERE age < 40 GROUP BY job
```

❖ UPDATE

Tương tự như INSERT, ta có thể lựa chọn viết theo SQL thuần hoặc format JSON

➢ ANSI-92

```
orientdb> UPDATE Employee SET local=TRUE WHERE city='London'
```

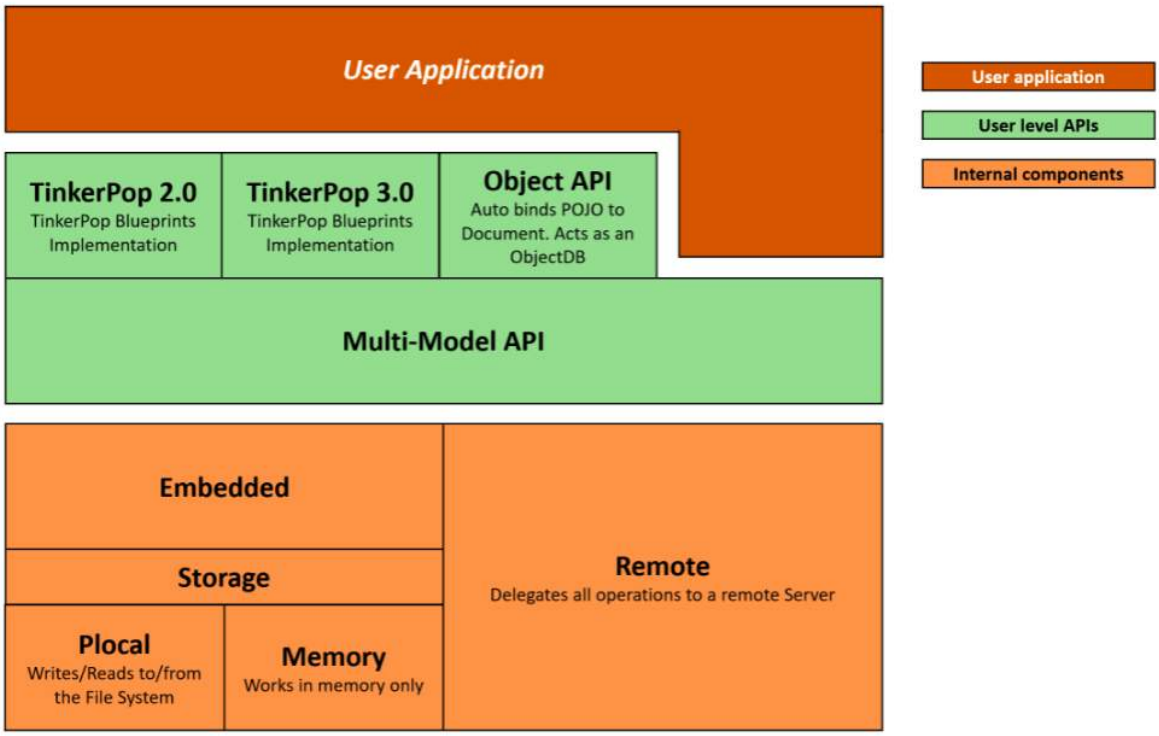
➢ JSON (keyword MERGE thay cho SET trong các câu truy vấn của MySQL)

```
orientdb> UPDATE Employee MERGE { local : TRUE } WHERE city='London'
```

❖ DELETE

```
orientdb> DELETE FROM Employee WHERE city <> 'London'
```

2.3 Kiến trúc



Hình 2.1: Kiến trúc phía dưới của OrientDB

- Index: có 5 loại index đã được OrientDB cung cấp
  - MVRB tree (deprecated từ bản 1.6)
    Đây là 1 thuật toán open-source, kết hợp sức mạnh của B+ tree và Red-Black tree, có khả năng hỗ trợ các giao dịch. MVRB tree có thời gian phản hồi với các thao tác insert, update và lookup nhanh. Tuy nhiên, không ổn định, có nhiều bug đã bị cộng đồng report
  - Hash Index
    Lookup rất nhanh, tốn rất ít bộ nhớ
  - Lucene Full Text Index
  - Lucene Spatial Index
  - SB tree
    Hiện là index mặt định của OrientDB, với khả năng dung hòa mặt tốt của các index trên. SB tree ổn định, hỗ trợ giao dịch, hỗ trợ truy vấn khoảng và hiệu năng cho các thao tác tốt
    Với các tree-based index khác, chi phí tìm kiếm cho N bản ghi thường là log2(N). Còn SB tree chỉ tốn logX(N), với X xấp xỉ 500

User cũng có thể tạo Custom Index Engine, dựa vào việc implement interface OIndexFactory hoặc interface OIndexEngine

- ➢ Với OIndexFactory
- ➢ Với OIndexEngine

❖ Storage Engine
Ở đây, ta xét tình huống đang ở máy server, và quan sát xem engine của OrientDB đã làm gì với dữ liệu
Có 2 lựa chọn là Memory, dữ liệu lưu trữ trên RAM. Tốc độ truy xuất rất nhanh, nhưng cũng đầy nguy hiểm

2.4 Scalability

❖ OrientDB có khả năng scale tuyệt vời, nhờ vào cơ chế phân tán multi-master (hoặc tên gọi khác là master-less). DBMS này đã sử dụng Hazelcast project cho việc tự động cấu hình và đồng bộ hóa giữa các node

❖ Nói thêm về Hazelcast, đây là 1 mạng lưu trữ dữ liệu trên RAM, được thiết kế rất nhẹ và dễ sử dụng, được viết bằng Java và không phụ thuộc bất kì thư viện nào khác. Không khó hiểu khi OrientDB lựa chọn công cụ này để thể hiện khả năng scale.
Hazelcast có khả năng mở rộng rất cao, khi lên tới hàng trăm hay hàng nghìn node. Tất cả các node chỉ cần duy trì 1 kết nối TCP và giao tiếp nội bộ thông qua tầng này
Hazelcast lưu trữ mọi thứ trên RAM, vì thế tốc độ thực thi rất nhanh cho các thao tác đọc ghi
Hazelcast có khả năng chịu lỗi tốt, khi lưu trữ các bản sao của mỗi mảnh dữ liệu node này trên các node khác. Khi 1 node bị lỗi, dữ liệu sẽ được khôi phục bằng việc chắp vá từ các bản sao, cụm các node vẫn hoạt động bình thường mà không hề có downtime

❖ OrientDB cũng cho phép sharding và replication như những NoSQL DBMS khác

❖ OrientDB hỗ trợ distributed transaction (giao dịch phân tán). Khi 1 giao dịch được commit, thông tin mới sẽ được gửi qua toàn bộ các node trên server, và yêu cầu các node đó phải có trách nhiệm commit cho transaction. Nếu có ít nhất 1 node đang mất kết nối, quorum sẽ được kiểm tra. Dựa vào quorum, thì toàn bộ các node sẽ commit hoặc toàn bộ rollback
Khi thực hiện, OrientDB vẫn sử dụng giao thức 2 phase commit, chỉ là với niềm tin mọi thứ sẽ “optimistic" (tức sẽ hoạt động trơn tru theo kì vọng, dựa vào những ràng buộc từ thuật toán). Vậy nên, thay vì phải đợi phản hồi từ tất cả các node, và khoá data trong suốt khoảng thời gian đợi phản hồi của phase 1, ta chỉ đặt trước 1 mốc hẹn, và mọi thứ được xử lí tại phase 2
Trong khi commmit, OrientDB sẽ bắt đầu khóa và kiểm tra lại version những dữ liệu (optimistic Multi Version Concurrency Control). Sẽ có 3 tình huống có thể xảy ra:
  - ➢ Toàn bộ data có thể khóa và không ai đụng đến kể từ thời điểm bắt đầu giao dịch. Mọi thứ suôn sẻ
  - ➢ Có ai đó chỉnh sửa dữ liệu cùa giao dịch này. Giao dịch thất bại, người dùng có thể thử lại sau
  - ➢ Có 1 giao dịch khác khóa vào dữ liệu của giao dịch này. Giao dịch thất bại, nhưng có cơ chế để có thể tự động retry

WriteQuorum của OrientDB là “majority", tức đa số đúng. Nếu có 5 server, cùng thực hiện giao dịch, 3 tình huống có thể xảy ra:

- ➢ Toàn bộ 5 server đều có thể commit. Mọi thứ suôn sẻ
- ➢ 1 hoặc 2 server bị lỗi. Giao dịch vẫn được tiếp tục (do đa số là 3), những server bị lỗi sẽ phải cập nhật kết quả lại theo số đông
- ➢ 3 server trở lên bị lỗi. Giao dịch thất bại, các node được rollback về giá trị trước khi thực hiện commit

2.5 Use Case: Time Series

- ❖ Với những app phục vụ cho lượng lớn khách hàng, lượng log cần lưu trữ cho các mục đích khác nhau là rất lớn, hàng triệu bản ghi trong 1 bảng là chuyện rất bình thường. Và khi có chuyện xảy ra trong ngày 24/08/2021, ta sẽ làm như thế nào để lấy ra các log trong ngày hôm đấy ?
  Ví dụ về 1 bản ghi:

```
{
  "date" : 12293289328932,
  "priority" : "critical",
  "note" : "System reboot"
}
```

- ❖ Nếu sử dụng những RDBMS như MySQL, việc ta cần làm là quét qua toàn bộ các bản ghi, kiểm tra điều kiện, rồi lặt ra 1 lượng nhỏ các bản ghi phù hợp. Ngay cả khi sử dụng index, thì số bản ghi cần truy cập cũng xấp xỉ log2(N), với N là tổng số bản ghi
- ❖ Thay vì làm như trên, ta có thể gom nhóm dữ liệu theo mô hình DAG (directed acyclic graph, đồ thị có hướng) như sau:
  Year -> Month -> Day -> Hour -> minute -> second -> millisecond
- ❖ Coi Year, Month, Day … là các đỉnh của đồ thị, ta chỉ việc đi qua các cạnh nối (như reference giữa các bảng trong RDBMS), và lấy ra 1 lượng dữ liệu thỏa mãn. Cách xử lí range query như thế này sẽ tối ưu hơn rất nhiều
- ❖ Sau đây là code demo cho use case trên
  Phần khởi tạo mô hình dữ liệu:

```
CREATE CLASS Year extends V
CREATE CLASS Month extends V
CREATE CLASS Day extends V
CREATE CLASS Hour extends V
CREATE CLASS Log extends V

CREATE PROPERTY Year.value STRING
CREATE PROPERTY Year.month LINKMAP Month
CREATE PROPERTY Month.day LINKMAP Day
CREATE PROPERTY Day.hour LINKMAP Hour
CREATE PROPERTY Hour.log LINKSET Log
```

  Truy vấn lấy ra các log ứng với thời điểm 24/08/2021, lúc 10h sáng:

```
SELECT EXPAND(month[8].day[24].hour[10].log) FROM Year WHERE value='2021'
```

- Chú thích:
  - V: database lưu trữ các vertex (đỉnh), đã được khởi tạo từ đầu bởi OrientDB
  - Class: tựa như Schema, đã giải thích ở phần 2.1
  - Property: là edge (cạnh), nêu lên mối quan hệ giữa các vertex
  - LinkMap: 1 map với key là các string, không chấp nhận key trùng lặp
  - LinkSet: 1 set với các phần tử không trùng lặp

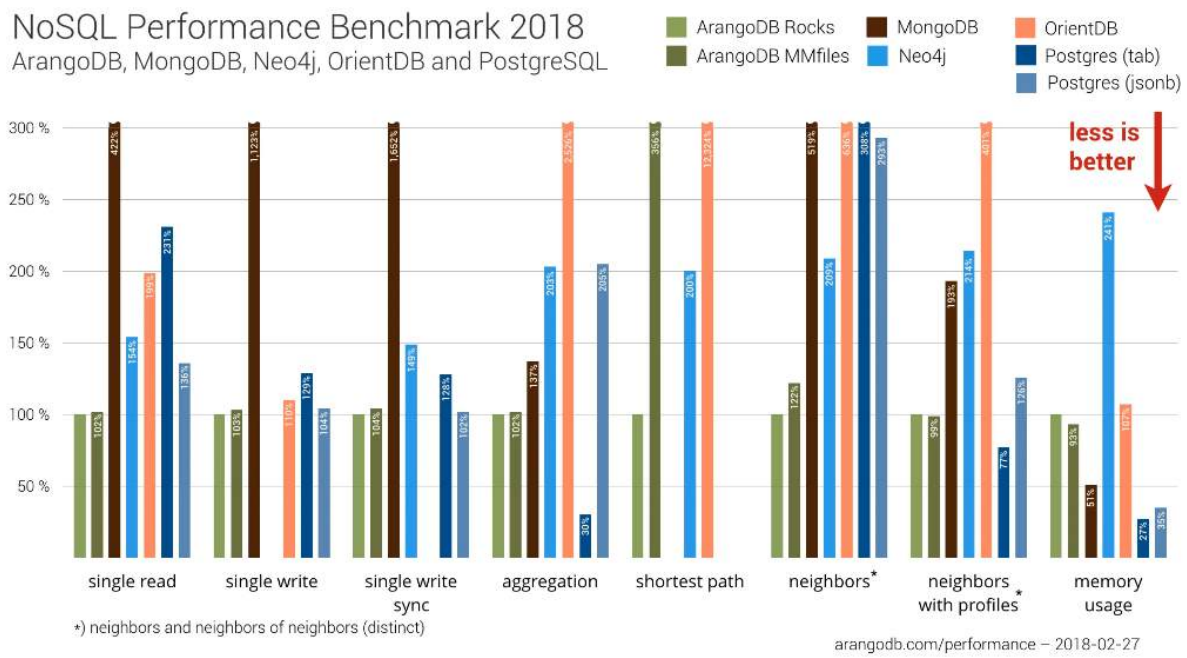
CHƯƠNG III: SO SÁNH ORIENTDB

3.1 Với MongoDB, Arrango và Neo4j

- Cấu hình được sử dụng cho bài test hiệu năng
  - Server: i3.4xlarge trên AWS với 16 nhân ảo, 122 GB RAM
  - Client: c3.xlarge trên AWS với 4 nhân CPU, 7.5 GB RAM, 40 GB SSD
- Các DBMS được sử dụng cho bài test
  - Neo4j 3.3.1
  - MongoDB 3.6.1
  - PostgreSQL 10.1
  - OrientDB 2.2.29
  - ArangoDB 3.3.3
- Sau đây là các trường hợp đã được sử dụng cho bài test

- **single-read:** these are single document reads of profiles (i.e., 100,000 different documents).
- **single-write:** these are single document writes of profiles (i.e., 100,000 different documents).
- **single-write sync:** these are the same as single-writes, but we waited for fsync on every request.
- **aggregation:** these are ad-hoc aggregation over a single collection (i.e., 1,632,803 documents). We computed the age distribution for everyone in the network, simply counting how often each age occurs.
- **neighbors second:** we searched for distinct, direct neighbors, plus the neighbors of the neighbors, returning ID's for 1,000 vertices.
- **neighbors second with data:** we located distinct, direct neighbors, plus the neighbors of the neighbors and returned their profiles for 100 vertices.
- **shortest path:** this the 1,000 shortest paths found in a highly connected social graph. This answers the question how close to each other two people are in the social network.
- **memory:** this is the average of the maximum main memory consumption during test runs.

- Và đây là kết quả của bài test



Hình 3.1. Kết quả hiệu năng của các cơ sở dữ liệu NoSQL

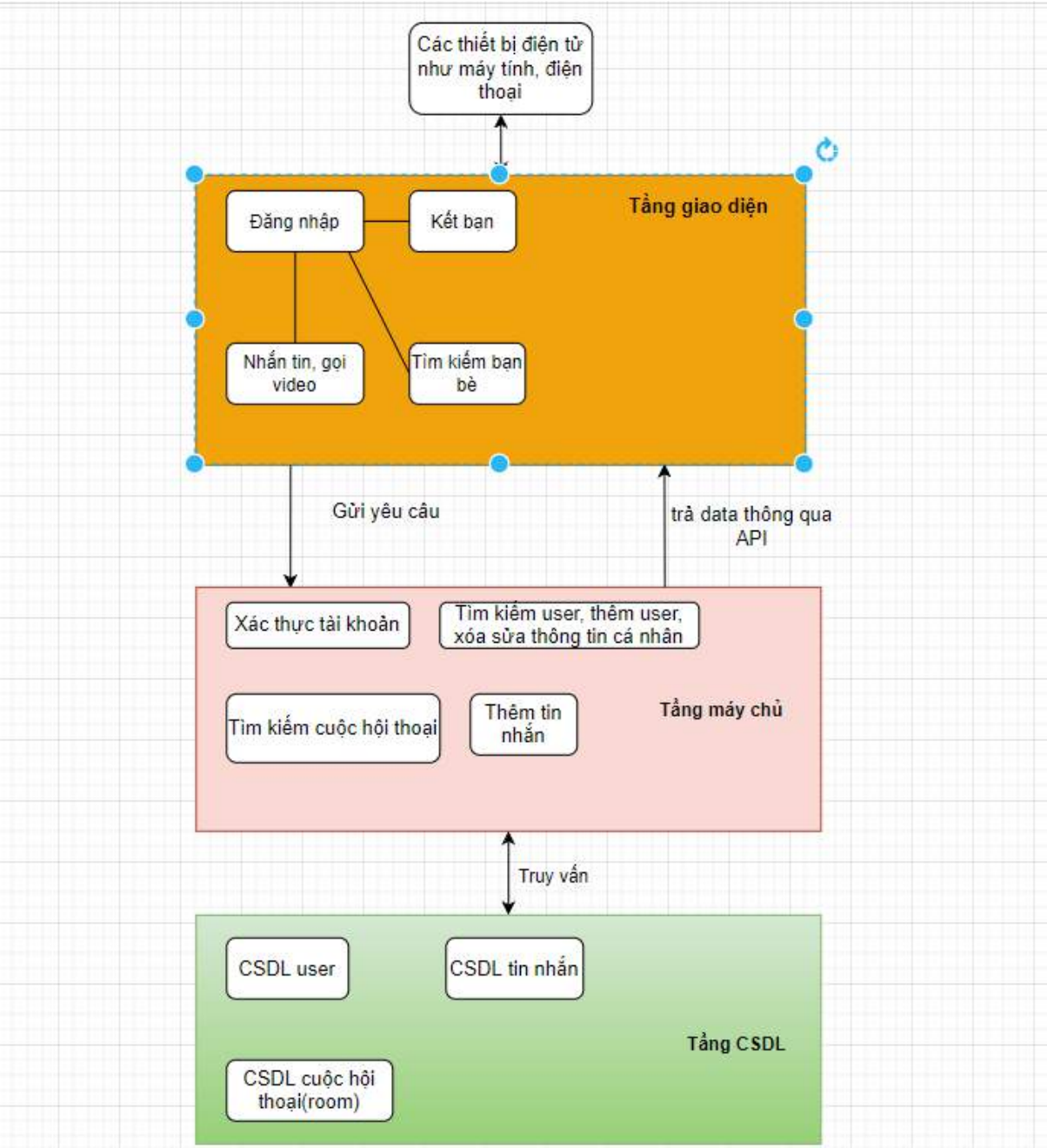
- Như chúng ta thấy
  - Với các operation cơ bản và phần lớn được sử dụng như single read và single write, OrientDB có tốc độ vượt trội so với MongoDB, vầ xấp xỉ trong các trường hợp còn lại. Chỉ có tình huống aggregation, là MongoDB thắng lợi tuyệt đối
  - Hiệu năng của OrientDB và Neo4j là xấp xỉ trong các test case
  - Còn ArrangoDB đã chứng minh sức mạnh ưu việt của mình so với các đối thủ còn lại

3.2 Lí do cho sự thất thế

- Là 1 trong những dự án đời đầu của thế hệ NoSQL, OrientDB gặp nhiều khó khăn với tham vọng của mình. Họ hướng đến 1 sản phẩm phục vụ quá nhiều nhu cầu. Chạy theo các tính năng mới, mà bỏ qua việc hoàn thiện các tính năng đã có, làm cho OrientDB luôn ở trong tình trạng thiếu ổn định và nhiều bug
- Không thân thiện với người dùng. Cộng đồng bé, lại không có tài liệu đầy đủ chi tiết, những người mới gặp rất nhiều khó khăn trong quá trình sử dụng và đưa sản phẩm vào dự án. Họ chuyển dịch sang những DBMS khác với hệ sinh thái đầy đủ như MongoDB. Câu chuyện con gà - quả trứng lại từ đó mà sinh ra

CHƯƠNG IV: KIẾN TRÚC CHAT APP

4.1 Tổng quan hệ thống



Hình 4.1. Ảnh kiến trúc hệ thống

4.1.1 Tầng giao diện

Tầng giao diện bao gồm các thiết bị người dùng cuối như máy tính, thiết bị Android, iOS. Tầng này quản lý, thực hiện việc hiển thị các thông tin đến người dùng. Người dùng có thể thực hiện các chức năng, yêu cầu đến tầng máy chủ thông qua RESTful API để lấy thông tin người dùng , tin nhắn, cuộc hội thoại,....

4.1.2 Tầng máy chủ

Tầng máy chủ web service có nhiệm vụ xử lý các vấn đề về logic, thuật toán, giải thuật, xử lý truy vấn, thao tác trực tiếp với CSDL, ... Thêm vào đó, trong bài toán này, tầng máy chủ sử dụng socket.io một module thực hiện kết nối ngay lập tức giữa Client và Server để có thể xây dựng ứng dụng Real-time.

4.1.3 Tầng CSDL

Tầng CSDL là tầng không thể thiếu trong bất cứ hệ thống lớn nhỏ nào. Chúng đảm nhiệm việc lưu trữ, ghi, đọc, sửa, xóa cơ sở dữ liệu. Đối với ứng dụng này, hệ thống sử dụng cơ sở dữ liệu phi quan hệ là MongoDB.

4.2 Các công nghệ dự kiến sử dụng

4.2.1. Front-End

- **FrameWork**: ReactJs dùng để xây dựng giao diện người dùng (UI). React được sử dụng rộng rãi và có hệ sinh thái đa dạng phong phú.
- **Library:** Một thư viện dùng để quản lý các State của ứng dụng

4.2.2 Back-End

- NodeJs/ Express: Expressjs là một framework được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. Expressjs hỗ trợ các method HTTP và midleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.
- Socket.io: Socket.io là một module trong Node.js được phát triển vào năm 2010. Nó được phát triển để sử dụng các kết nối mở để tạo điều kiện giao tiếp thời gian thực, trả về giá trị thực ở tại thời điểm đó. Socket.io cho phép giao tiếp hai chiều giữa máy khách và máy chủ.Nó được sử dụng trong việc xây dựng các ứng dụng web real-time cần tốc độ phản hồi ngay lập tức như: chat, trực tiếp bóng đá,....

4.2.3 CSDL: MongoDB

MongoDB là một database hướng tài liệu (document), một dạng NoSQL database. Vì thế, MongoDB sẽ tránh cấu trúc table-based của relational database để thích ứng với các tài liệu như JSON có một schema rất linh hoạt gọi là BSON. MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ các các kích cỡ và các document khác nhau. Các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

4.3 Các chức năng dự kiến

4.3.1 Các yêu cầu chức năng

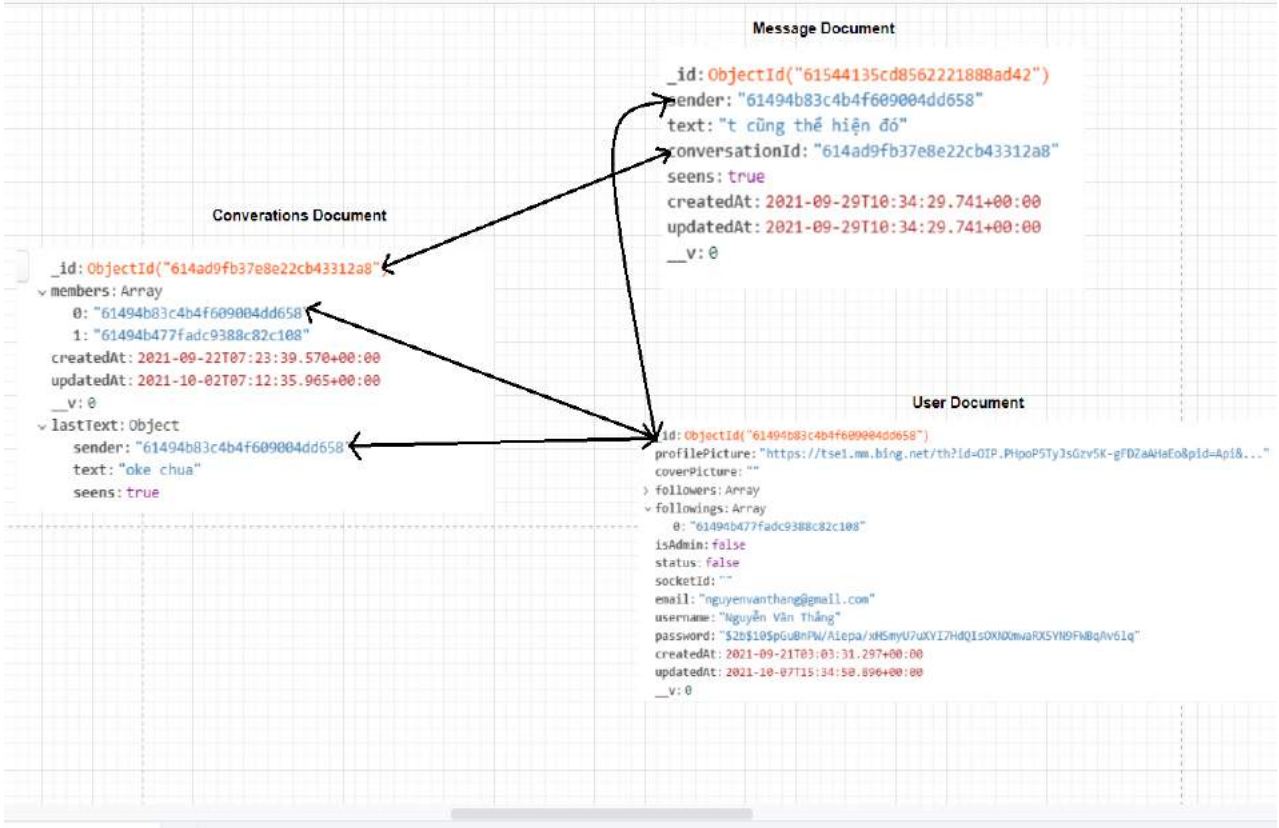
- Đăng ký
- Đăng nhập
- Nhắn tin (text, image, video,...)
- Call video, Call audio
- Tìm kiếm bạn bè
- Kết bạn
- Xem thông tin bạn bè, cá nhân
- Sửa thông tin cá nhân

4.3.2 Các yêu cầu phi chức năng

- Tin nhắn được gửi đi thì người nhận phải lập tức nhận được
- Khi gọi video hay audio thì âm thanh hình ảnh truyền theo đúng thời gian thực

4.4 Thiết kế CSDL

4.4.1 Lược đồ cơ sở dữ liệu



Hình 4.2. Hình ảnh minh họa lược đồ cơ sở dữ liệu

4.4.2 Thông tin user Document

Trường	Mô tả	Kiểu dữ liệu
_id	id người dùng	objectId
profilePicture	ảnh người dùng	String
followers	Người theo dõi	Array
followings	Người đang theo dõi	Array
status	Trạng thái đăng nhập	Boolean
socketId	Socket Id	String
email	email người dùng	String
username	tên người dùng	String
password	mật khẩu người dùng	String
createdAt	Ngày tạo nick	Date
updatedAt	Ngày chỉnh sửa thông tin	Date

Bảng 4.4.2.1: Thông tin user Document

4.4.3 Thông tin Conversations Document

Trường	Mô tả	Kiểu dữ liệu
_id	id cuộc hội thoại	objectId
member	Thành viên trong cuộc hội thoại	Array
createdAt	Ngày tạo cuộc hội thoại	Date
updatedAt	Ngày chỉnh sửa thông tin cuộc hội thoại	Date
lastText	Tin nhắn cuối cùng của cuộc hội thoại	Object

Bảng 4.4.3.1: Thông tin Conversations Document

4.4.4 Thông tin bảng Message Document

Trường	Mô tả	Kiểu dữ liệu
_id	id tin nhắn	objectId
sender	người gửi	String
text	tin nhắn	String
conversationId	id cuộc hội thoại	String
seens	đã xem tin nhắn hay chưa	Boolean
createdAt	Ngày tin nhắn lần đầu tiên được nhắn	Date
updatedAt	Ngày tin nhắn cuối cùng được nhắn	Date

Bảng 4.4.4.1: Thông tin Message Document

THAM KHẢO

- **http://www.orientdb.com/docs/last/index.html**
- **https://www.tutorialspoint.com/orientdb/index.htm**
- https://www.arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/
- https://socket.io/docs/v4/
- https://docs.mongodb.com/manual/tutorial/getting-started/