Backend

1. Các layers của backend thường bao gồm:

Web server layer: Layer này giúp xử lý các request đến từ client thông qua giao thức HTTP hoặc HTTPS. Web server có nhiệm vụ nhận và phản hồi các request của client chuyển tới các ứng dụng hoặc API phù hợp. Ví dụ: Apache, Nginx …

Application layer: Layer này đảm nhiệm xử lý các request từ web server và trả về kết quả cho client. Nhiệm vụ của layer này là xử lý tất cả các logic của hệ thống, bao gồm xử lý dữ liệu, truy vấn database và trả về kết quả cho client.

Database layer: Layer này bao gồm các hệ quản trị cơ sở dữ liệu như MySQL, PostgreSQL hoặc MongoDB. Nhiệm vụ của layer này là lưu trữ và quản lý dữ liệu của ứng dụng, đảm bảo tính nhất quán và độ tin cậy của dữ liệu.

Storage layer: Layer này bao gồm các hệ thống lưu trữ tập tin, dữ liệu như Amazon S3, Google Cloud Storage hay Dropbox. Layer này đảm bảo rằng dữ liệu được lưu trữ an toàn và có khả năng mở rộng khi hệ thống mở rộng.

2.

Trong cơ sở dữ liệu, session là một phiên làm việc giữa backend và cơ sở dữ liệu. Nó bao gồm một loạt các thao tác đối với database, ví dụ như query, insert, update dữ liệu. Session được bắt đầu khi ứng dụng backend kết nối đến cơ sở dữ liệu và kết thúc khi kết nối bị đóng.

Các lợi ích của việc sử dụng session trong database bao gồm:

Quản lý transaction: Session cung cấp các transaction để đảm bảo tính toàn vẹn của dữ liệu trong quá trình cập nhật. Nó cho phép các thao tác trên cơ sở dữ liệu được nhóm lại và thực hiện một cách an toàn (ACID), đồng thời đảm bảo rằng dữ liệu sẽ không bị mất hoặc bị hỏng trong quá trình xử lý.

Quản lý locking: Session cũng cho phép quản lý các lock, đảm bảo rằng các truy vấn trên cùng một bảng sẽ không gây ra xung đột hoặc xóa bỏ dữ liệu của nhau.

Tối ưu hóa performance: Session có thể giúp tối ưu hóa performance bằng cách sử dụng connection pooling hoặc statement caching. Connection pooling cho phép sử dụng lại các kết nối đã mở, giúp giảm thời gian kết nối và nguy cơ gây ra tình trạng số lượng kết nối đến cơ sở dữ liệu quá lớn. Statement caching giúp lưu trữ lại các truy vấn đã thực thi trước đó để sử dụng lại trong các lần sau, giúp giảm thời gian truy vấn và tăng hiệu suất hệ thống.

3.

Triển khai docker: bao gồm 2 service

- **etl** (bao gồm môi trường và các dependency dành cho python)
- **db** (gồm database postgres và 1 file init db để tạo bảng lúc build container)

Data engineer

1.

Columnar database (còn gọi là column-store database) và row-oriented database (còn gọi là row-store database) là hai kiểu cơ sở dữ liệu có cấu trúc lưu trữ khác nhau. Các khác nhau chính giữa chúng là cách lưu trữ dữ liệu trên đĩa.

Trong row-oriented database, các bản ghi được lưu trữ theo từng hàng liên tiếp nhau trên đĩa. Mỗi bản ghi chứa tất cả các cột của dữ liệu trong hàng đó. Điều này làm cho việc truy vấn dữ liệu từ các bảng với số lượng hàng lớn trở nên chậm do phải truy xuất tất cả các cột của tất cả các hàng, bất kể chúng có được sử dụng trong truy vấn hay không.

Trong columnar database, các cột của các bảng được lưu trữ dưới dạng các file riêng lẻ trên đĩa. Mỗi file chứa giá trị của một cột cho tất cả các hàng. Điều này làm cho việc truy vấn dữ liệu nhanh hơn bởi vì chỉ cần đọc các cột cần thiết trong truy vấn, không cần phải đọc toàn bộ bảng.

Các lợi ích của columnar database bao gồm:

- Tốc độ truy vấn nhanh hơn cho các truy vấn với số lượng hàng lớn.
- Hiệu quả khi phân tích dữ liệu và tính toán số liệu thống kê.
- Hiệu quả khi sử dụng các dịch vụ lưu trữ đám mây.

Tuy nhiên, columnar database cũng có một số hạn chế:

- Tốn nhiều không gian lưu trữ hơn so với row-oriented database.
- Thời gian cập nhật dữ liệu chậm hơn, đặc biệt là khi cần phải thêm hoặc xóa hàng mới.

2.

Distributed và non-distributed database là hai loại database khác nhau dựa trên cách chúng được phân phối trên các nodes hoặc servers.

Non-distributed database là một database được lưu trữ trên một server duy nhất. Tất cả các truy vấn và thay đổi dữ liệu đều được thực hiện trên server này. Các ví dụ về non-distributed database bao gồm SQLite, MySQL, Oracle, SQL Server.

Trong khi đó, distributed database là một database được phân tán trên nhiều server hoặc nodes, mỗi node có thể chứa một phần hoặc toàn bộ dữ liệu. Việc phân phối này giúp tăng tính sẵn sàng, khả năng chịu lỗi và hiệu suất của hệ thống. Khi truy vấn hoặc thay đổi dữ liệu, hệ thống sẽ phân bổ tác vụ cho các node khác nhau. Các ví dụ về distributed database bao gồm Cassandra, Hadoop, MongoDB.

Một số khác nhau khác giữa distributed và non-distributed database bao gồm:

- Tính sẵn sàng và khả năng chịu lỗi: Distributed database cung cấp khả năng chịu lỗi cao hơn so với non-distributed database. Vì dữ liệu được phân tán trên nhiều node, nếu một node bị lỗi, các node khác vẫn có thể tiếp tục hoạt động.
- Hiệu suất: Distributed database có thể cung cấp hiệu suất tốt hơn cho các ứng dụng có tải lớn. Vì dữ liệu được phân tán trên nhiều node, các tác vụ truy vấn và thay đổi dữ liệu có thể được phân bổ cho các node khác nhau để tăng tốc độ xử lý.
- Chi phí: Distributed database có chi phí cao hơn so với non-distributed database, vì cần phải có nhiều server hoặc node để phân phối dữ liệu và xử lý các tác vụ. Ngoài ra, việc quản lý và vận hành cũng phức tạp hơn.

3.

Các package đã được sử dụng:

- Pandas: load dữ liệu từ file excel lên dataframe
- Openpyxl: để sử dụng hàm read_excel của pandas
- Psycopg2: sử dụng để kết nối với database và query

Các bước xử lý:

- Load file excel lên bằng hàm read_excel vào dataframe
- Chia nhỏ dataframe thành các chunks
- Loop các chunks để insert record vào bảng StockPrice
- Insert dữ liệu kết hợp với aggregate dữ liệu từ bảng StockPrice 1 phút sang bảng StockPriceHour dữ liệu theo giờ

Một pipeline ETL cơ bản cần có những logic sau:

- Extract: Trích xuất dữ liệu từ nguồn dữ liệu (như database, file, API...) để chuẩn bị cho bước tiếp theo.
- Transform: Chuyển đổi và định dạng dữ liệu để nó phù hợp với mô hình dữ liệu đích và được xử lý dễ dàng hơn. Trong giai đoạn này, các phép biến đổi dữ liệu có thể được áp dụng, chẳng hạn như lọc, sắp xếp, ghép nối, đánh giá, tổng hợp và tính toán dữ liệu mới.
- Load: Lưu trữ dữ liệu đã được xử lý vào hệ thống đích (như database, data warehouse, data lake...).

4.

Nếu dữ liệu tăng gấp 1000 lần thì tốc độ sao lưu cũng sẽ chậm đi đáng kể.

Để tối ưu hiệu suất sao lưu dữ liệu, có thể áp dụng một số kỹ thuật như:

- Sử dụng một cấu trúc phân tán để tăng tốc độ sao lưu. Chia file và lưu dữ liệu trên nhiều node
- Tăng tốc độ đọc và ghi dữ liệu
- Thay vì sử dụng excel thì sử dụng csv để tăng hiệu suất đọc file