Tài liệu Hướng dẫn Cài đặt và Chạy SparkSQL với SQLite trong Docker

## Tổng quan

Tài liệu này cung cấp hướng dẫn từng bước về cách thiết lập môi trường Docker, kết nối SparkSQL với cơ sở dữ liệu SQLite, và thực hiện các thao tác CRUD cùng với các bài kiểm tra hiệu suất. Các phần sau đây sẽ bao gồm quá trình cài đặt, hướng dẫn xây dựng và chạy container Docker, và các ví dụ về truy vấn SQL.

## Yêu cầu

Đã cài đặt Docker trên máy của bạn

Đã cài đặt Python và PySpark

Đã cài đặt SQLite

A screen shot of a computer

Description automatically generated

Yêu cầu tạo thư mục có directory tree như hình.

## Quá trình cài đặt

### 1. Thiết lập Môi trường Docker

Để tạo và chạy một container Docker với hỗ trợ SparkSQL và SQLite, làm theo các bước sau:

Sử dụng `Dockerfile` đã cung cấp để xây dựng Docker image. `Dockerfile` được thiết kế như sau:

Giải thích các lệnh trong Dockerfile

FROM openjdk:11-jre-slim

Lệnh này xác định hình ảnh cơ sở (base image) là OpenJDK 11 với phiên bản JRE slim. Đây là một hình ảnh nhẹ, phù hợp để chạy các ứng dụng Java mà không cần môi trường phát triển đầy đủ.

ENV SPARK\_VERSION=3.5.1

Đặt biến môi trường SPARK\_VERSION là 3.5.1, đây là phiên bản của Apache Spark mà chúng ta sẽ sử dụng.

ENV HADOOP\_VERSION=3

Đặt biến môi trường HADOOP\_VERSION là 3, đây là phiên bản của Hadoop mà chúng ta sẽ sử dụng cùng với Spark.

ENV SCALA\_VERSION=2.13

Đặt biến môi trường SCALA\_VERSION là 2.13, đây là phiên bản của Scala mà Spark sẽ sử dụng.

RUN apt-get update && apt-get install -y curl python3 python3-pip sqlite3 procps && apt-get clean

Lệnh này cập nhật danh sách các gói có sẵn và cài đặt các gói cần thiết: curl, python3, python3-pip, sqlite3, và procps. Sau khi cài đặt xong, lệnh apt-get clean sẽ xóa bỏ các tệp tin không cần thiết để giảm dung lượng của hình ảnh Docker.

RUN pip3 install JPype1 JayDeBeApi pyspark

Lệnh này cài đặt các thư viện Python cần thiết thông qua pip3, bao gồm JPype1, JayDeBeApi và pyspark. Các thư viện này cần thiết để kết nối và làm việc với cơ sở dữ liệu SQLite từ Spark.

RUN wget https://archive.apache.org/dist/spark/spark-${SPARK\_VERSION}/spark-${SPARK\_VERSION}-bin-hadoop${HADOOP\_VERSION}.tgz && \

tar -xzf spark-${SPARK\_VERSION}-bin-hadoop${HADOOP\_VERSION}.tgz && \

mv spark-${SPARK\_VERSION}-bin-hadoop${HADOOP\_VERSION} /opt/spark && \

rm spark-${SPARK\_VERSION}-bin-hadoop${HADOOP\_VERSION}.tgz

Lệnh này tải và giải nén thư mục spark-3.5.1-bin-hadoop3-scala2.13 vào thư mục /opt/spark trong container. Đây là nơi chứa các tệp tin của Apache Spark.

ENV SPARK\_HOME=/opt/spark

Đặt biến môi trường SPARK\_HOME trỏ tới thư mục Spark tại /opt/spark.

ENV PATH=$SPARK\_HOME/bin:$PATH

Thêm thư mục bin của Spark vào biến môi trường PATH để có thể gọi các lệnh Spark từ bất kỳ đâu trong container.

COPY sqlite-jdbc-3.34.0.jar /opt/sqlite-jdbc-3.34.0.jar

Lệnh này sao chép tệp tin SQLite JDBC driver vào thư mục /opt trong container. JDBC driver này cần thiết để kết nối Spark với cơ sở dữ liệu SQLite.

COPY setup\_db.py /opt/setup\_db.py

Lệnh này sao chép tệp tin setup\_db.py vào thư mục /opt trong container. Tệp tin này sẽ được sử dụng để khởi tạo cơ sở dữ liệu SQLite.

COPY run\_crud.py /opt/run\_crud.py

Lệnh này sao chép tệp tin run\_crud.py vào thư mục /opt trong container. Tệp tin này sẽ được sử dụng để thực hiện các thao tác CRUD với SparkSQL.

EXPOSE 4040 7077 8080 18080

Lệnh này khai báo các cổng mà container sẽ lắng nghe. Các cổng này bao gồm:

- 4040: Spark UI

- 7077: Spark Master

- 8080: Spark Worker

- 18080: Lịch sử ứng dụng Spark

ENTRYPOINT ["tail", "-f", "/dev/null"]

Lệnh này đặt điểm nhập (entrypoint) của container là lệnh `tail -f /dev/null`. Điều này giữ cho container chạy liên tục để có thể thực hiện các thao tác khác bên trong container sau khi nó được khởi động.

### Xây dựng Hình ảnh Docker

Chạy lệnh sau trong thư mục chứa Dockerfile để xây dựng Docker image:

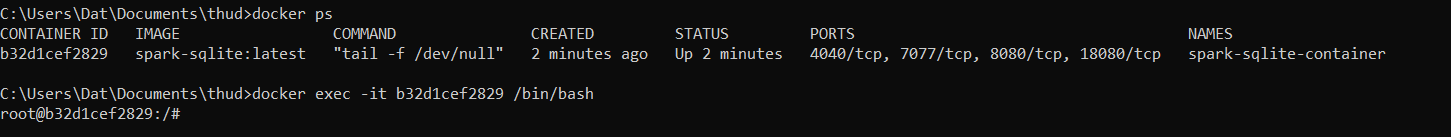
***docker build -t spark-sqlite:latest .***

Chạy container Docker với lệnh sau:

***docker run -d --name spark-sqlite-container spark-sqlite:latest***

***docker exec -it spark-sqlite-container /bin/bash***

Sau khi chạy 2 lệnh trên, ta sẽ được đưa vào bên trong docker có dạng như sau:



## Thiết lập Cơ sở dữ liệu

Đảm bảo cơ sở dữ liệu được thiết lập với lược đồ và dữ liệu ban đầu cần thiết. Sử dụng tập tin `setup\_db.py` để khởi tạo cơ sở dữ liệu SQLite.

### 1. Import các thư viện cần thiết:

sqlite3: Thư viện để làm việc với cơ sở dữ liệu SQLite.

random: Thư viện để sinh ra các giá trị ngẫu nhiên.

### 2. Hàm initialize\_db():

Đây là hàm chính để khởi tạo cơ sở dữ liệu. Hàm này thực hiện các bước tạo bảng và chèn dữ liệu mẫu vào.

### 3. Kết nối đến cơ sở dữ liệu SQLite:

Kết nối đến cơ sở dữ liệu SQLite và tạo một con trỏ (cursor) để thực hiện các thao tác SQL.

### 4. Tạo bảng people:

Tạo bảng people với các cột: id, name, age, và country.

### 5. Danh sách các quốc gia có thể:

Tạo một danh sách các quốc gia để chèn ngẫu nhiên vào bảng people.

### 6. Chèn dữ liệu mẫu:

Vòng lặp này tạo ra 1,000,000 bản ghi với tên ngẫu nhiên, tuổi ngẫu nhiên từ 1 đến 100, và quốc gia ngẫu nhiên từ danh sách countries.

### 7. Lưu và đóng kết nối:

Lưu các thay đổi vào cơ sở dữ liệu và đóng kết nối.

Khi vào bên trong docker, ta sử dụng lệnh sau để tạo và điền vào database 1 triệu dòng:

***python3 /opt/setup\_db.py***

A screen shot of a computer program

Description automatically generated

Khi thấy thông báo này, database đã được hoàn thành, ta có thể kiểm tra bằng cách đếm số lượng dòng trong database:

A screen shot of a computer

Description automatically generated

Chúng ta có thể thấy số lượng dòng trong database. Sau đó sử dụng lệnh sau để cấp các quyền cho phép truy cập chỉnh sửa database:

***chmod 777 /opt/spark/examples/src/main/resources/people.db***

## Chạy các thao tác CRUD của SparkSQL

Tập tin run\_crud.py thực hiện các thao tác CRUD và kiểm tra hiệu suất trên cơ sở dữ liệu SQLite bằng cách sử dụng SparkSQL.

File run\_crud.py chứa các hàm để thực hiện các thao tác CRUD (Create, Read, Update, Delete) với SparkSQL và cơ sở dữ liệu SQLite.

### 1. Cấu hình logging:

Cấu hình logging để ghi lại thông tin trong quá trình thực hiện các thao tác.

### 2. Hàm create\_spark\_session():

Tạo một SparkSession để làm việc với Spark. Đồng thời, cấu hình Spark để sử dụng JDBC driver cho SQLite.

### 3. Hàm connect\_to\_db():

Kết nối đến cơ sở dữ liệu SQLite và trả về kết nối.

### 4. Hàm create\_temp\_view():

Tạo một temporary view từ bảng people trong cơ sở dữ liệu SQLite để có thể thực hiện các truy vấn SQL với Spark.

### 5. Hàm read\_data():

Đọc và hiển thị 20 dòng dữ liệu đầu tiên từ bảng people.

### 6. Hàm insert\_data():

Chèn thêm hai bản ghi mới vào bảng people.

### 7. Hàm update\_data():

Cập nhật tuổi của bản ghi có id = 1000001 thành 40.

### 8. Hàm delete\_data():

Xóa các bản ghi có id >= 1000001 bằng cách lọc và ghi đè lại bảng people.

### 9. Hàm performance\_test():

Đo lường thời gian thực hiện các truy vấn đếm tổng số bản ghi và đếm số bản ghi có age > 50.

### 10. Hàm show\_bottom\_rows():

Hiển thị 5 bản ghi cuối cùng của bảng people theo thứ tự giảm dần của id.

Sử dụng lệnh sau:

***python3 /opt/run\_crud.py***

để chạy file run\_crud. Sau đây là kết quả khi chạy file:

A black background with white text

Description automatically generated

Thông báo về việc đã kết nối database và tạo temporary view.

A screenshot of a computer screen

Description automatically generated

Thông báo về việc đọc và in ra thông tin ở 5 dòng cuối.

A screen shot of a computer

Description automatically generated

Thông báo thêm thông tin và in 5 dòng cuối sau khi thêm thông tin.

A computer screen with white text

Description automatically generated

Thông báo về việc update thông tin về tuổi của id 1000001 và hiện thị thông tin sau thay đổi.

A computer screen shot of a black screen

Description automatically generated

Xóa 2 dòng cuối đã thêm vào và in ra 5 dòng cuối để kiểm tra.

A screen shot of a computer code

Description automatically generated

Kiểm tra thời gian count toàn bộ record và count có điều kiện WHERE và đóng kết nối server với database.