**TÀI LIỆU HƯỚNG DẪN THỰC HÀNH**

**Tên bài thực hành:**

**Cài đặt và triển khai Spark SQL**

**10/2023**

**MỤC LỤC**

[1. CƠ SỞ LÝ THUYẾT 2](#_Toc147408149)

[1.1. Giới thiệu chung 2](#_Toc147408150)

[1.2. Nguyên lý hoạt động 9](#_Toc147408151)

[1.3. Nguyên lý dịch & gỡ rối 10](#_Toc147408152)

[2. BÀI THÍ NGHIỆM THỰC HÀNH 12](#_Toc147408153)

[2.1. Mục đích - Yêu cầu 12](#_Toc147408154)

[**2.1.1. Mục đích 12**](#_Toc147408155)

[**2.1.2. Yêu cầu: 12**](#_Toc147408156)

[**2.1.3. Thời gian thực hiện 12**](#_Toc147408157)

[2.2. Chuẩn bị 12](#_Toc147408158)

[**2.2.1. Danh mục thiết bị thực hành 12**](#_Toc147408159)

[2.3. Nội dung 13](#_Toc147408160)

[**2.3.1. Các bước thực hiện 13**](#_Toc147408161)

[**2.3.2. Kết quả thực hiện 25**](#_Toc147408162)

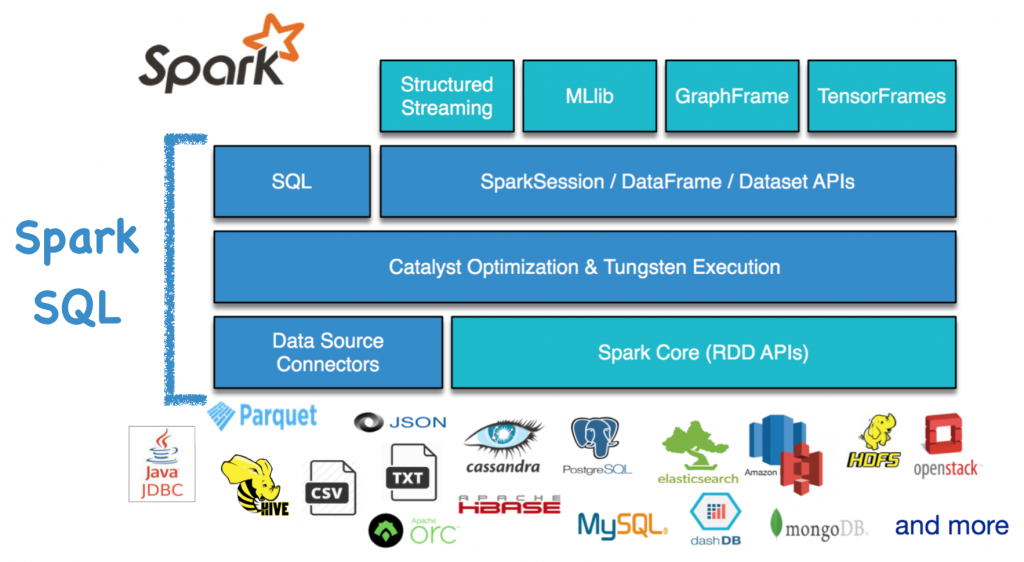
[TÀI LIỆU HƯỚNG DẪN THỰC HÀNH 25](#_Toc147408163)

[PHỤ LỤC 26](#_Toc147408164)

# 1. CƠ SỞ LÝ THUYẾT

## 1.1. Giới thiệu chung

***Làm quen với kiến trúc***



Hình 1 Giới thiệu Spark SQL

**+** Khái niệm***:***

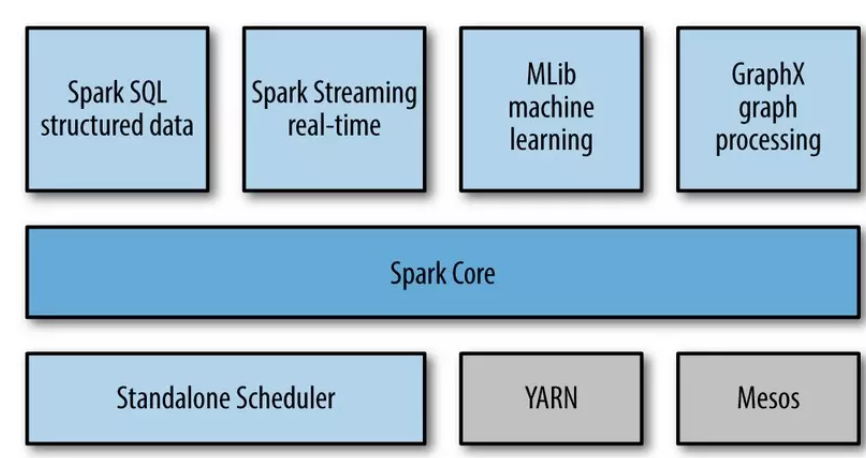
* Apache Spark: Apache Spark là một framework xử lý dữ liệu phân tán mã nguồn mở được phát triển bởi Apache Software Foundation. Nó cung cấp khả năng xử lý và phân tích dữ liệu lớn một cách hiệu quả, hỗ trợ xử lý batch, xử lý dự đoán thời gian thực, và xử lý dữ liệu trực tiếp từ nhiều nguồn khác nhau. Spark sử dụng cơ sở dữ liệu phân tán in-memory để tăng tốc quá trình xử lý dữ liệu.
* Python - Truyền phát có cấu trúc Spark: Đây là một cách sử dụng Python để xây dựng ứng dụng dựa trên Apache Spark để xử lý và phân tích dữ liệu có cấu trúc. Python là một ngôn ngữ lập trình phổ biến trong cộng đồng khoa học dữ liệu và Spark cung cấp API cho Python để xử lý dữ liệu một cách dễ dàng.

**+** Đặc điểm

* Python:
* Python là một ngôn ngữ lập trình mã nguồn mở, dễ đọc và dễ hiểu.
* Đa năng: Python được sử dụng cho nhiều mục đích, bao gồm phát triển ứng dụng web, khoa học dữ liệu, máy học, và nhiều lĩnh vực khác.
* Cộng đồng lớn: Python có cộng đồng phát triển và hỗ trợ lớn, có nhiều thư viện và framework sẵn sàng.
* API (Application Programming Interface):
* Giao diện lập trình ứng dụng: API là tập hợp các quy tắc và hướng dẫn cho phép các ứng dụng giao tiếp với nhau.
* Hỗ trợ tích hợp: API cho phép các ứng dụng và dịch vụ khác nhau tương tác và chia sẻ dữ liệu một cách dễ dàng.
* Apache Spark:
* Xử lý dữ liệu phân tán: Spark là một framework xử lý dữ liệu phân tán, có khả năng xử lý dữ liệu nhanh chóng và hiệu quả.
* Hỗ trợ đa ngôn ngữ: Spark hỗ trợ nhiều ngôn ngữ lập trình và cung cấp thư viện mạnh mẽ cho xử lý dữ liệu.

+ Cấu trúc

* Apache Spark



Hình 2 kiến trúc của Apache Spark

* Thành phần chung của Spark là Spark Core: cung cấp những chức năng cơ bản nhất của Spark như lập lịch cho các tác vụ, quản lý bộ nhớ, fault recovery, tương tác với các hệ thống lưu trữ…Đặc biệt, Spark Core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet) là tập hợp của các item được phân tán trên các node của cluster và có thể được xử lý song song.
* Spark có thể chạy trên nhiều loại Cluster Managers như Hadoop YARN, Apache Mesos hoặc trên chính cluster manager được cung cấp bởi Spark được gọi là Standalone Scheduler.
* Spark SQL cho phép truy vấn dữ liệu cấu trúc qua các câu lệnh SQL. Spark SQL có thể thao tác với nhiều nguồn dữ liệu như Hive tables, Parquet, và JSON.
* MLlib Cung cấp nhiều thuật toán của học máy như: classification, regression, clustering, collaborative filtering…
* GraphX là thư viện để xử lý đồ thị.

• Spark SQL là mô-đun Spark để xử lý dữ liệu có cấu trúc. Không giống như API Spark RDD cơ bản, các giao diện do Spark SQL cung cấp cung cấp cho Spark nhiều thông tin hơn về cấu trúc của cả dữ liệu và tính toán đang được thực hiện. Trong nội bộ, Spark SQL sử dụng thông tin bổ sung này để thực hiện các tối ưu hóa bổ sung.

## *1.2. Nguyên lý hoạt động*

***Nguyên lý hoạt động của quy trình này có thể được mô tả như sau:***

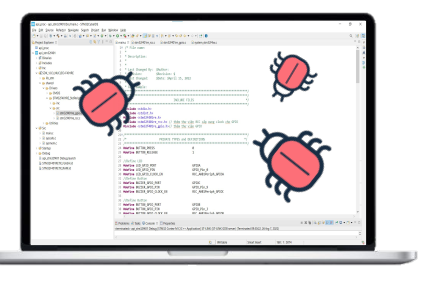
* Khi thực thi, Spark SQL vẫn sẽ gọi xuống lớp Core bên dưới, sử dụng RDD để tính toán. Được xây dựng phía trên tầng Spark Core, thừa hưởng tất cả các tính năng mà RDD có. Làm việc với tập dữ liệu là DataSet hoặc DataFrame (tập dữ liệu phân tán, có cấu trúc). Hiệu năng cao, khả năng mở rộng và chịu lỗi tốt. Tương tích với các thành phần khác trong tổng thể Spark Framework (như Streaming/ Mllib, GraphX)
* Có một số cách để tương tác với Spark SQL bao gồm SQL và API bộ dữ liệu. Khi tính toán một kết quả, cùng một công cụ thực thi sẽ được sử dụng, không phụ thuộc vào API/ngôn ngữ bạn đang sử dụng để thể hiện tính toán. Sự thống nhất này có nghĩa là các nhà phát triển có thể dễ dàng chuyển đổi qua lại giữa các API khác nhau, dựa vào đó cung cấp cách tự nhiên nhất để thể hiện một chuyển đổi nhất định.
* SparkSQL có thể chạy trong spark-shell, pysparkshell hoặc sparkRshell.

***=>Toàn bộ quy trình này cho phép xử lý, truy cập dữ liệu một cách hiệu quả và có khả năng mở rộng, phù hợp cho các ứng dụng đòi hỏi xử lý lượng dữ liệu lớn và đáng tin cậy.***

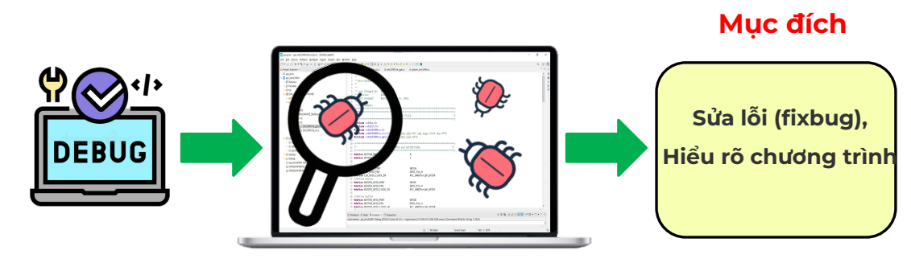
## *1.3. Nguyên lý dịch & gỡ rối*

***+*** Bug được hiểu là thuật ngữ dùng để chỉ các lỗi kỹ thuật có thể xảy ra trong quá trình thiết kế và vận hành các chương trình lập trình.

+ Bug khiến cho phần mềm, ứng dụng không thực thi được hoặc thực thi sai.

**

+ Debug là quá trình tìm kiếm và phát hiện nguyên nhân gây ra lỗi

****

***+ Các tùy chọn với chế độ debug***

* *Debugging API Data Ingestion (Gỡ lỗi Nhận dữ liệu từ API):*
* In-Memory Logging: Dữ liệu nhận được từ API có thể được ghi lại trong bộ nhớ để dễ dàng kiểm tra và gỡ lỗi.
* Data Validation: Kiểm tra tính chính xác của dữ liệu đầu vào và gửi thông báo lỗi nếu có sự cố.
* Debugging Spark Data Processing (Gỡ lỗi Xử lý dữ liệu bằng Spark):

Logging và Tracing: Sử dụng các hệ thống log và truy vết để theo dõi dữ liệu và tiến trình xử lý trong Spark.

* Unit Testing: Phát triển các bài kiểm tra đơn vị để xác định xem các phần xử lý dữ liệu hoạt động đúng cách.
* Spark UI: Sử dụng giao diện người dùng Spark để xem xét thông tin về quá trình xử lý và đánh giá hiệu suất.

# 2. BÀI THÍ NGHIỆM THỰC HÀNH

## 2.1. Mục đích - Yêu cầu

### 2.1.1. Mục đích

+ Giúp sinh viên làm quen với các công nghệ hiện đại, nắm rõ cách thức sử dụng Spark SQL để thực thi các truy vấn SQL.

+ Biết cách đọc thư viện được dựng sẵn.

+ Biết các khởi tạo SparkSession

+ Biết kiểm tra dữ liệu: dtypes, show, head, first, take, describe, count, distinct, explain, printSchema,...

+ Biết và sử dụng các lệnh truy vấn dữ liệu thông dụng nhất trong thực tế:

* Filter, Duplicate Values
* Queries: select, when, like, Startswith Endswith, Substring, Between
* Add, Update & Remove Columns
* Missing & Replacing Values
* GroupBy, Sort, Repartitioning
* Ghi file
* Stop Session

### 2.1.2. Yêu cầu:

* Nắm vững kiến thức được giới thiệu, tự tìm hiểu thêm nhiều chức năng khác.
* Hoàn thành bài code truy vấn dữ liệu sử dụng Spark theo yêu cầu của giảng viên.

### 2.1.3. Thời gian thực hiện

+ Thời gian mỗi buổi thực hành là từ 3-4 giờ, chia làm 10-12 nhóm nhỏ, mỗi nhóm 2 sinh viên/1 máy tính. Sinh viên tìm hiểu cơ sở lý thuyết ngắn gọn và các bước thực hiện có thể thao tác dễ dàng.

## 2.2. Chuẩn bị

### 2.2.1. Danh mục thiết bị thực hành

+ Phần cứng: 01 máy tính. Tất cả được được đồng bộ theo số thứ tự từ 1-20

+ Phần mềm: Apache Spark, PySpark

**Danh mục thiết bị thực hành phòng lab**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | Tên thiết bị | Số lượng | Thông số  kỹ thuật | Vai trò |
| 1 | Máy tính để bàn | 01 | - Intel Core i3-2100 3.1 Ghz/3M Cache  - 4GB DDR3 (nâng cấp lên ram 8GB thêm 300K)  -SSD 120G Gb chuẩn SATA 3 - 6Gb/s Ổ SSD | -Tính toán, viết chương trình, chạy chương trình |

## 2.3. Nội dung

### 2.3.1. Các bước thực hiện

***Nội dung 1:*** cài đặt Apache Spark, cài đặt pip, cài đặt Pyspark.

**Bước 1: Tạo một project mới “PROJECT1”**

**Bước 2: Cài đặt Apache Spark**

|  |
| --- |
| """  Cài đặt các gói cần thiết cho Spark  """  sudo apt install default-jdk scala git -y  """  Tải xuống và thiết lập Spark cho Ubuntu  """  <https://spark.apache.org/downloads.html>  tar xvzf <ten\_file\_nen\_spark>.tgz  """  Cấu hình môi trường Spark  """  export SPARK\_HOME=<duong\_dan\_toi\_thu\_muc\_ban\_vua\_dat\_spark>  export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin  export PYSPARK\_PYTHON=/usr/bin/python3  """  Cài đọc pip, pyspark, pandas  """  sudo apt install pip  sudo apt install python-pip  pip install pyspark  pip install pandas |

***Nội dung 2: thực hành truy vấn SQL với Spark***

|  |
| --- |
| # Khai báo các thư viện cần thiết  from pyspark.sql import SparkSession  from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType  # Mở sparksesstion  spark = SparkSession \  .builder \  .appName("Python Spark SQL basic example") \  .config("spark.some.config.option", "some-value") \  .getOrCreate()  # Tạo định dạng schema  schema = StructType([StructField('Country', StringType(), True),  StructField('Code', StringType(), True),  StructField('Year', IntegerType(), True),  StructField('GDP', FloatType(), True)  ])  # Đọc dữ liệu  df1 = spark.read.csv('drive/My Drive/PTIT/BigDataCourse/Data/country\_data.csv', header=True, schema=schema)  # Kiểm tra dữ liệu  df1.show(5)  print(df1.count())  df1.printSchema()  # Chúng ta có thể thực hiện các thao tác SQL trên khung dữ liệu bằng cách tạo temporary vie  df1.createOrReplaceTempView('GDP')  # Selecting multiple columns from dataframe  spark.sql('select Country, GDP from GDP').show(5)  # length counts the length of the string  spark.sql('select Country, length(Country) from GDP').show(5)  # Selecting first three letters from the country  spark.sql("select Country, left(Country, 3) from GDP").show(5)  # Selecting multiple columns with multiple conditions  spark.sql("select Country, Year, GDP from GDP\  where Year = 2000 and GDP < 35E9").show()  # Selecting country not having vowel a in its name and starting with S  spark.sql("select Country from GDP\  where Country not like '%a%' \  and Country like 'S%'").show(5)  # Use of between  spark.sql("select Year, GDP from GDP \  where Country = 'Vietnam' \  and Year between 2005 and 2010").show(5)  # applying condition on selection  spark.sql("select Year, GDP from GDP where Country = 'Nepal'").show(5)  # applying mathematical condition on selection  spark.sql('select Country from GDP where GDP < 10E9').show(5)  # Star selects all the columns from the dataframe  spark.sql("select \* from GDP where Year = 2011").show(5)  # Select multiple column with condition  spark.sql("select Country, Year from GDP where Year = 2000").show(5)  # Selecting countries whose name starts from Z  spark.sql("select Country from GDP where Country like 'Z%'").show(5)  # Use of in  spark.sql("select Country, GDP from GDP \  where Country in ('Nepal', 'India') \  and Year = 2000").show(5)  # Use of in outside condition can be used to create yes no question  spark.sql("select distinct Country, Country in ('Pakistan', 'India') from GDP").show(4)  # Use of distinct  spark.sql("select distinct \* from GDP where length(Country) < 5").show(5)  # A combination for fun  spark.sql("select distinct \* from GDP where Country like '%l'").show(5)  # Order by  spark.sql("select \* from GDP where Year = 2015 order by GDP desc").show(4)  # We can create another temp view  df1.createOrReplaceTempView('Country')  # We can select from COuntry now  spark.sql('SELECT DISTINCT Country, GDP FROM Country\  WHERE GDP < 10').show()  # GDP is still working  spark.sql('SELECT Country FROM GDP\  WHERE Year=2010').show(5)  # How many country name start with United?  spark.sql("select distinct Country from GDP where Country like 'United%'").show()  # Subquerry  # Which country in 2010 had GDP larger than the GDP of Germany in 2015?  spark.sql("select distinct Country from GDP \  where Year = 2010 and \  GDP > (select GDP from GDP \  where Country = 'Germany'\  and Year = '2015')").show()  # Lowest GDP which is not null  spark.sql("select \* from GDP where GDP > 0 order by GDP asc").show(3)  # Countries with the shortest name  spark.sql("select distinct Country, length(Country) from GDP \  where length(Country)>0 \  order by length(Country)").show(10)  # Longest country name  spark.sql("select Country, Year from GDP order by length(Country) desc").show(4)  # Viewing the full name  spark.sql("select Country, Year from GDP order by length(Country) desc").first()  # Total GDP of the world in 2000  spark.sql("select sum(GDP) from GDP where Year = 2000").show()  # Average GDP of Chile  spark.sql("select mean(GDP) from GDP where Country = 'Chile'").show()  # Number of countires in the world  spark.sql("select count(distinct Country) from GDP").show()  # Total GDP of India and Pakistan in 2000  spark.sql("select sum(GDP) from GDP \  where Country in ('India', 'Pakistan')\  and Year = 2000").show()  # Number of countries with GDP more than 30000 in 2000  spark.sql("select count(Country) from GDP \  where Year = 2000 \  and GDP > 30000").show() |

**Cấu hình và Chạy Spark**

|  |
| --- |
| ***spark-submit --master local[2] <Đường dẫn>/<file\_code>.py*** |

# TÀI LIỆU HƯỚNG DẪN THỰC HÀNH

[1] Tài liệu:

https://github.com/roshankoirala/pySpark\_tutorial/blob/master/SQL\_commands\_in\_pySpark.ipynb

# PHỤ LỤC

**Phiếu báo cáo kết quả thực hành (Sinh viên)**

Tên bài: ……………………………………………………………………………...

Họ và tên sinh viên……………………………Mã sinh viên……………………...

………………………………………………………………..……………………...

Nhóm…………………..Lớp……………… ..Ngày…..tháng…..năm…………..

Giáo viên hướng dẫn…………………………Ca thực tập……………………….

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Nội dung  thực hành | Mức độ  hoàn thành (%) | Thời gian  hoàn thành | Đánh giá kết quả (100) |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| Thảo luận sinh viên: | | | | |

**Phiếu đánh giá kết quả thực hành (Giảng viên)**

Tên bài: ……………………………………………………………………………...

Họ và tên sinh viên……………………………Mã sinh viên……………………...

Nhóm…………………..Lớp…………………Ngày…..tháng…..năm…………..

Giáo viên hướng dẫn………………………….Ca thực tập……………………….

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thứ tự** | **Nội dung đánh giá** | **Điểm chuẩn** | **Yêu cầu** | **Ghi chú** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Tổng điểm: | | | | |
| Nhận xét giảng viên: | | | | |