BTVN Buổi 5 (BT2): Medallion Architecture (Bronze, Silver, Gold)

Tác giả: Đinh Thiên Ân - 22520010
Giảng viên hướng dẫn: Cử nhân Trần Quốc Khánh

1. Khái niệm Medallion Architecture

Medallion Architecture (còn được gọi là Multi-hop Architecture) là một mô hình thiết kế dành cho data lakehouse, được phát triển bởi Databricks. Kiến trúc này tổ chức dữ liệu thành các tầng chất lượng khác nhau, thường được biểu thị bằng các "huy chương" Bronze, Silver và Gold. Mỗi tầng đại diện cho một mức độ tinh chế và giá trị dữ liệu tăng dần.

Kiến trúc này giải quyết các thách thức chính trong việc quản lý dữ liệu lớn:

- Tích hợp dữ liệu từ nhiều nguồn khác nhau
- Đảm bảo chất lượng và tính nhất quán của dữ liệu
- Hỗ trợ nhiều trường hợp sử dụng khác nhau
- Quản lý lineage (nguồn gốc) của dữ liệu

Các tầng trong Medallion Architecture:

1. **Bronze (Raw Layer)**: Dữ liệu thô, chưa qua xử lý
2. **Silver (Validated Layer)**: Dữ liệu đã được làm sạch và chuẩn hóa
3. **Gold (Curated Layer)**: Dữ liệu sẵn sàng cho các báo cáo và phân tích

2. Cách thực thi Medallion Architecture

Tầng Bronze

- **Mục đích**: Lưu trữ dữ liệu thô từ nhiều nguồn khác nhau.
- **Đặc điểm**:
  - Dữ liệu được lưu ở dạng gốc, không hoặc ít biến đổi
  - Thường bao gồm metadata như timestamp, nguồn dữ liệu
  - Có thể chứa dữ liệu trùng lặp, thiếu, hoặc không chính xác
- **Công nghệ thường dùng**:
  - Apache Kafka hoặc Kinesis cho dữ liệu streaming
  - Spark để xử lý batch
  - Lưu trữ dưới dạng Parquet hoặc Delta Lake

Tầng Silver

- **Mục đích**: Làm sạch, chuẩn hóa và xác thực dữ liệu từ tầng Bronze.
- **Đặc điểm**:
  - Loại bỏ dữ liệu trùng lặp
  - Xử lý dữ liệu thiếu
  - Chuẩn hóa và chuyển đổi schema

    - Kết hợp dữ liệu từ nhiều nguồn
    - Thêm validation rules
- **Công nghệ thường dùng**:
    - Apache Spark SQL
    - Delta Lake với time travel và ACID transactions
    - Great Expectations hoặc Deequ cho data validation

Tầng Gold

- **Mục đích**: Cung cấp dữ liệu cho người dùng cuối, phân tích và báo cáo.
- **Đặc điểm**:
    - Tổng hợp dữ liệu theo nhiều chiều
    - Tạo các business metrics và KPIs
    - Tối ưu hóa cho truy vấn và phân tích
    - Có tính bảo mật cao
- **Công nghệ thường dùng**:
    - Data warehousing solutions (Snowflake, Redshift, BigQuery)
    - BI tools (Tableau, Power BI, Looker)
    - Feature stores cho ML

Luồng dữ liệu điển hình:

1. Thu thập dữ liệu thô --> **Bronze Layer**
2. Validation, cleaning, transformation --> **Silver Layer**
3. Aggregation, enrichment, feature engineering --> **Gold Layer**
4. Consumption, visualization, machine learning

3. Các nguyên lý của Medallion Architecture

1. Immutability (Tính bất biến)

- Dữ liệu sau khi được đưa vào tầng không thay đổi nội dung
- Các thay đổi được thực hiện bằng cách thêm bản ghi mới
- Hỗ trợ việc phục hồi dữ liệu và kiểm tra lịch sử

2. Incremental Processing (Xử lý gia tăng)

- Chỉ xử lý dữ liệu mới hoặc đã thay đổi
- Giảm thời gian và tài nguyên cần thiết
- Hỗ trợ cả batch processing và stream processing

3. Schema Enforcement (Thực thi schema)

- Áp dụng schema validation ở mỗi tầng
- Đảm bảo tính nhất quán của dữ liệu
- Ngăn chặn dữ liệu không hợp lệ

4. Data Lineage (Nguồn gốc dữ liệu)

- Theo dõi luồng dữ liệu từ nguồn đến đích

- Giúp hiểu rõ quá trình chuyển đổi dữ liệu
- Hỗ trợ audit và compliance

5. Separation of Concerns (Phân tách các mối quan tâm)

- Mỗi tầng có một nhiệm vụ rõ ràng
- Tách biệt giữa ingestion, processing và serving
- Dễ bảo trì và mở rộng

4. Tiêu chí đánh giá Medallion Architecture

1. Khả năng mở rộng (Scalability)

- **Đánh giá**: Khả năng xử lý khối lượng dữ liệu lớn và tăng trưởng theo thời gian
- **Phương pháp đo lường**: Thời gian xử lý, chi phí tài nguyên khi khối lượng dữ liệu tăng

2. Tính linh hoạt (Flexibility)

- **Đánh giá**: Khả năng thích ứng với các nguồn dữ liệu và use cases mới
- **Phương pháp đo lường**: Thời gian triển khai nguồn dữ liệu mới, thay đổi schema

3. Tính tin cậy (Reliability)

- **Đánh giá**: Khả năng phục hồi sau lỗi, đảm bảo tính chính xác của dữ liệu
- **Phương pháp đo lường**: Tỷ lệ thành công của job, thời gian phục hồi sau lỗi

4. Hiệu suất (Performance)

- **Đánh giá**: Tốc độ xử lý và truy vấn dữ liệu
- **Phương pháp đo lường**: Thời gian xử lý batch, độ trễ của real-time data

5. Khả năng quản trị (Governance)

- **Đánh giá**: Mức độ tuân thủ các quy định, kiểm soát truy cập, data lineage
- **Phương pháp đo lường**: Khả năng audit, level of compliance with regulations

5. Áp dụng Medallion Architecture vào dự án phân tích dữ liệu việc làm

Dự án phân tích dữ liệu việc làm tại Việt Nam có thể áp dụng Medallion Architecture để cải thiện quá trình xử lý dữ liệu và tăng giá trị của các phân tích.

Bronze Layer - Thu thập dữ liệu thô

- **Dữ liệu đầu vào**:
  - Dữ liệu web scraping từ các trang tuyển dụng (trong `data/raw`)
  - Logs của quá trình thu thập dữ liệu
  - Metadata (thời gian thu thập, nguồn, version)
- **Script mẫu**:

```
# Bronze Layer - Lưu trữ dữ liệu thô từ web scraping
from pyspark.sql import SparkSession
from datetime import datetime

spark = SparkSession.builder.appName("JobDataBronzeLayer").getOrCreate()

# Đọc dữ liệu từ CSV files sau khi scrape
raw_jobs_df = spark.read.csv("data/raw/job_data_raw.csv", header=True,
inferSchema=True)

# Thêm metadata
bronze_df = raw_jobs_df.withColumn("ingestion_date",
lit(datetime.now().strftime("%Y-%m-%d")))
bronze_df = bronze_df.withColumn("source", lit("job_portal_X"))
bronze_df = bronze_df.withColumn("batch_id", lit(uuid.uuid4().hex))

# Lưu vào bronze layer
bronze_df.write.format("delta").mode("append").save("data/bronze/raw_jobs")

# Ghi log
print(f"Saved {bronze_df.count()} records to bronze layer")
```

Silver Layer - Dữ liệu đã làm sạch và chuẩn hóa

- **Quá trình xử lý**:
  - Xử lý giá trị thiếu (missing values)
  - Chuẩn hóa dữ liệu văn bản và loại bỏ duplicate
  - Chuyển đổi dữ liệu kinh nghiệm sang định dạng chuẩn (số năm)
  - Chuẩn hóa thông tin lương
  - Xác thực dữ liệu theo business rules
- **Script mẫu**:

```
# Silver Layer - Làm sạch và chuẩn hóa dữ liệu
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
import re

spark = SparkSession.builder.appName("JobDataSilverLayer").getOrCreate()

# Đọc dữ liệu từ Bronze layer
bronze_df = spark.read.format("delta").load("data/bronze/raw_jobs")

# 1. Loại bỏ duplicate
silver_df = bronze_df.dropDuplicates(["job_title", "company", "url"])

# 2. Chuẩn hóa thông tin lương
def normalize_salary(salary_text):
    # Chuyển đổi các format lương khác nhau về một format chung
    # Ví dụ: "15-20 triệu" -> min_salary: 15000000, max_salary: 20000000
    # Code xử lý...
```

```
    return min_salary, max_salary

# Đăng ký UDF
spark.udf.register("normalize_salary", normalize_salary, StructType([
    StructField("min_salary", LongType()),
    StructField("max_salary", LongType())
]))

# Áp dụng hàm normalize_salary
silver_df = silver_df.withColumn(
    "salary_struct",
    expr("normalize_salary(salary_text)")
)
silver_df = silver_df.withColumn("min_salary", col("salary_struct.min_salary"))
silver_df = silver_df.withColumn("max_salary", col("salary_struct.max_salary"))

# 3. Chuẩn hóa thông tin kinh nghiệm
# Code xử lý tương tự...

# 4. Thêm quality metrics
silver_df = silver_df.withColumn("data_quality_score",
    when(col("min_salary").isNull() | col("max_salary").isNull(), 0.7)
    .when(col("experience").isNull(), 0.8)
    .otherwise(1.0)
)

# Lưu vào silver layer
silver_df.write.format("delta").mode("overwrite").save("data/silver/clean_jobs")
```

Gold Layer - Dữ liệu phân tích

- **Các bảng/views**:
  - `job_salary_by_industry`: Phân tích lương theo ngành
  - `job_distribution_by_location`: Phân bố công việc theo địa điểm
  - `experience_salary_correlation`: Mối quan hệ giữa kinh nghiệm và lương
  - `company_size_analysis`: Phân tích theo quy mô công ty
  - Feature tables cho mô hình dự đoán lương
- **Script mẫu**:

```
# Gold Layer - Tạo các aggregations cho phân tích
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

spark = SparkSession.builder.appName("JobDataGoldLayer").getOrCreate()

# Đọc dữ liệu từ Silver layer
silver_df = spark.read.format("delta").load("data/silver/clean_jobs")

# 1. Tạo bảng phân tích lương theo ngành
salary_by_industry = silver_df.groupBy("industry").agg(
```

```
    avg("min_salary").alias("avg_min_salary"),
    avg("max_salary").alias("avg_max_salary"),
    stddev("max_salary").alias("stddev_salary"),
    count("*").alias("job_count")
)

# Lưu vào gold layer
salary_by_industry.write.format("delta").mode("overwrite").save("data/gold/salary_
by_industry")

# 2. Tạo bảng phân tích theo địa điểm
job_by_location = silver_df.groupBy("location").agg(
    count("*").alias("job_count"),
    avg("max_salary").alias("avg_salary"),
    countDistinct("industry").alias("industry_diversity")
)

job_by_location.write.format("delta").mode("overwrite").save("data/gold/job_by_loc
ation")

# 3. Tạo bảng feature cho mô hình ML
ml_features = silver_df.select(
    "industry", "location", "company_size", "experience_years",
    "min_salary", "max_salary", "required_skills"
)

ml_features.write.format("delta").mode("overwrite").save("data/gold/ml_features")
```

6. Lợi ích của Medallion Architecture trong dự án

1. **Chất lượng dữ liệu tốt hơn**: Phát hiện và xử lý sớm các vấn đề dữ liệu.
2. **Tái sử dụng và tính nhất quán**: Mỗi tầng có thể được sử dụng bởi nhiều ứng dụng khác nhau.
3. **Khả năng mở rộng**: Dễ dàng thêm nguồn dữ liệu mới (thêm trang tuyển dụng).
4. **Hiệu suất tốt hơn**: Dữ liệu được tối ưu hóa cho từng mục đích sử dụng.
5. **Truy xuất nguồn gốc**: Có thể theo dõi dữ liệu từ báo cáo cuối cùng đến nguồn gốc thu thập.
6. **Quản lý phiên bản**: Dễ dàng theo dõi thay đổi và phục hồi khi cần.