EA4. Documentación de la Arquitectura y Modelo de Datos

Por

Shara Milena Mosquera Goez

Docente

Andres Felipe Callejas

CURSO

Infraestructura y arquitectura para Big Data

INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

MEDELLÍN

2025

Introducción

Este documento describe la arquitectura y el modelo de datos de un proyecto que simula un entorno de Big Data en la nube, enfocado en procesar datos de los laureados y premios Nobel. Utilizamos tecnologías como SQLite para manejar los datos de manera eficiente y scripts en Python para la limpieza y enriquecimiento de los datos, permitiendo análisis más complejos.

El proyecto se desarrolla en varias etapas, desde la recolección de datos hasta su enriquecimiento, mejorando así el acceso y la calidad de los datos para futuras consultas. Esta documentación detalla cada componente del sistema, explica la estructura de datos, y justifica la elección de herramientas usadas en el proyecto.

Finalmente, se ofrecen recomendaciones para mejorar y adaptar el sistema para su implementación en entornos de nube reales, buscando mantener la relevancia del sistema y aprovechar las nuevas tecnologías para optimizar su rendimiento.

Descripción Global de la Arquitectura

Este proyecto está diseñado para simular un entorno de Big Data en la nube, optimizado para el manejo y procesamiento de vastos volúmenes de datos relacionados con los laureados y premios Nobel. La arquitectura del sistema se estructura en varias fases clave, cada una crucial para el tratamiento efectivo de los datos desde su origen hasta su análisis final.

1. **Ingesta de Datos:**
    - Objetivo: Recolección y almacenamiento inicial de datos.
    - Proceso: Los datos se recopilan de diversas fuentes y se almacenan en una base de datos SQLite, asegurando una estructura organizada desde el principio. Durante esta fase, también se genera un archivo de auditoría que es fundamental para rastrear la integridad y el origen de los datos, garantizando la transparencia y la fiabilidad del sistema desde el comienzo.
2. **Limpieza de Datos:**
    - Objetivo: Asegurar la calidad de los datos.
    - Proceso: Una vez almacenados los datos, el siguiente paso crucial es su limpieza. Esto incluye eliminar entradas duplicadas o nulas y verificar la validez de los tipos de datos, con especial atención a las fechas. Este proceso no solo mejora la calidad de los datos, sino que también optimiza el rendimiento para las consultas que se realizarán posteriormente.
3. **Enriquecimiento de Datos:**

    - Objetivo: Aumentar el valor de los datos existentes.

    - Proceso: En esta etapa, los datos ya limpios se enriquecen mediante la integración con otras fuentes de datos utilizando claves foráneas. Esto mejora significativamente la utilidad de la información almacenada, permitiendo análisis más complejos y detallados.

La implementación de estas fases se lleva a cabo en un entorno simulado de Big Data en la nube, lo que permite una escalabilidad eficiente de las operaciones de procesamiento de datos. Este enfoque facilita la gestión de grandes volúmenes de información y mejora su accesibilidad desde diferentes ubicaciones geográficas, destacando la capacidad del sistema para adaptarse a necesidades de análisis globalizado.

Componentes Principales del Proyecto

Base de Datos Analítica (SQLite): Utilizamos SQLite por su simplicidad y eficiencia, ideal para manejar volúmenes considerables de datos con requerimientos mínimos de configuración. Es perfecta para leer y escribir datos simultáneamente, lo cual es esencial para nuestro proyecto.

Scripts de Procesamiento:

- **Ingesta.py**: Crea la base de datos y las tablas, inserta datos y genera registros de auditoría.

- **cleaning.py**: Se conecta a la base de datos, extrae datos y los limpia.

- **enriched_data.py**: Enriquece los datos limpios usando técnicas de fusión de datos a través de claves foráneas.

Mecanismo de Automatización (GitHub Actions)

- **Descripción**: Implementamos GitHub Actions como nuestro mecanismo de automatización para gestionar y automatizar la ejecución secuencial de los scripts de ingesta, limpieza y enriquecimiento de datos.

- **Detalles del Workflow**:

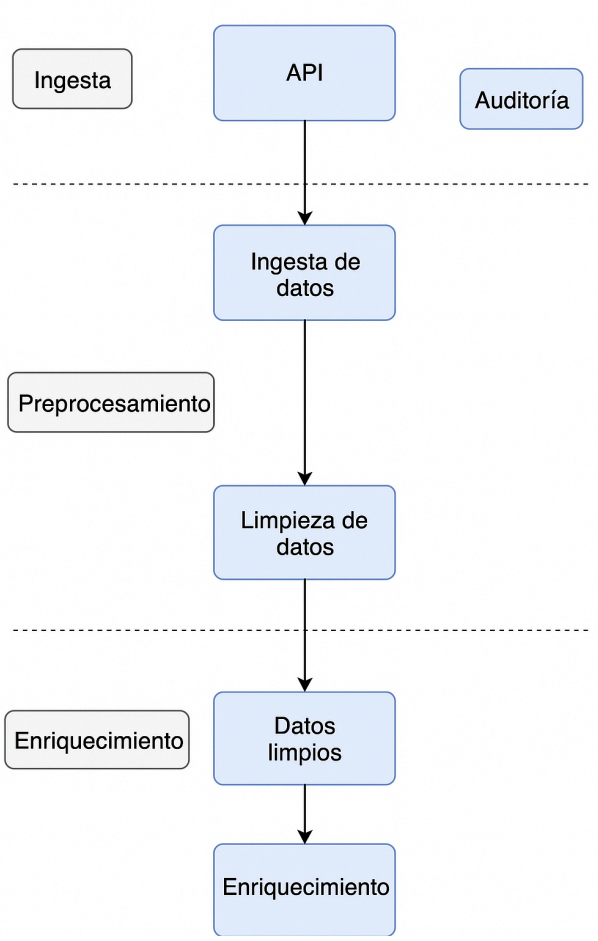
    - **Activación**: El workflow se activa con cada 'push' en la rama principal (main).

    - **Procesos Automatizados**:

- Configuración y preparación del entorno de Python.
- Limpieza de archivos residuales de ejecuciones anteriores.
- Ejecución de los scripts Ingesta.py, cleaning.py, y enriched_data.py para procesar los datos.
- Compromiso y carga automática de los cambios realizados durante el proceso.

Estos componentes son fundamentales para el funcionamiento del sistema y garantizan que el proyecto pueda manejar, procesar y analizar datos de manera eficiente en un entorno simulado de Big Data.

Diagrama de flujo



Definición del Esquema

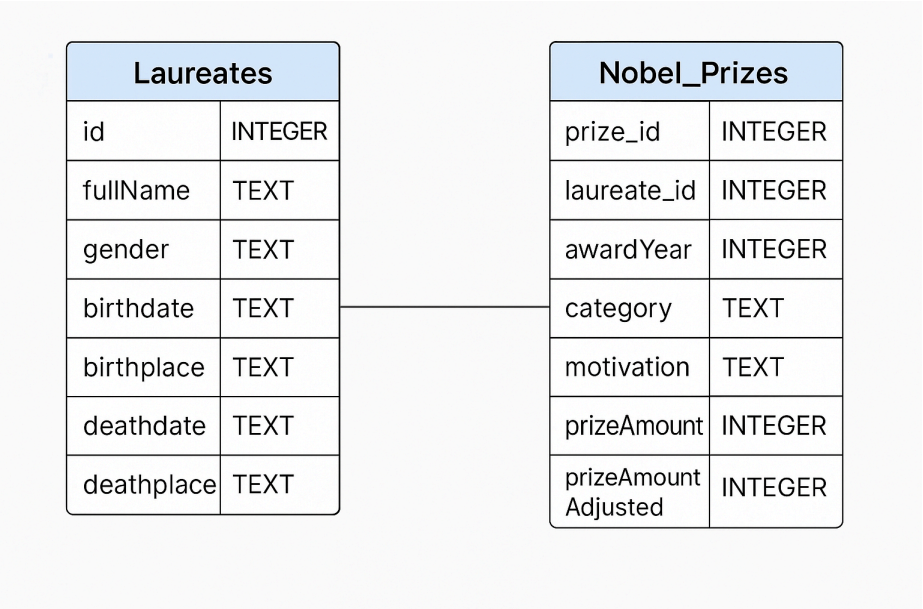
Tablas y sus Campos

1. **Laureates**
    - **id** (INTEGER, PRIMARY KEY): Identificador único de cada laureado.
    - **fullName** (TEXT): Nombre completo del laureado.
    - **gender** (TEXT): Género del laureado.
    - **birthdate** (TEXT): Fecha de nacimiento del laureado.
    - **birthplace** (TEXT): Lugar de nacimiento del laureado.
    - **deathdate** (TEXT): Fecha de fallecimiento del laureado.
    - **deathplace** (TEXT): Lugar de fallecimiento del laureado.
2. **Nobel_Prizes**
    - **prize_id** (INTEGER, PRIMARY KEY, AUTOINCREMENT): Identificador único de cada premio Nobel.
    - **laureate_id** (INTEGER, FOREIGN KEY): Referencia a Laureates.id.
    - **awardYear** (INTEGER): Año en que se otorgó el premio.
    - **category** (TEXT): Categoría del premio (p.ej., Física, Química, Paz).
    - **motivation** (TEXT): Motivación citada para la adjudicación del premio.
    - **prizeAmount** (INTEGER): Cantidad del premio en la moneda original.
    - **prizeAmountAdjusted** (INTEGER): Cantidad del premio ajustada a la inflación actual.

Relaciones entre las Tablas

- La tabla **Laureates** y **Nobel_Prizes** están relacionadas mediante laureate_id. Esta clave foránea en la tabla **Nobel_Prizes** apunta al id en **Laureates**, permitiendo que múltiples premios estén asociados a un solo laureado.

Diagrama de Datos



Justificación de Herramientas y Tecnologías

Elección de Herramientas:

1. **SQLite**
    - **Justificación**: SQLite es una base de datos ligera y autocontenida que no requiere un servidor de base de datos separado. Es ideal para proyectos que necesitan ser ágiles y portátiles, y donde la configuración simplificada es una prioridad. Aunque no escala horizontalmente como otros sistemas de gestión de bases de datos, es suficientemente robusta para manejar un volumen significativo de datos sin la complejidad adicional de un servidor dedicado.
    - **Contribución al Proyecto**: Facilita el desarrollo rápido y la experimentación sin necesidad de una infraestructura compleja, permitiendo que el proyecto sea fácilmente adaptable y portable. También asegura una baja barrera de entrada para configurar y ejecutar el sistema.
2. **Pandas**
    - **Justificación**: Pandas es una biblioteca de Python extremadamente popular para manipulación y análisis de datos. Ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales, ideal para la limpieza y transformación de datos que este proyecto requiere.
    - **Contribución al Proyecto**: Permite realizar transformaciones de datos complejas de manera eficiente y con código que es relativamente fácil de leer y mantener. Además, la integración con otras bibliotecas y herramientas de Python facilita la expansión y adaptación del proyecto.

3. **PySpark**

    - **Justificación**: PySpark es una interfaz para Apache Spark, que permite el procesamiento distribuido de grandes conjuntos de datos a través de clusters de computadoras. PySpark es crucial para escalar operaciones de datos que exceden la capacidad de memoria de una sola máquina.

    - **Contribución al Proyecto**: Al simular un entorno de Big Data, PySpark permite que el proyecto se escale para manejar volúmenes de datos mucho mayores sin reescribir significativamente los procesos de datos. También ofrece la capacidad de procesar datos en tiempo real, lo cual es esencial para proyectos que pueden crecer en complejidad y requisitos de velocidad.

4. **GitHub Actions**

    - **Justificación**: GitHub Actions es una herramienta de CI/CD que permite automatizar flujos de trabajo directamente desde GitHub. Esto incluye desde la integración y entrega continua hasta la automatización de pruebas y despliegue.

    - **Contribución al Proyecto**: Facilita la automatización de pruebas, la construcción de software y el despliegue, asegurando que el código se pruebe y despliegue de manera consistente. Además, promueve prácticas de desarrollo colaborativo y asegura que los cambios en el código sean integrados y validados automáticamente, mejorando la calidad y la fiabilidad del software.

Simulación del Entorno Cloud:

- **Simulación mediante Scripts y Base de Datos**: Utilizando SQLite y scripts de Python (como Ingesta.py, cleaning.py, y enriched_data.py), el proyecto simula un entorno en la nube al permitir que el proceso de datos pueda ser ejecutado en cualquier entorno que soporte Python y SQLite, incluidos servicios en la nube que ofrecen máquinas virtuales o contenedores. Esto significa que, aunque se desarrolle y pruebe localmente, el proyecto puede ser fácilmente transferido a un entorno en la nube para aprovechar recursos computacionales escalables y distribuidos.

- **Integración con PySpark y GitHub Actions**: PySpark permite que el procesamiento de datos pueda escalarse a múltiples nodos en un cluster, que puede ser simulado localmente y luego implementado en la nube. GitHub Actions puede automatizar el despliegue de estos scripts en un entorno de nube, asegurando que se ejecuten consistentemente y en el entorno correcto. Esto es crítico para simular un escenario de Big Data realista, donde la carga de trabajo puede ser distribuida entre varios recursos computacionales en la nube.

Conclusiones y Recomendaciones

Beneficios:

- **Flexibilidad y Escalabilidad**: Usando SQLite y Python, nuestro sistema es flexible y escalable, ideal para proyectos en desarrollo.
- **Simplicidad y Portabilidad**: La arquitectura simple facilita mover el sistema entre diferentes entornos, como del desarrollo local a la nube.
- **Integridad de los Datos**: Nuestro modelo de datos relacional mantiene una integridad rigurosa, crucial para relaciones precisas entre los datos.
- **Automatización**: La automatización en la ingesta y limpieza de datos reduce errores y mejora nuestra eficiencia.

Limitaciones:

- **Escalabilidad de Datos**: SQLite puede ser limitado con grandes volúmenes de datos.
- **Acceso Multiusuario**: SQLite puede enfrentar problemas en entornos con muchos usuarios simultáneos.
- **Dependencia de Tecnología Local**: El uso intensivo de tecnologías locales puede restringir la adaptabilidad en entornos de nube.

Recomendaciones:

- **Actualizar la Base de Datos**: Considerar sistemas más robustos como PostgreSQL para mejorar el manejo de grandes datos.
- **Microservicios**: Modularizar la aplicación para mejorar la escalabilidad y eficiencia.
- **Contenedores**: Usar tecnologías como Docker para mejorar la portabilidad y consistencia del sistema.
- **CI/CD**: Implementar un pipeline de CI/CD con GitHub Actions para automatizar pruebas y despliegues.
- **Monitoreo**: Utilizar herramientas de monitoreo para optimizar el rendimiento y responder rápidamente a problemas técnicos.