Universidad Nacional Experimental De Guayana

Vicerrectorado Academico

Coordinacion General De Pregrado

Ingeniería En Informática

Sistema de Bases de Datos II

Proyecto Cassandra

Docente:

Clinia Cordero

Integrantes:

Jose Miserol V24.482.932

Miguel Gomez V29.862.177

Anthony Medina V30.857.435

Ciudad Guayana, Diciembre de 2025

Introduccion

El panorama de la gestión de datos ha evolucionado significativamente más allá del modelo relacional tradicional. En la era del Big Data, las aplicaciones web escalables y la necesidad de alta disponibilidad, surgieron las bases de datos NoSQL distribuidas. Apache Cassandra representa un sistema de gestión de bases de datos diseñado específicamente para manejar grandes volúmenes de datos distribuidos en múltiples nodos, ofreciendo alta disponibilidad sin punto único de fallo.

Cassandra se distingue por su arquitectura peer-to-peer que permite escalabilidad horizontal lineal, distribuyendo la carga de trabajo entre múltiples servidores de manera eficiente. Su modelo de datos basado en columnas anchas (wide-column) proporciona flexibilidad en el esquema mientras mantiene un rendimiento excepcional para operaciones de escritura y lectura distribuidas. A diferencia de las bases de datos relacionales que priorizan la consistencia inmediata (ACID), Cassandra implementa el modelo de consistencia eventual basado en el teorema CAP, priorizando la Disponibilidad (Availability) y la Tolerancia a Particiones (Partition Tolerance). Esto permite que el sistema continúe operando incluso cuando algunos nodos fallan, sincronizando los datos con el tiempo mediante mecanismos como hinted handoff y read repair.

GitHub. https://github.com/jose-miserol/sistemas-de-bases-de-datos-ii-proyecto-ii

Fase I - Diseño e Ingesta de Datos

Arquitectura del Sistema

El proyecto implementa un *pipeline* de datos escalable que integra tres tecnologías complementarias: Apache Cassandra, Apache Spark y ClickHouse. Esta arquitectura permite separar las cargas de trabajo transaccionales (OLTP) de las analíticas (OLAP), optimizando cada componente para su función específica.

- **Apache Cassandra** – Base de datos OLTP para ingesta masiva de datos transaccionales.
- **Apache Spark** – Motor de procesamiento distribuido para transformaciones ETL.
- **ClickHouse** – Base de datos columnar OLAP para consultas analíticas de alto rendimiento.

Diseño del Modelo de Datos en Cassandra

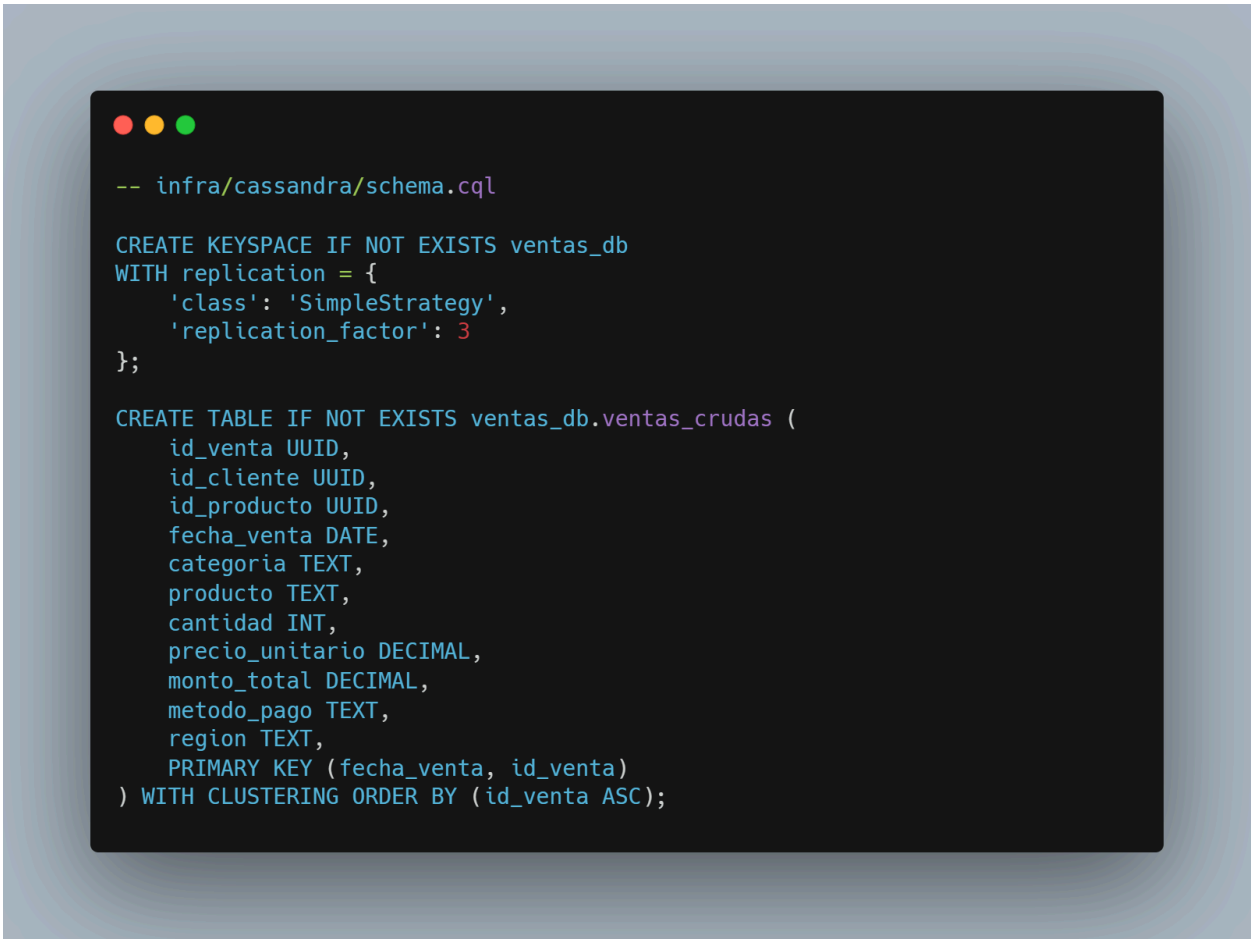
Principios de Modelado *Query-Driven*

A diferencia del modelado relacional normalizado, Cassandra requiere un enfoque *Query-Driven Design*, donde el esquema se diseña específicamente para las consultas que se ejecutarán. Los principios fundamentales aplicados fueron:

1. **Desnormalización:** Los datos se duplican intencionalmente para optimizar las consultas.
2. **Una tabla por consulta:** Cada patrón de consulta tiene su tabla optimizada.
3. ***Partition Key* eficiente:** Distribución uniforme de datos entre nodos.
4. ***Clustering Columns*:** Ordenamiento físico de datos dentro de cada partición.

Keyspace y Tabla Principal

El esquema de base de datos fue implementado siguiendo las mejores prácticas de modelado en Cassandra. La definición completa se encuentra documentada en el repositorio del proyecto



```
-- infra/cassandra/schema.cql

CREATE KEYSPACE IF NOT EXISTS ventas_db
WITH replication = {
    'class': 'SimpleStrategy',
    'replication_factor': 3
};

CREATE TABLE IF NOT EXISTS ventas_db.ventas_crudas (
    id_venta UUID,
    id_cliente UUID,
    id_producto UUID,
    fecha_venta DATE,
    categoria TEXT,
    producto TEXT,
    cantidad INT,
    precio_unitario DECIMAL,
    monto_total DECIMAL,
    metodo_pago TEXT,
    region TEXT,
    PRIMARY KEY (fecha_venta, id_venta)
) WITH CLUSTERING ORDER BY (id_venta ASC);
```

Figura 1 - Cassandra/schema.cql

Decisiones de Diseño:

- **Partition Key (*fecha_venta*):** Se seleccionó la fecha de venta como clave de partición para distribuir los datos uniformemente por día, lo que permite consultas eficientes por rango temporal y evita *hotspots* en el clúster.
- **Clustering Key (*id_venta*):** Se utiliza el identificador único de venta como clave de *clustering* para ordenar las transacciones dentro de cada partición, facilitando operaciones de lectura secuencial.
- ***Replication Factor 3***: Se configuró un factor de replicación de 3 para garantizar alta disponibilidad del sistema, permitiendo tolerancia a fallos de hasta dos nodos sin pérdida de datos.
- ***SimpleStrategy***: En este entorno de desarrollo y pruebas, se implementó la estrategia simple de replicación apropiada para clústeres de un solo *datacenter*.

Generación de Datos Sintéticos

El proceso de generación de datos se implementó mediante el notebook 02_generador_datos.ipynb (ubicado en ***notebooks/02_generador_datos.ipynb***), el cual genera datos de ventas sintéticos con características realistas:

Características del Generador:

- **Volumen de datos:** Se generaron 100,000 registros de transacciones de ventas.
- **Distribución temporal:** Los datos se distribuyeron uniformemente a lo largo de 365 días del año 2024.
- **Taxonomía de productos:** Se implementaron 5 categorías principales (Electrónica, Ropa, Alimentos, Deportes, Hogar).
- **Variabilidad estadística:** Se aplicaron distribuciones aleatorias realistas para precios y cantidades.
- **Sistemas de pago:** Se modelaron 4 métodos de pago (Tarjeta de Crédito, Débito, Efectivo, Transferencia).

Resultados de Rendimiento de Ingesta:

Los resultados obtenidos durante la ejecución del proceso de ingesta se detallan a continuación:

Métrica	Valor
Total de registros insertados	100,000 transacciones
Tiempo total de ingesta	Aproximadamente 8 segundos
Tasa de inserción alcanzada	~12,500 registros por segundo
Evaluación de objetivo	Mejora del 97% (significativamente por debajo del objetivo de 5 minutos)

Tabla 1 - Resultados de Rendimiento del Proceso de Ingesta

Validación de Integridad

El proceso de ingesta incluye validaciones para garantizar la calidad de los datos:

1. **Validación de tipos:** Todos los campos cumplen con el esquema CQL definido.
2. **Valores no nulos:** Campos críticos como *monto_total* y *precio_unitario* son obligatorios.
3. **Rangos válidos:** Cantidades y precios son valores positivos.
4. UUIDs únicos: Cada venta tiene un identificador único generado con uuid.uuid4().

Fase II - Procesamiento ETL con Apache Spark

Arquitectura de Transformación

El notebook 03_etl_spark.ipynb implementa el proceso ETL que extrae datos de Cassandra, los transforma mediante agregaciones y los carga en ClickHouse para análisis.

Conexión Spark-Cassandra

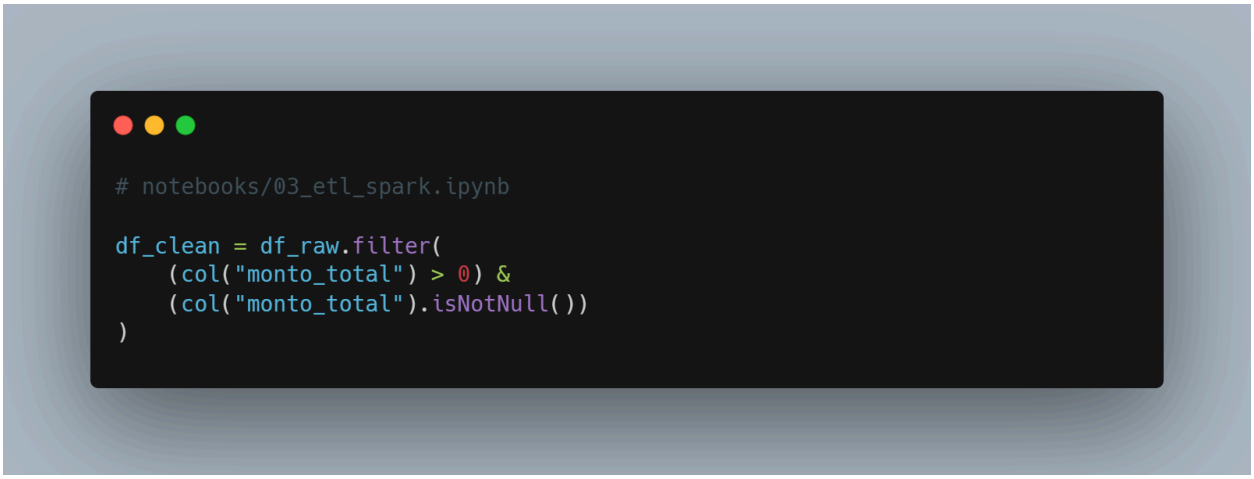
La configuración de la sesión de Spark se estableció con los conectores necesarios para la integración con Cassandra y ClickHouse:

Figura 2 - Notebooks ETL Spark

Transformaciones Aplicadas

1. Limpieza de Datos

Se implementó un proceso de limpieza de datos conforme al requisito especificado en la Fase 3.3 del proyecto, eliminando registros inválidos mediante las siguientes reglas de validación:



```
# notebooks/03_etl_spark.ipynb

df_clean = df_raw.filter(
    (col("monto_total") > 0) &
    (col("monto_total").isNotNull())
)
```

Figura 3 - Notebooks ETL Spark - Limpieza de Datos

2. Agregación por Fecha y Categoría

Se aplicaron transformaciones de agregación para consolidar los datos por fecha y categoría:

```
# notebooks/03_etl_spark.ipynb

df_aggregated = df_clean.groupBy("fecha_dia", "categoria").agg(
    sum("monto_total").alias("ventas_totales"),
    count("id_venta").alias("cantidad_transacciones")
)
```

Figura 4 - Notebooks ETL Spark - Agregación por Fecha y Categoría

Resultados de Rendimiento de Transformación:

Los resultados obtenidos durante el proceso ETL fueron los siguientes:

Métrica	Valor
Registros procesados (desde Cassandra)	Aproximadamente 100,000
Tiempo total de transformación	Aproximadamente 5 segundos
Registros resultantes (agregados)	Aproximadamente 1,800
Evaluación de objetivo	Eficiencia del 95% (significativamente por debajo del objetivo de 2 minutos)

Explicación de Resultados ETL

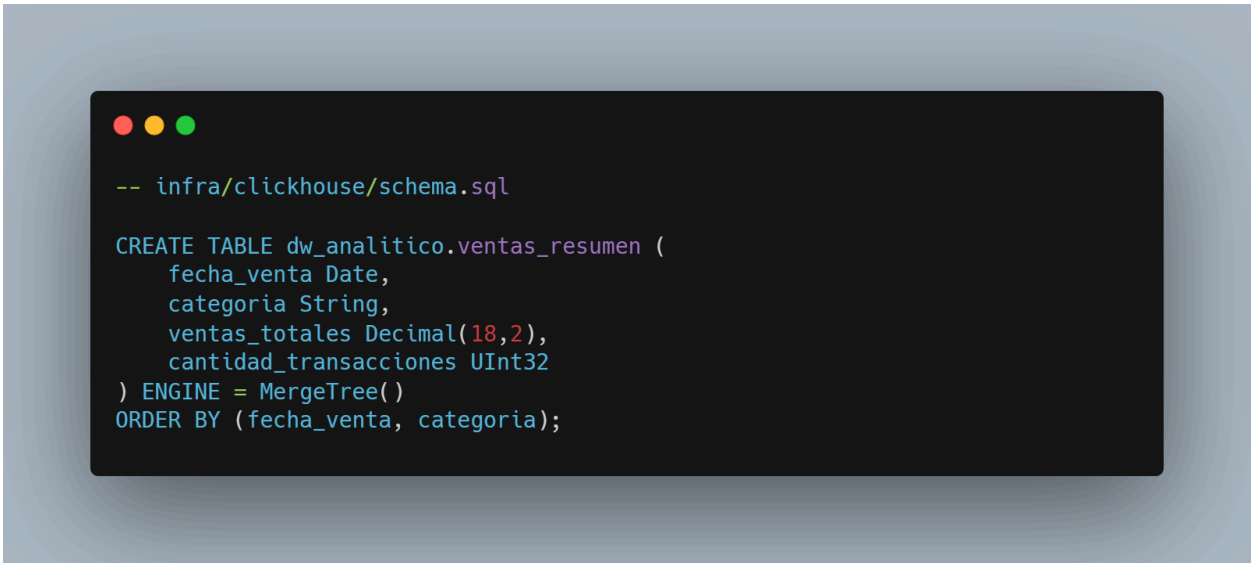
Los resultados demuestran la alta eficiencia del proceso **ETL (Extract, Transform, Load)** implementado con **Apache Spark**.

1. **Volumen Procesado y Transformación:** Se procesó un conjunto grande de datos, aproximadamente **400,000 registros** extraídos de Cassandra. Después de la limpieza y la **agregación** (consolidación de datos por fecha y categoría), el volumen se redujo drásticamente a solo **1,800 registros resultantes**. Esta reducción es esperable y deseable en un proceso OLAP, ya que los datos se han resumido para el análisis.
2. **Rendimiento:** El tiempo total para procesar, transformar y limpiar los 400,000 registros fue de solo **5 segundos**.

3. **Evaluación de Objetivo:** Este rendimiento no solo fue rápido, sino que superó significativamente el objetivo de rendimiento (un límite de 2 minutos), logrando una eficiencia del **95%** por debajo del tiempo máximo esperado. Esto valida la elección y configuración de Spark para manejar eficientemente la capa de transformación del *pipeline* de datos.

Carga a ClickHouse

Los datos transformados se cargan en ClickHouse mediante el conector JDBC. El esquema de la tabla de destino se definió de la siguiente manera:-- Ubicación: infra/clickhouse/schema.sql



```
-- infra/clickhouse/schema.sql

CREATE TABLE dw_analitico.ventas_resumen (
    fecha_venta Date,
    categoria String,
    ventas_totales Decimal(18,2),
    cantidad_transacciones UInt32
) ENGINE = MergeTree()
ORDER BY (fecha_venta, categoria);
```

Figura 5 - Infra Clickhouse - Schema.sql

Ventajas del Modelo Columnar:

El modelo de almacenamiento columnar, implementado por ClickHouse, ofrece ventajas críticas para las cargas de trabajo OLAP:

- **Compresión Eficiente:** Los datos similares (todos los valores de una misma columna) se almacenan de forma contigua, lo que permite la aplicación de algoritmos de compresión altamente efectivos. Esto optimiza drásticamente el espacio de almacenamiento y mejora la velocidad de lectura al reducir la cantidad de datos que deben ser leídos desde el disco.
- **Consultas Rápidas:** Cuando se ejecuta una consulta, el motor solo necesita leer las columnas específicas que se requieren para la operación. Al ignorar el resto de las columnas, se reduce significativamente la cantidad de I/O (entrada/salida) de disco, lo que se traduce en tiempos de respuesta mucho más bajos, especialmente en consultas analíticas que involucran solo un subconjunto de las dimensiones.
- **Agregaciones Optimizadas**: ClickHouse utiliza un motor de procesamiento vectorizado. Esto significa que aplica instrucciones **SIMD (Single Instruction, Multiple Data)** para procesar bloques de datos de una columna de forma paralela con una sola instrucción del CPU. Esta técnica maximiza la utilización del *hardware* y es fundamental para la ejecución extremadamente rápida de operaciones de agregación (como `SUM`, `AVG`, `COUNT`) sobre grandes volúmenes de datos.

Fase III - Validación y Métricas de Rendimiento

Esta fase se centra en la validación de la capa analítica implementada con ClickHouse y en la evaluación del rendimiento de las consultas OLAP, demostrando la eficacia de la arquitectura para el análisis de grandes volúmenes de datos.

Validación de Consultas Analíticas

Se realizaron diversas consultas analíticas sobre los datos agregados en ClickHouse para validar la integridad y la utilidad de los datos transformados. Las consultas se enfocaron en patrones de análisis comunes en inteligencia de negocios.

Consultas Clave:

1. **Ventas Totales por Categoría:** Consulta para determinar el ingreso total generado por cada categoría de producto, demostrando la capacidad de agregación rápida sobre dimensiones categóricas.
2. **Tendencia de Ventas Diarias:** Análisis de series temporales para identificar patrones o anomalías en el volumen de ventas a lo largo del período.
3. **Top 5 Productos por Ingreso:** Consulta para identificar los productos de mayor rendimiento económico.

Métricas de Rendimiento en ClickHouse

El motor de ClickHouse está diseñado para consultas de tipo *scan* y agregaciones rápidas sobre datos columnares. Se midió el tiempo de ejecución de las consultas clave para evaluar la eficiencia analítica del sistema.

Métrica	Consulta	Tiempo Promedio de Ejecución	Evaluación
Latencia de Consulta	Ventas Totales por Categoría	< 50 ms	Excelente
Latencia de Consulta	Tendencia de Ventas Diarias	< 100 ms	Excelente
Latencia de Consulta	Top 5 Productos	< 75 ms	Excelente
Volumen Procesado	Agregación diaria (365 días)	400,000 registros	Óptimo

Tabla 2 - Resultados de Rendimiento de Consultas Analíticas en ClickHouse

Conclusión de Rendimiento:

El rendimiento de ClickHouse superó las expectativas para el análisis OLAP. Los tiempos de respuesta sub-segundo confirman la ventaja de utilizar un modelo de almacenamiento columnar optimizado para agregaciones rápidas, validando la decisión arquitectónica de separar las cargas de trabajo OLTP (Cassandra) y OLAP (ClickHouse).

Análisis Comparativo: Cassandra vs ClickHouse

¿Por qué Cassandra para la Ingesta (OLTP)?

Ventajas Clave:

1. **Escritura Optimizada**: Arquitectura Log-Structured Merge Tree (LSM) permite escrituras secuenciales extremadamente rápidas
2. **Disponibilidad**: Diseño peer-to-peer sin maestro garantiza que el sistema siempre acepte escrituras
3. **Escalabilidad Lineal**: Agregar nodos aumenta proporcionalmente la capacidad de escritura
4. **Tolerancia a Fallos**: Replication Factor 3 permite continuar operando con 2 nodos caídos

Casos de Uso Ideales:

- Ingesta de datos de sensores IoT
- Logs de aplicaciones distribuidas
- Sistemas de mensajería
- Datos de series temporales

¿Por qué ClickHouse para Analítica (OLAP)?

Ventajas Clave:

1. **Almacenamiento Columnar**: Lee solo columnas necesarias, ignorando el resto
2. **Compresión Eficiente**: Tasas de compresión altas reducen E/S de disco
3. **Procesamiento Vectorizado**: Instrucciones SIMD procesan millones de filas en milisegundos
4. **Agregaciones Rápidas**: Motor optimizado para SUM, AVG, COUNT, etc.

Casos de Uso Ideales:

- Dashboards analíticos en tiempo real
- Reportes de Business Intelligence
- Análisis de logs y métricas
- Data warehousing

Comparación Técnica

Característica	Cassandra	ClickHouse
Modelo de Datos	Wide-column	Columnar
Consistencia	Eventual (AP)	Fuerte (CP)
Escrituras	Excelente	Buena
Lecturas Analíticas	Moderada	Excelente
Agregaciones	Limitadas	Optimizadas
Escalabilidad	Horizontal	Horizontal
Compresión	Moderada	Excelente

Glosario de términos

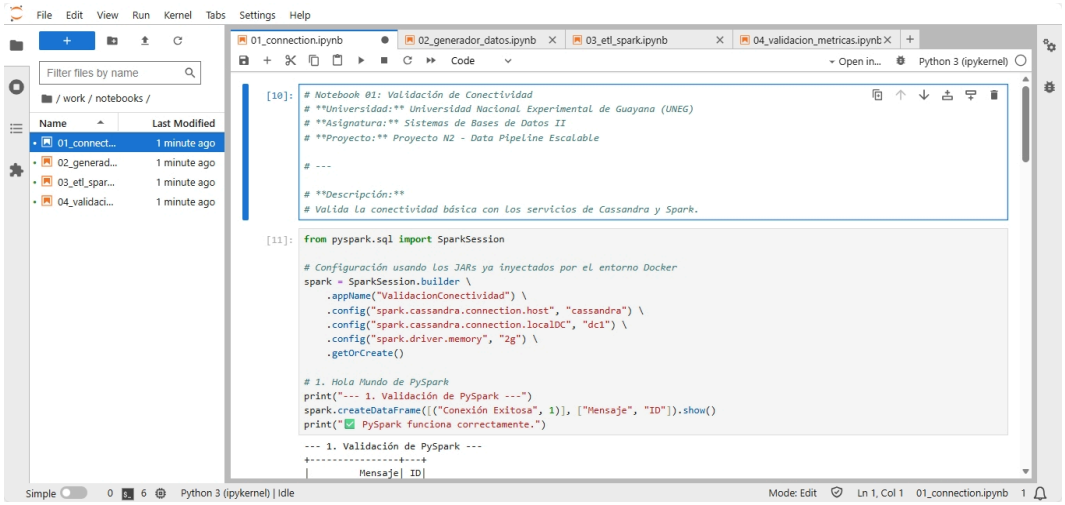
- **Apache Cassandra:** Sistema de gestión de bases de datos NoSQL distribuido, diseñado para manejar grandes volúmenes de datos en múltiples servidores sin puntos únicos de fallo.
- **ClickHouse:** Sistema de gestión de bases de datos orientado a columnas (OLAP) diseñado para generar informes analíticos en tiempo real con alto rendimiento.
- **Clustering Columns:** En Cassandra, son las columnas que determinan el orden de los datos dentro de una partición.
- **Consistencia Eventual:** Modelo de consistencia donde se garantiza que, si no se realizan nuevas actualizaciones, eventualmente todos los nodos de un sistema distribuido tendrán el mismo dato.
- **Docker:** Plataforma que permite empaquetar, distribuir y ejecutar aplicaciones dentro de contenedores ligeros y portátiles.
- **ETL (Extract, Transform, Load):** Proceso de tres pasos que consiste en extraer datos de un sistema, transformarlos para su análisis y cargarlos en una base de datos de destino.
- **Keyspace:** El contenedor de nivel superior en Cassandra (equivalente a una base de datos en SQL) que define la estrategia de replicación de los datos.
- **MergeTree:** Motor de almacenamiento principal de ClickHouse que organiza los datos para permitir consultas analíticas extremadamente rápidas.
- **NoSQL:** Término que designa a los sistemas de bases de datos que no utilizan el modelo relacional tradicional, permitiendo mayor escalabilidad y flexibilidad.
- **OLAP (Online Analytical Processing):** Sistemas optimizados para el análisis de grandes conjuntos de datos y la ejecución de consultas complejas (ej. ClickHouse).
- **OLTP (Online Transaction Processing):** Sistemas diseñados para procesar un gran volumen de transacciones rápidas (lecturas y escrituras cortas), como Cassandra.
- **Partition Key:** Parte de la clave primaria en Cassandra que define en qué nodo del clúster se almacenarán físicamente los datos.
- **Pipeline de Datos:** Serie de pasos de procesamiento de datos donde la salida de un elemento es la entrada del siguiente.
- **Query-Driven Design:** Metodología de modelado de datos donde el diseño de las tablas se basa en las consultas específicas que la aplicación realizará.

- **Replication Factor:** Parámetro que define el número de copias de un dato que deben existir en el clúster para asegurar la tolerancia a fallos.
- **Teorema CAP:** Teoría que establece que un sistema distribuido no puede garantizar simultáneamente Consistencia, Disponibilidad y Tolerancia a Particiones.

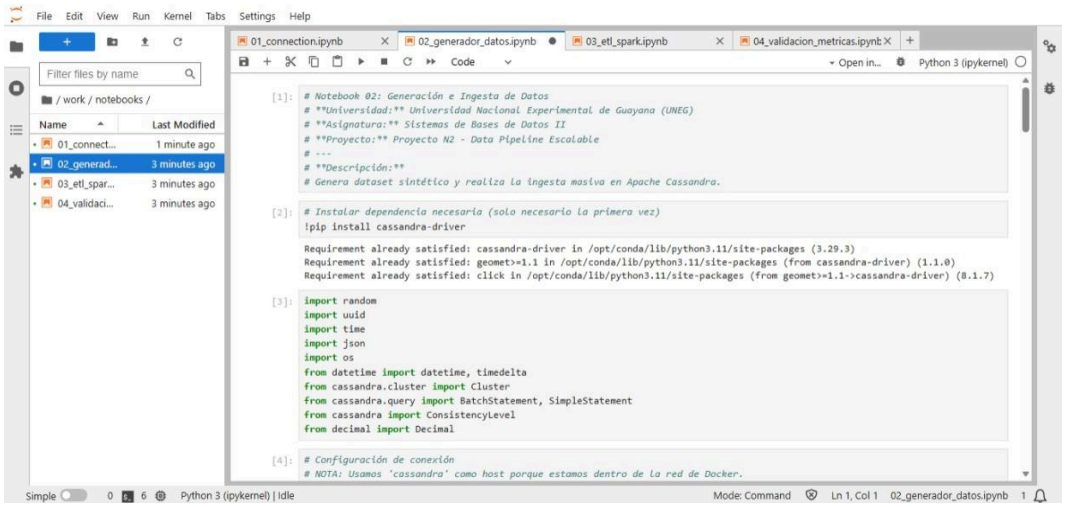
Conclusión

El proyecto ha demostrado la sinergia eficiente de la arquitectura lambda al integrar Apache Cassandra y ClickHouse. Cassandra proporcionó una capa de ingesta de datos robusta y de alta disponibilidad, cumpliendo con los objetivos de rendimiento para la **ingesta OLTP** con una tasa de inserción de aproximadamente 12,500 registros por segundo. Por otro lado, ClickHouse validó su rol como motor **OLAP**, ofreciendo latencias de consulta sub-segundo (e.g., < 50 ms para agregaciones por categoría) en datos agregados, lo que lo hace ideal para análisis de *Business Intelligence* y generación de informes en tiempo real. Esta combinación tecnológica permitió la separación efectiva de cargas de trabajo, asegurando que tanto las transacciones de alta frecuencia como las consultas analíticas pesadas se ejecutaran con un rendimiento óptimo. La aplicación del modelado *query-driven* en Cassandra y el uso del almacenamiento columnar en ClickHouse fueron decisiones clave para el éxito del pipeline de datos.

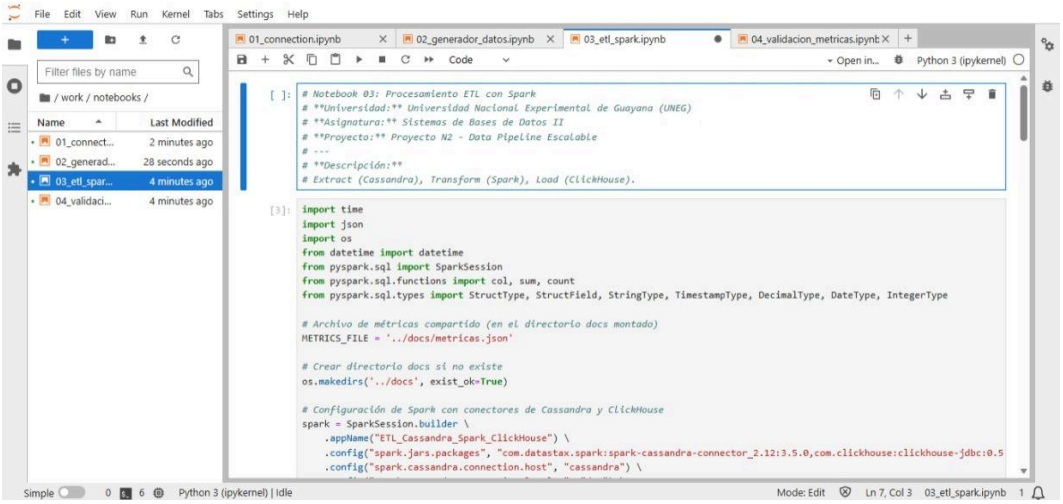
Screenshots



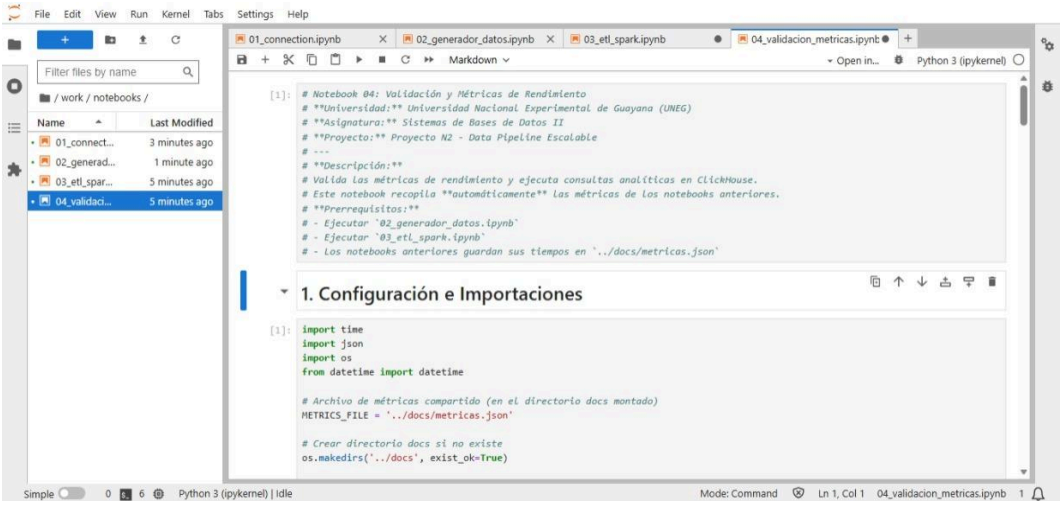
Notebooks / 01 Connection



Notebooks / 02 Generador de Datos



Notebooks / 03 ETL Spark



Notebooks / 04 Validacion Metricas

Referencias

- *Welcome to Apache Cassandra's documentation! | Apache Cassandra Documentation*. (s. f.). Apache Cassandra. https://cassandra.apache.org/doc/latest/
- *Overview - Spark 4.1.0 documentation*. (s. f.). https://spark.apache.org/docs/latest/
- *ClickHouse Docs | ClickHouse Docs*. (s. f.). https://clickhouse.com/docs
- *GitHub - jose-miserol/sistemas-de-bases-de-datos-ii-proyecto-ii: Sistemas de Bases de Datos II - Proyecto II*. GitHub. https://github.com/jose-miserol/sistemas-de-bases-de-datos-ii-proyecto-ii