BASES DE DATOS BASADAS EN COLUMNAS - PROYECTO 1

UNIVERSIDAD NACIONAL EXPERIMENTAL DE GUAYANA
DEPARTAMENTO DE CIENCIA Y TECNOLOGÍA
COORDINACION DE INGENIERIA INFORMÁTICA
SISTEMAS DE BASES DE DATOS II
SEMESTRE: 2025-I

AUTORES:	**PROFESOR:**
Shirley Cedeño C.I: 30.413.183	**CLINIA CORDERO**
Angel Rodriguez C.I: 30.437.205	

CIUDAD GUAYANA, 06 / 06 / 2025.

📘 Especificación y Diseño (Intensivo)

1. Modelo de Casos de Uso (Muy Básico)

El objetivo de esta sección es identificar las interacciones principales entre los usuarios y el sistema, enfocándose en los procesos clave que sustentan la funcionalidad general del sistema de recomendación de música.

Caso de Uso 1: Registrar una Escucha

Cuando un usuario reproduce una canción, el sistema debe registrar esta acción para su posterior análisis y generación de estadísticas. Se guarda el identificador del usuario, la canción que escuchó y la fecha del evento. Además, se actualizan automáticamente los contadores de reproducción para los reportes por género y país.

Elementos registrados:

- `usuario_id`
- `cancion_id`
- `fecha_escucha` (proporcionada por el sistema o el usuario)
- Actualización de:
  - `escuchas_por_genero`
  - `reporte_por_genero`
  - `reporte_por_pais`

Consulta de ejemplo en Cassandra:

```
INSERT INTO escuchas (usuario_id, cancion_id, fecha_escucha)
VALUES (101, 3, '2024-05-20');
```

Caso de Uso 2: Visualizar Reporte Mensual por Género

Descripción:

Este caso de uso permite a un administrador del sistema consultar un resumen mensual del número total de escuchas por cada género musical. Esta funcionalidad es parte del análisis OLAP simplificado, y permite visualizar el comportamiento agregado de los usuarios en el sistema.

Elementos consultados:

- `genero`
- `mes`
- `total_escuchas`

Consulta de ejemplo en Cassandra:

```
SELECT * FROM reporte_por_genero;
```

Caso de Uso 3: Consultar Reporte por País en un Mes Específico

Descripción:

Este caso de uso está destinado a los usuarios que desean analizar las escuchas registradas por país en un mes determinado. Como la tabla `reporte_por_pais` está organizada por país como clave primaria, se requiere usar `ALLOW FILTERING` para filtrar por el campo `mes`.

Elementos consultados:

- `pais`
- `mes`
- `total_escuchas`

Consulta de ejemplo en Cassandra:

```
SELECT * FROM reporte_por_pais WHERE mes = '2024-02' ALLOW FILTERING;
```

---

2. Modelo de Datos NoSQL

Se diseñó una estructura de almacenamiento NoSQL utilizando **Apache Cassandra**, una base de datos distribuida orientada a columnas, ideal para soportar altas tasas de escritura y consultas rápidas. El modelo se enfoca en permitir el registro eficiente de actividades de los usuarios y consultas analíticas básicas relacionadas con reproducciones musicales.

Tablas Definidas

2.1. Tabla `usuarios`

Contiene los datos básicos necesarios para identificar a cada usuario y su ubicación.

```
CREATE TABLE usuarios (
    usuario_id INT PRIMARY KEY,
    nombre TEXT,
    ciudad TEXT,
    pais TEXT
);
```

- **Clave primaria**: `usuario_id`
- **Propósito**: Identificar de forma única a cada usuario y permitir el análisis por país o ciudad.

2.2. Tabla `canciones`

Almacena la información esencial de cada canción disponible en el sistema.

```
CREATE TABLE canciones (
    cancion_id INT PRIMARY KEY,
    titulo TEXT,
    artista TEXT,
    genero TEXT
);
```

- **Clave primaria**: `cancion_id`
- **Propósito**: Clasificar las canciones por género y usarlas como referencia en reportes y recomendaciones.

2.3. Tabla `escuchas`

Registra cada vez que un usuario escucha una canción, con fecha incluida.

```
CREATE TABLE escuchas (
    usuario_id INT,
    fecha_escucha DATE,
    cancion_id INT,
    PRIMARY KEY (usuario_id, fecha_escucha, cancion_id)
) WITH CLUSTERING ORDER BY (fecha_escucha DESC);
```

- **Clave primaria compuesta**: `(usuario_id, fecha_escucha, cancion_id)`
- **Propósito**: Consultar el historial de reproducción de cada usuario en orden cronológico.

2.4. Tabla `escuchas_por_genero`

Tabla de agregación que mantiene el conteo total de escuchas por canción dentro de cada género.

```
CREATE TABLE escuchas_por_genero (
    genero TEXT,
    cancion_id INT,
    total_escuchas COUNTER,
    PRIMARY KEY (genero, cancion_id)
);
```

- **Propósito**: Soportar el algoritmo de recomendación que muestra las canciones más escuchadas por género.

2.5. Tabla `reporte_por_genero`

Almacena la cantidad total de reproducciones por género y por mes.

```
CREATE TABLE reporte_por_genero (
    genero TEXT,
    mes TEXT,
    total_escuchas COUNTER,
    PRIMARY KEY (genero, mes)
);
```

- **Propósito**: Permitir análisis de tendencias musicales por género en períodos mensuales.

2.6. Tabla `reporte_por_pais`

Registra el total de escuchas por país y mes, como parte del análisis geográfico simplificado.

```
CREATE TABLE reporte_por_pais (
    pais TEXT,
    mes TEXT,
    total_escuchas COUNTER,
    PRIMARY KEY (pais, mes)
);
```

- **Propósito**: Soportar reportes de distribución de escuchas por región geográfica.

Consideraciones de diseño

- Se emplean columnas del tipo `COUNTER` en las tablas de agregación (`escuchas_por_genero`, `reporte_por_genero`, `reporte_por_pais`) para garantizar actualizaciones incrementales y atómicas.
- La base de datos fue estructurada para responder rápidamente consultas típicas del sistema, como: recomendaciones por género, reportes por mes o país, y análisis del historial de usuarios.
- No se implementó una tabla combinada multidimensional para mantener el enfoque del proyecto en la simplicidad y claridad de las consultas OLAP básicas requeridas.

---

3. Requisitos OLAP (Simplificados)

Establecer los elementos clave para implementar un análisis OLAP básico dentro del sistema, empleando un enfoque simplificado y adaptado a las capacidades del modelo de datos en Apache Cassandra. Este análisis tiene como fin proporcionar una visión agregada del comportamiento de los usuarios en términos de consumo musical por género, periodo de tiempo y región geográfica.

Dimensiones

> Una dimensión es una perspectiva desde la cual se analiza un proceso de negocio.

En el contexto del sistema de recomendación, las dimensiones representan los ejes de análisis a partir de los cuales se pueden organizar y agrupar los datos. Se han definido las siguientes:

Dimensión	¿Qué representa?	Ejemplo de uso
Tiempo	Cuándo ocurrió la escucha	"¿Cuántas canciones se escucharon en enero?"
Género musical	Qué tipo de música se escuchó	"¿Qué género fue más popular en marzo?"
País	Dónde estaba el usuario que escuchó	"¿Qué escuchan más en Venezuela?"

3.1. Dimensión: Tiempo

- **Atributos**: `fecha_escucha`, `mes`
- **Jerarquía propuesta**: Esto permite hacer drill-down desde año hasta día o roll-up desde día hasta mes.

  ```
  Día → Mes → Año
  ```

- **Uso**: Permite analizar tendencias de escucha en distintos periodos de tiempo, facilitando comparaciones entre meses o años.

3.2. Dimensión: Género

- **Atributos**: `genero`
- **Jerarquía**: No jerárquica, todos los valores (Rock, Pop, Grunge) están al mismo nivel.

  ```
  Género → (no hay niveles)
  ```

- **Uso**: Agrupar y filtrar datos por estilos musicales, como Rock, Pop o Grunge.

3.3. Dimensión: País

- **Atributos**: `pais`
- **Jerarquía**: No jerárquica, en este caso solo se usa País, así que esta dimensión no tiene jerarquía activa (plana), aunque puede crecer en el futuro.

```
País → (no hay niveles)
```

- **Uso**: Permite segmentar el análisis por ubicación geográfica de los usuarios.

Hechos / Medidas

Los hechos representan las variables numéricas que serán objeto de análisis dentro del modelo OLAP.

Medida definida:

- **Nombre**: `total_escuchas`
- **Tipo**: Contador (`COUNTER`)
- **Descripción**: Representa la cantidad de veces que se ha registrado una reproducción de canciones, agregada según las combinaciones de las dimensiones.

> Nota: Se ha decidido utilizar una única medida para mantener la simplicidad del sistema, conforme al alcance definido en el proyecto.

---

4. Diseño del Cubo OLAP Conceptual

El cubo OLAP conceptual tiene como propósito permitir el análisis multidimensional de los datos de escucha musical almacenados en el sistema. Este diseño no se implementa directamente en una base de datos multidimensional, sino que se **simula utilizando tablas agregadas en Apache Cassandra**.

El cubo está centrado en una **única medida cuantitativa**, el número total de escuchas, y considera **tres dimensiones** principales: tiempo, género musical y país de origen del usuario.

Tabla de hechos

Hecho	Descripción
`total_escuchas`	Representa el número de veces que se ha registrado una escucha

Dimensiones

Dimensión	Descripción
Tiempo	Representa el momento de la reproducción
Género	Clasificación musical de la canción
País	Ubicación del usuario

Navegación en el Cubo: Operaciones OLAP posibles

- **Slice**: Filtrar por una dimensión específica

  Ejemplo: Ver todas las escuchas del género Rock.

- **Dice**: Filtrar por múltiples valores de múltiples dimensiones

  Ejemplo: Ver escuchas de Pop y Rock en Venezuela y Colombia durante enero.

- **Roll-up**: Agrupar a un nivel más general

  Ejemplo: Agrupar las escuchas diarias a nivel mensual.

- **Drill-down**: Desagrupar a un nivel más detallado

  Ejemplo: Desglosar el total mensual por día.

- **Pivot**: Cambiar la disposición visual de las dimensiones

  Ejemplo: Mostrar países como columnas y géneros como filas.

> En lugar de implementar un cubo OLAP físico como en una base de datos multidimensional (MOLAP o ROLAP), se optó por **simular el comportamiento OLAP** mediante la creación de **tablas agregadas en Cassandra**, que almacenan de forma anticipada las combinaciones clave de dimensiones y sus respectivas medidas. Este enfoque es coherente con las características del modelo de datos en Cassandra, donde las operaciones de agregación deben resolverse a través de diseño de esquema, y no mediante consultas dinámicas.

Ejemplo Conceptual del Cubo (Vista tabular simplificada)

Género	Mes	Venezuela	Colombia	México
Rock	2024-01	120	80	60
Pop	2024-01	200	150	100

Cada celda representa el valor de la medida `total_escuchas` para una combinación específica de género, mes y país.

---

5. Diseño General del Sistema (Capas Simplificadas)

El sistema de recomendación de música fue desarrollado siguiendo una arquitectura funcional segmentada en capas, lo que permite una clara separación de responsabilidades y facilita su mantenimiento, extensión y comprensión. Este diseño está centrado en la simplicidad, la compatibilidad local y el cumplimiento de los objetivos académicos del proyecto.

El sistema se estructura en **cuatro capas funcionales**:

1. Capa de Presentación (Interfaz de Usuario)

- **Responsabilidad**: Proporcionar una interfaz accesible desde el navegador donde los usuarios y administradores puedan interactuar con el sistema.
- **Implementación**: Interfaz web construida con **Flask** (microframework de Python), ejecutada de forma local. El contenido HTML está embebido directamente en la aplicación.
- **Funcionalidades ofrecidas**:
  - Registrar nuevas reproducciones de canciones manualmente.
  - Visualizar las canciones más escuchadas por género.

- Filtrar reportes de escuchas por mes o por país.
- Ejecutar consultas manuales en CQL.
- Ver resultados en **tablas HTML** estilizadas y **gráficos dinámicos** usando **Chart.js** (barras y pastel).

> La interfaz mantiene una estética simple pero funcional, totalmente integrada y ejecutable sin necesidad de herramientas externas. Las visualizaciones interactivas permiten interpretar fácilmente los datos.

---

2. Capa de Aplicación (Lógica del Negocio)

- **Responsabilidad**: Manejar las operaciones internas, procesar peticiones del usuario y comunicarse con la base de datos.
- **Implementación**: Desarrollada en **Python 3**, usando el driver oficial de Cassandra (`cassandra-driver`) para ejecutar operaciones CQL.
- **Tareas clave**:
  - Procesar la inserción de nuevas escuchas.
  - Actualizar automáticamente las tablas de agregación (`COUNTER`) al registrar nuevas reproducciones.
  - Ejecutar consultas OLAP básicas y recomendaciones.
  - Preparar los datos para su visualización en la interfaz web.
  - Interpretar y ejecutar consultas manuales desde la interfaz.

---

3. Capa de Datos (Base de Datos NoSQL)

- **Responsabilidad**: Gestionar el almacenamiento de datos operativos y agregados de manera eficiente.
- **Tecnología utilizada**: **Apache Cassandra**
- **Estructura**:
  - Tablas desnormalizadas, modeladas para facilitar lecturas específicas y rápidas.
  - Uso de `COUNTER` en tablas de análisis (`escuchas_por_genero`, `reporte_por_genero`, `reporte_por_pais`) para registrar y consultar métricas acumuladas.
  - Datos cargados inicialmente desde archivos `.csv` mediante scripts Python (`insert_data.py`).

---

4. Capa de Análisis OLAP (Básico y Preagregado)

- **Responsabilidad**: Permitir consultas analíticas básicas sobre el comportamiento de las escuchas, simulando un análisis multidimensional.

- **Implementación**:
  - Tablas `reporte_por_genero` y `reporte_por_pais` almacenan los totales de escuchas agregados por mes.
  - Estas estructuras actúan como una representación simplificada de un cubo OLAP, permitiendo consultas por dimensión (género, país) y por tiempo (mes).
- **Visualización de Resultados**:
  - Los datos se presentan en la interfaz web usando **gráficas interactivas** de Chart.js (barras para género, pastel para países).
  - Las respuestas a las consultas manuales también se muestran en tablas HTML dinámicas.

> Esta capa no realiza operaciones OLAP complejas como drill-down o roll-up, pero cumple con los objetivos del proyecto al facilitar consultas resumidas por dimensión y tiempo.

✅ Pruebas, Evaluación y Finalización

1. Pruebas Funcionales del Sistema

Se realizaron pruebas prácticas para verificar el correcto funcionamiento de cada uno de los componentes principales del sistema de recomendación musical. Estas pruebas se llevaron a cabo de forma manual mediante la interfaz web y mediante la ejecución de scripts de prueba.

1.1 Pruebas de Registro de Escuchas

Prueba	Acción	Resultado Esperado	Resultado Obtenido
P1	Insertar escucha: `usuario_id=101`, `cancion_id=1`, `fecha='2024-03-01'`	Registro exitoso en tabla `escuchas`; actualización de contadores en `escuchas_por_genero`, `reporte_por_genero`, `reporte_por_pais`	✅ Registro correcto y actualización exitosa
P2	Insertar escucha con fecha inválida	Rechazo de la operación	✅ Error controlado
P3	Insertar escucha con ID inexistente	No actualiza contadores por falta de datos de referencia	✅ Manejada sin fallo, no se actualizan contadores

1.2 Pruebas de Consultas OLAP

Prueba	Consulta	Resultado Esperado	Resultado Obtenido
P4	Reporte mensual por género	Lista de géneros con sus totales de escuchas por mes	✅ Datos mostrados correctamente
P5	Reporte mensual por país	Totales de escuchas agrupados por país y mes	✅ Consulta funcional con `ALLOW FILTERING` si se filtra por mes

P6	Top canciones por género (Pop)	Las 5 canciones más escuchadas del género Pop	✅ Resultado correcto con datos simulados

2. Evaluación Cualitativa de Recomendaciones

Se evaluó de forma cualitativa la lógica de recomendación usada en el sistema: **mostrar las canciones más escuchadas por género**. Si bien se trata de un enfoque sencillo, se verificó que:

- Las recomendaciones se basan en datos reales agregados (`escuchas_por_genero`).
- Las canciones más populares por género se reflejan adecuadamente.
- El sistema es capaz de mostrar recomendaciones ajustadas al género seleccionado por el usuario.

> Conclusión: La lógica de recomendación es simple pero efectiva para el propósito del proyecto académico. Se puede extender en el futuro con algoritmos más avanzados.

3. Evaluación de Consultas OLAP

Se evaluaron las consultas OLAP implementadas desde la perspectiva de usabilidad y rendimiento:

- Las consultas se ejecutan de forma eficiente gracias al uso de tablas pre-agregadas.
- Se logra un comportamiento similar al de un cubo OLAP conceptual mediante diseño de esquema.
- Las visualizaciones en **Chart.js** facilitan la interpretación de los resultados.

> Conclusión: Las operaciones OLAP básicas implementadas (reporte por género, reporte por país, recomendación por popularidad) cumplen su función educativa y técnica sin requerir un motor OLAP complejo.

4. Consideraciones Finales

- **Objetivos cumplidos**: Todos los objetivos del proyecto fueron abordados con éxito: modelado NoSQL, consultas OLAP básicas, sistema funcional y visualizaciones integradas.
- **Limitaciones conocidas**:
  - No se incluye validación exhaustiva de entradas en la interfaz.
  - El sistema se probó con un conjunto de datos pequeño y sintético.
  - No se implementaron filtros avanzados ni segmentación por ciudades u otros niveles jerárquicos.
- **Oportunidades de mejora**:
  - Incorporar filtros adicionales por ciudad, artista o usuario.

- Extender las operaciones OLAP con funciones como drill-down o roll-up real.
- Reemplazar `ALLOW FILTERING` por diseño de tablas adicionales para consultas más eficientes.

📄 Documentación Técnica del Proyecto

1. Tecnologías Utilizadas

Componente	Tecnología / Herramienta	Descripción
Base de Datos NoSQL	Apache Cassandra	Base de datos distribuida, orientada a columnas, ideal para grandes volúmenes de datos y escrituras rápidas.
Lenguaje de programación	Python 3.x	Utilizado para scripts de carga, lógica de aplicación y servidor web.
Framework web	Flask	Microframework para construir la interfaz web del sistema.
Visualización	Chart.js	Biblioteca JavaScript para gráficos interactivos en la interfaz web.
CQL (Cassandra Query Language)	Consultas NoSQL	Utilizado para crear tablas, insertar datos y realizar consultas en Cassandra.
Archivos CSV	`.csv`	Archivos con los datos de usuarios, canciones y escuchas para pruebas.

2. Estructura del Proyecto

```
/proyecto-musica/
│
├── data/
│   ├── usuarios.csv
│   ├── canciones.csv
│   └── escuchas.csv
│
├── 1_setup.py           # Crea el keyspace y las tablas en Cassandra
├── 2_insert_data.py     # Carga los datos desde los CSVs y actualiza contadores
├── 3_queries.py         # Ejecuta consultas desde consola (recomendaciones y reportes)
├── app.py               # Aplicación web Flask para interfaz y visualización
├── README.txt           # (Opcional) Instrucciones rápidas de ejecución
```

3. Requisitos del Sistema

- **Python** versión 3.8 o superior
- **Apache Cassandra** versión 3.x o superior instalado (Cassandra usado 3.11.10)

- **Bibliotecas Python**:
  - `cassandra-driver`
  - `Flask`
- **Navegador moderno** (Chrome, Firefox, Edge, etc.)

Para instalar las bibliotecas necesarias:

```
pip install cassandra-driver flask
```

---

4. Instrucciones de Ejecución

Paso 1: Iniciar Cassandra

Asegúrate de que el servicio de Cassandra esté en ejecución:

```
cassandra -f
```

Paso 2: Crear las tablas

Ejecuta el script `1_setup.py` para crear el keyspace y las tablas:

```
python 1_setup.py
```

Paso 3: Cargar los datos

Coloca los archivos `.csv` en la carpeta `data/` y ejecuta:

```
python 2_insert_data.py
```

Esto insertará usuarios, canciones, escuchas y actualizará las tablas `COUNTER`.

Paso 4: Ejecutar consultas desde consola (opcional)

```
python 3_queries.py
```

Paso 5: Iniciar la interfaz web

```
python app.py
```

Luego abre tu navegador en:

```
http://127.0.0.1:5000
```

Desde allí podrás:

- Consultar las canciones más escuchadas por género
- Ver reportes por mes y por país
- Ejecutar consultas CQL manuales
- Visualizar gráficos interactivos

---

5. Observaciones Técnicas

- **Persistencia distribuida**: Cassandra asegura que los datos se almacenen de forma distribuida (aunque en este caso se usó `SimpleStrategy` con `replication_factor=1` para pruebas locales).
- **Tablas COUNTER**: Solo aceptan columnas tipo `counter`, y sus valores no se pueden decrementar directamente.
- **ALLOW FILTERING**: Se utilizó en consultas específicas por limitaciones del modelo de clave primaria (en `reporte_por_pais`).
- **Seguridad**: El sistema es solo para fines académicos. No incluye autenticación, validación avanzada ni cifrado.
