



**UNIVERSIDAD HISPANOAMERICANA**

**ANÁLISIS DE DATOS**

**MÓDULO V:**

**ARQUITECTURA DE BASE DE DATOS**

**PROYECTO:**

**GLOBAL STREAM HUB**

**ESTUDIANTE:**

**DANIEL VÁSQUEZ GONZÁLEZ**

**PROFESOR:**

**LEONARDO FERNÁNDEZ AGUILAR**

**AÑO:**

**2025**

# 1. Introducción

Global Stream Hub es una plataforma de entretenimiento que ofrece contenido de video y música a usuarios a nivel global, operando con un modelo híbrido de suscripción premium sin publicidad y un plan de bajo costo financiado con anuncios. La base de datos diseñada en este proyecto tiene como objetivo soportar de forma robusta y escalable la operación de la plataforma, así como sus necesidades analíticas y de reporting.

La solución se implementa en PostgreSQL 17 y está organizada siguiendo la arquitectura Medallón (Bronze, Silver, Gold), incorporando mecanismos de auditoría, manejo de historiales, vistas de negocio y vistas materializadas para optimizar el acceso a datos de alto valor.

## 2. Escenario de negocio

Global Stream Hub debe manejar grandes volúmenes de datos relacionados con usuarios, contenido, actividad de reproducción, publicidad y suscripciones. Entre las operaciones clave se encuentran:

- Gestión de usuarios, sus perfiles y planes de suscripción.
- Catálogo jerárquico de contenido multimedia (películas, series, videos, canciones).
- Registro del historial de reproducciones y actividad por dispositivo.
- Gestión de listas de reproducción y contenido favorito.
- Sistema de publicidad con campañas, anuncios, impresiones, clics y segmentación por tipo de contenido.

Los requerimientos de negocio incluyen tanto transacciones operativas (ingesta de logs, actualizaciones de suscripciones) como análisis agregados (popularidad de contenido, métricas de uso, rendimiento de anuncios).

## 3. Arquitectura de la Solución

### 3.1 Arquitectura Medallón

Se adoptó una arquitectura Medallón con tres capas: Bronze, Silver y Gold.

- **Capa Bronze (Raw / Ingesta Cruda)**
  - Propósito: almacenar los datos tal como llegan desde las fuentes, sin limpieza ni transformación, de forma inmutable.
  - Implementación: tablas en el esquema `bronze` que reflejan la estructura de los archivos de entrada (`user_registrations`, `raw_streaming_logs`, `raw_catalog_data`, `raw_ad_impressions`, `raw_subscription_data`).
- **Capa Silver (Clean / Normalizada)**
  - Propósito: limpiar, estandarizar y enriquecer los datos provenientes de Bronze, aplicando reglas de negocio y normalización.

- Implementación: tablas relacionales normalizadas en el esquema `silver` (usuarios, suscripciones, contenido, géneros, artistas, sesiones de streaming, dispositivos, playlists, campañas y anuncios).
- **Capa Gold (Curated / Analítica)**
  - Propósito: optimizar la estructura de datos para análisis de negocio, BI y consumo por APIs, usando modelos dimensionales y agregaciones.
  - Implementación: esquema `gold` con tablas de hechos y dimensiones, vistas de negocio y vistas materializadas para métricas de alto nivel.

Esta arquitectura separa claramente ingesta, limpieza y explotación analítica, lo que facilita la evolución del modelo y el gobierno de datos.

## 4. Diseño de la Base de Datos

### 4.1 Capa Bronze

La capa Bronze almacena datos crudos que simulan archivos CSV/JSON de entrada.

#### Tablas principales (ejemplos):

- `bronze.user_registrations`: nuevos usuarios con datos básicos, país y tipo de suscripción.
- `bronze.raw_streaming_logs`: logs de reproducción (usuario, contenido, timestamps, dispositivo, porcentaje visto).
- `bronze.raw_catalog_data`: catálogo de contenido (id, título, tipo, género, artista/director, año, descripción).
- `bronze.raw_ad_impressions`: logs de impresiones de anuncios (timestamp, `ad_id`, `user_id`, `content_id`, `placement`).
- `bronze.raw_subscription_data`: información cruda de suscripciones.

#### Decisiones de diseño:

- Se modelan tablas con tipos flexibles, siguiendo de cerca la estructura de los archivos de origen.
- No se aplican constraints estrictos en Bronze para no bloquear la ingesta, dejando la validación para Silver.

### 4.2 Capa Silver (Modelo Normalizado)

La capa Silver implementa un modelo relacional normalizado, separando entidades maestras, relaciones y hechos operativos.

#### Grupos de tablas:

- **Usuarios y suscripciones:**
  - `silver.users`, `silver.countries`, `silver.subscription_plans`, `silver.subscriptions`.

- Justificación: separar usuarios, país y planes facilita cambios en catálogos y evita duplicidad de datos.
- **Contenido multimedia:**
  - `silver.content`, `silver.genres`, `silver.content_genres`, `silver.artists_directors`, `silver.content_artists`.
  - Justificación: un contenido puede tener múltiples géneros y múltiples artistas/directores, lo que requiere tablas puente N:N.
- **Actividad de streaming y dispositivos:**
  - `silver.streaming_sessions`, `silver.devices`.
  - `streaming_sessions` incluye duración, porcentaje visto y relación con usuario y contenido.
- **Playlists:**
  - `silver.playlists`, `silver.playlist_items` para modelar listas de reproducción y sus elementos.
- **Publicidad:**
  - `silver.ad_campaigns`, `silver.ads`, `silver.ad_impressions` cubren campañas, anuncios individuales y sus impresiones.

#### Decisiones de diseño:

- Se usan secuencias (\*\_seq) para gestionar claves primarias numéricas, evitando dependencia de SERIAL y facilitando control explícito.
- Se aplican claves foráneas entre usuarios, contenido, sesiones, anuncios y campañas para mantener integridad referencial.
- Se calculan campos derivados como duración de sesión y porcentaje visto a partir de logs crudos en las transformaciones Bronze→Silver.

### 4.3 Capa Gold (modelo dimensional)

La capa Gold organiza los datos en un modelo dimensional (Star Schema) para acelerar consultas analíticas.

#### Dimensiones:

- `gold.dim_users`: snapshot analítico de usuarios, país, plan actual y atributos relevantes.
- `gold.dim_content`: información de contenido, tipo, género principal, año, artista/director.
- `gold.dim_devices`: tipo de dispositivo y categoría (móvil, escritorio, TV, etc.).
- `gold.dim_genres`: géneros normalizados.
- `gold.dim_date`: dimensión calendario (fecha, año, mes, trimestre, indicador de fin de semana).
- `gold.dim_time_of_day`: hora del día y franja horaria (mañana, tarde, noche).

#### Tablas de hechos:

- `gold.fact_user_activity`: métricas agregadas de actividad de usuarios (sesiones, duración, tasa de completado) por usuario, contenido, fecha y franja horaria.

- `gold.fact_content_popularity`: popularidad de contenido por fecha, con vistas, usuarios únicos y tiempo visto.
- `gold.fact_ad_performance`: rendimiento de anuncios y campañas (impresiones, clics, CTR) por contenido, fecha y usuario.

### **Justificación:**

- El modelo dimensional simplifica y acelera consultas típicas de BI (por ejemplo, “horas vistas por país y género por mes”).
- Separar dimensiones de hechos evita duplicidad de descripciones y favorece la creación de dashboards en herramientas como Power BI.

## **5. Transformaciones y ETL**

El flujo Bronze→Silver→Gold se implementa mediante scripts SQL de transformación.

- **De Bronze a Silver:**
  - Limpieza de datos (filtrado de registros inválidos, normalización de códigos de país, tipos de contenido).
  - Parseo y conversión de timestamps, cálculo de duración y porcentajes.
  - Carga de tablas maestras (usuarios, países, planes, contenido, géneros, artistas) y de tablas de hechos operativos (sesiones, impresiones).
- **De Silver a Gold:**
  - Población de dimensiones a partir de tablas Silver.
  - Agregación de métricas por usuario, contenido, fecha, género, dispositivo y campaña.
  - Cálculo de KPIs como horas vistas, tasa de completado y CTR.

Las transformaciones están encapsuladas en scripts idempotentes que pueden ejecutarse de forma repetida sin romper la consistencia.

## **6. Vistas y vistas materializadas**

### **6.1 Vistas de negocio**

Se definieron vistas en `gold` para exponer información lista para uso por aplicaciones y analistas.

Vistas clave exigidas por el enunciado:

- `vw_user_profile_details`: perfil completo de usuario, plan de suscripción y última actividad.
- `vw_content_catalog_summary`: resumen del catálogo con tipo, género principal y popularidad.
- `vw_user_playback_history`: historial de reproducciones de un usuario con detalles de sesiones.
- `vw_playlist_details`: contenido de playlists incluyendo información de cada ítem.

- `vw_ad_performance_summary`: rendimiento de anuncios y campañas (impresiones, clics, CTR, tiempo de despliegue).

Estas vistas simplifican consultas complejas combinando múltiples tablas en una sola interfaz lógica.

## 6.2 Vistas materializadas

Se diseñaron vistas materializadas para métricas de alto costo de cómputo:

- `mv_monthly_user_activity`: horas vistas por usuario al mes y métricas de engagement.
- `mv_content_popularity_by_genre`: reproducciones y popularidad por contenido y género.
- `mv_ad_performance_metrics`: impresiones, clics y CTR por campaña/anuncio.
- `mv_daily_platform_metrics`: KPIs diarios (DAU, sesiones, horas de streaming, métricas de ads).

Existe una función central para refrescar todas las vistas materializadas, pensada para ser programada con un scheduler como `pg_cron` en un entorno productivo.

## 7. Auditoría y log histórico

### 7.1 Requerimientos

El enunciado exige pistas de auditoría, clasificación de datos sensibles y un log histórico de transacciones basado en triggers y JSONB.

### 7.2 Implementación

- Tablas de auditoría dedicadas (`audit.history_users`, `audit.history_subscriptions`, `audit.history_content`, etc.).
- Triggers `AFTER INSERT OR UPDATE OR DELETE` que capturan el estado `OLD` y `NEW` de las filas y lo almacenan en columnas JSONB.
- Campos estándar en tablas maestras (`created_at`, `updated_at`, `updated_by`) para trazabilidad básica.

Esto permite reconstruir la historia de cambios de entidades críticas y auditar quién modificó qué y cuándo.

## 8. Seguridad y roles

Se diseñaron roles alineados con el principio de mínimo privilegio.

- `etl_process_role`:
  - Lectura sobre Bronze, escritura/control sobre Silver.
  - Pensado para procesos ETL y cargas batch.

- `bi_analyst_role`:
  - Solo lectura sobre la capa Gold (tablas, vistas y vistas materializadas).
  - Enfocado en usuarios que construyen reportes y dashboards.
- `api_consumer_role`:
  - Lectura sobre vistas de negocio en Gold.
  - Diseñado para servicios de API que exponen datos a aplicaciones externas.

Los privilegios sobre schemas, tablas y secuencias se asignan explícitamente a cada rol, evitando que usuarios no autorizados accedan a datos sensibles.

## 9. Consideraciones de particionamiento

El enunciado propone el particionamiento como opción para tablas muy grandes (logs de streaming, impresiones de anuncios). En este proyecto se deja documentada la estrategia, pero sin implementarla en los scripts para mantener el alcance manejable.

Estrategia recomendada:

- Particionar `streaming_sessions` y `ad_impressions` por rango de fecha (mensual o anual), lo que mejoraría rendimiento y manejo de datos históricos.

## 10. Conclusiones y trabajo futuro

La solución implementada cumple con los requerimientos del proyecto: arquitectura Medallón completa, modelo normalizado en Silver, modelo dimensional en Gold, sistema de auditoría y vistas optimizadas para análisis y APIs.

Líneas de mejora futura:

- Implementar particionamiento en tablas de alto volumen.
- Integrar `pg_cron` o un orquestador externo para automatizar refresco de vistas materializadas y procesos ETL.
- Exponer vistas de negocio a través de una API REST y construir dashboards en herramientas de BI.
- Extender el modelo para soportar recomendaciones personalizadas y A/B testing de anuncios.