Plan de Trabajo del Proyecto Final

Bases de datos no relacionales - 2025

1. Identificación del grupo

- **Número del grupo**: 8
- **Integrantes (nombre completo y número de cédula)**: Joana Auriello (5.004.559-1) - Pablo Molina (4.675.210-2)

2. Familia de proyecto elegida

La familia de proyecto elegida será la Familia 1: Migración entre modelos. Se eligió esta familia por tratarse de un escenario de migración real desde un modelo relacional clásico hacia un modelo documental orientado a optimizar consultas y escalabilidad.

3. Objetivos del proyecto

Objetivo general:

Migrar un modelo de base de datos relacional sobre visualizaciones de contenido en plataformas de streaming hacia una estructura documental en MongoDB, optimizando la organización, eficiencia de consultas y flexibilidad del sistema, con el foco en la escalabilidad de la solución.

Objetivos específicos:

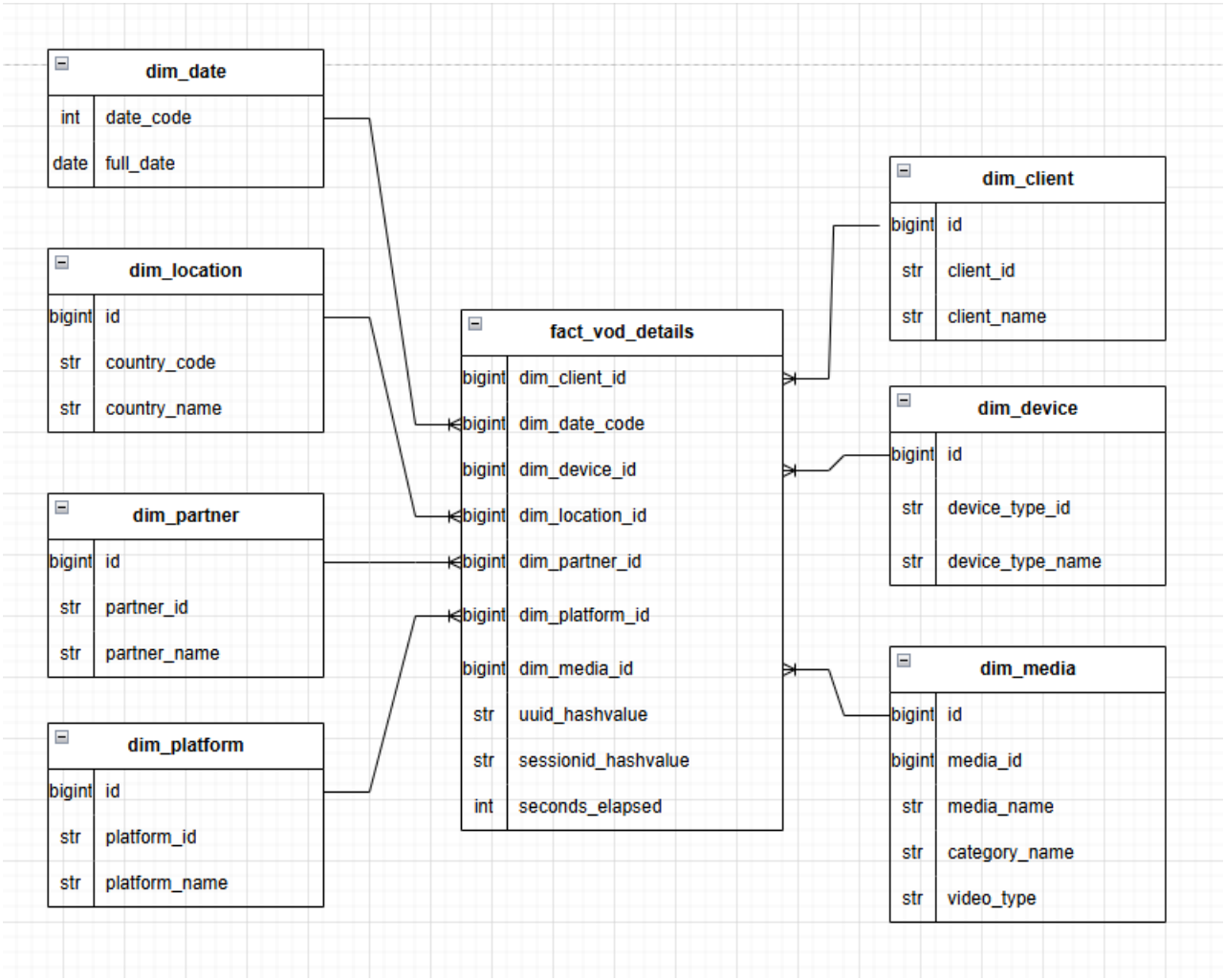
- Analizar la estructura actual del modelo entidad-relación y sus usos operativos.
- Definir un modelo documental adecuado, evaluando cuándo embeder y cuándo referenciar.
- Aplicar patrones de modelado como Computed Pattern, Extended Reference y Subset Pattern.
- Minimizar la necesidad de joins mediante desnormalización controlada.
- Diseñar consultas representativas sobre el nuevo modelo para validar su rendimiento.
- Documentar el proceso completo y justificar las decisiones tomadas para facilitar futuras implementaciones.

4. **Descripción general del proyecto**

La base de datos seleccionada modela la actividad de una empresa que ofrece plataformas de streaming como servicio a terceros interesados en distribuir contenido audiovisual. La estructura actual está diseñada en un esquema relacional clásico con una tabla de hechos (fact_vod_details) que centraliza el registro de visualizaciones, y múltiples dimensiones asociadas (cliente, marca, ubicación, socio, plataforma y contenido).

El objetivo del proyecto es migrar este modelo relacional a una estructura documental en MongoDB, aplicando buenas prácticas de modelado no relacional. Se busca reducir la dependencia de uniones complejas (joins), optimizar el acceso a la información relevante en consultas frecuentes, y mantener la flexibilidad para escalar en volumen o estructura sin comprometer el rendimiento. La reorganización estará orientada a mejorar el tiempo de respuesta y permitir la consulta de datos solo cuando sea necesario, apoyándose en patrones como embedding, referenciación extendida y campos precalculados según corresponda.

A continuación se muestra la estructura actual del modelo relacional:



5. **Actividades previstas y cronograma tentativo + Plan técnico de Tareas por etapa**

Completar la tabla con las actividades previstas para alcanzar los objetivos, estimando una duración tentativa para cada una (total aproximado: 60 horas).

Actividad	Breve descripción	Semana(s) prevista(s)	Plan técnico
Análisis del modelo relacional	Revisión del esquema actual (modelo entidad-relación), identificación de entidades, relaciones y patrones de uso clave para el rediseño	Semana 1	Explorar el modelo entidad-relación; extraer claves primarias/foráneas; mapear tablas fact/dimensión.
Definición de requerimientos de modelado	Establecer los requisitos del modelo documental: consultas frecuentes, relaciones críticas, campos sensibles, volumen de datos y patrones esperados	Semana 1	Identificar consultas clave; determinar cardinalidades; estimar volumen de datos por colección.
Diseño del modelo documental inicial	Propuesta de estructura MongoDB basada en el análisis anterior: decidir qué entidades embebidas, qué referencias mantener y cómo representar relaciones	Semana 2	Diseñar colecciones y estructuras embebidas; definir índices y claves externas si aplica; revisar límites de documentos.
Migración del modelo relacional	Transformación de datos desde el modelo relacional hacia documentos MongoDB, generando colecciones y poblando datos simulados	Semana 3	Exportar datos de origen (CSV/JSON); crear scripts de transformación (pandas o ETL); importar con mongoimport.
Aplicación de patrones de optimización	Incorporación de patrones como Computed Pattern, Extended Reference, Subset Pattern u otros para optimizar consultas o simplificar el acceso	Semana 4	Revisar puntos críticos de consulta; aplicar patrones de agregación, duplicación o resumen según necesidad.
Desarrollo de consultas representativas	Construcción de queries agregadas y de búsqueda que validen la utilidad del modelo y midan el rendimiento frente a las necesidades del negocio	Semana 5	Escribir queries con $match, $group, $sort, $unwind; validar tiempos de ejecución con explain().
Documentación y validación del modelo	Registro de decisiones de diseño, ventajas del nuevo modelo, limitaciones detectadas y validación funcional del esquema con ejemplos concretos	Semana 5	Redactar decisiones de diseño; documentar el modelo y ejemplos de uso; validar resultados esperados con usuarios.

Actividad	Breve descripción	Semana(s) prevista(s)	Plan técnico
Redacción del informe y presentación	Elaboración del informe final del proyecto y preparación de la presentación para docentes y compañeros	Semana 6	Unificar hallazgos; diseñar presentación visual del modelo y consultas; practicar exposición técnica.

6. Base teórica o artículos relacionados

¿El proyecto se inspira en algún artículo académico o proyecto existente?

La temática de los datos tiene similitud con la base MFix de MongoDB, aunque no se busca replicar este diseño, pueden existir coincidencias entre ambas soluciones.

Las referencias a utilizar serán las mismas que se utilizaron en el Laboratorio 1.

- Documentación MongoDB Atlas - Sample Mflix Dataset: Sample Mflix Dataset - MongoDB Atlas - MongoDB Docs
- Documentación MongoDB Atlas - $search **score** :
    - https://www.mongodb.com/docs/manual/core/indexes/index-types/index-text/control-text-search-results/#:~:text=When%20MongoDB%20returns%20text%20search,first%20in%20the%20result%20set.
    - https://www.mongodb.com/docs/atlas/atlas-search/scoring/
- Documentación MongoDB Atlas - Indices: Indexes - PyMongo Driver v4.11 - MongoDB Docs
- Documentación MongoDB Atlas - $unwind: https://www.mongodb.com/docs/manual/reference/operator/aggregation/unwind/
- Documentación MongoDB Atlas - $gte: $gte - Database Manual v8.0 - MongoDB Docs
- Documentación MongoDB Atlas - Resumen de patrones usuales: Building with Patterns: A Summary | MongoDB
- Documentación MongoDB Atlas - The Computed Pattern: Building With Patterns: The Computed Pattern | MongoDB

7. Datos y entornos

- Juegos de datos a utilizar (si ya se definieron):
  - La estructura de datos a utilizar corresponde a una estructura de datos reales utilizada en nuestros trabajos aunque se obtendrá una submuestra con datos anonimizados con el fin de simplemente hacer el ejercicio de rediseño que pueda potencialmente llegar a ser implementado a futuro.
- Entornos de prueba y herramientas: MongoDB Atlas como motor documental, Google Colaboratory para procesamiento en Python (pymongo, pandas)

8. Otras observaciones (opcional)

Cualquier comentario adicional que quieran incluir (dudas, riesgos previstos, aspectos a discutir con docentes, etc.).

Evaluación comparativa y alternativas de modelado

Comparación de rendimiento

Para validar si el rediseño propuesto en MongoDB efectivamente mejora los tiempos de respuesta, se propone una etapa de evaluación comparativa utilizando un conjunto de consultas representativas. Estas consultas se medirán sobre:

- Modelo original relacional (por ejemplo, en PostgreSQL o MySQL).
- Modelo rediseñado en MongoDB.

Las métricas a comparar incluirán:

- Tiempo promedio de respuesta.
- Volumen de datos transferido.
- Complejidad de la consulta (cantidad de uniones, agregaciones, etc.).

Consultas comparativas sugeridas:

Consulta	**SQL (relacional)**	**Equivalencia en MongoDB (NoSQL) [se completara más adelante]**
Top 10 contenidos del primero al 10 de mayo según total de tiempo visto	SELECT dm.video_type, dm.media_name,	

	SUM(fvd.seconds_elapsed) AS total_seconds FROM prod.fact_vod_uu_details AS fvd INNER JOIN prod.dim_media AS dm ON fvd.dim_media_id = dm.id INNER JOIN prod.dim_date AS dd ON fvd.dim_date_code = dd.date_code WHERE dd.full_date BETWEEN '20250501' AND '20250510' AND dm.client_id='tg' AND dm.video_type!='Linear' GROUP BY dm.video_type, dm.media_name ORDER BY total_seconds DESC LIMIT 10 ;	
Tiempo total visto por cliente en la última semana	SELECT dc.client_name, SUM(fvd.seconds_elapsed) AS total_seconds FROM prod.fact_vod_uu_details AS fvd INNER JOIN prod.dim_client AS dc ON fvd.dim_client_id = dc.id INNER JOIN prod.dim_date AS dd ON fvd.dim_date_code = dd.date_code INNER JOIN prod.dim_media AS m ON m.id=fvd.dim_media_id WHERE dd.full_date BETWEEN '20250501' and '20250510' AND dc.client_id='tg' AND m.video_type!='Linear' GROUP BY dc.client_name ORDER BY total_seconds DESC; ;	

Cantidad de Usuarios que vieron contenido por día	SELECT dd.full_date, COUNT(DISTINCT uuid_hashvalue) AS total_users FROM prod.fact_vod_uu_details AS fvd INNER JOIN prod.dim_date AS dd ON fvd.dim_date_code = dd.date_code INNER JOIN prod.dim_client AS c ON c.id=fvd.dim_client_id INNER JOIN prod.dim_media AS m ON m.id=fvd.dim_media_id WHERE dd.full_date BETWEEN '20250501' AND '20250510' AND c.client_id='tg' AND m.video_type!='Linear' GROUP BY dd.full_date ORDER BY dd.full_date ASC;	
Tiempo total visto por plataforma y cliente (principales plataformas utilizadas)	SELECT dc.client_name, dp.partner_name, SUM(fvd.seconds_elapsed) AS total_seconds FROM prod.fact_vod_uu_details AS fvd INNER JOIN prod.dim_date AS dd ON dd.date_code=fvd.dim_date_code INNER JOIN prod.dim_client AS dc ON fvd.dim_client_id = dc.id INNER JOIN prod.dim_media AS m ON m.id=fvd.dim_media_id INNER JOIN prod.dim_partner AS dp ON fvd.dim_partner_id = dp.id WHERE dd.full_date BETWEEN '20250501' AND '20250510' AND dc.client_id='tg' AND m.video_type!='Linear'	

	GROUP BY dc.client_name, dp.partner_name ORDER BY total_seconds DESC;	

Este análisis permitirá cuantificar la ganancia (o no) que representa el uso de una base no relacional en este contexto.

Para que la comparación sea válida, todas las pruebas deben realizarse bajo condiciones equivalentes de hardware, red y volumen de datos. Se propone utilizar una única instancia virtualizada para correr ambos motores (MongoDB y PostgreSQL) con datasets de igual tamaño y origen.

Exploración del uso de bases de datos orientadas a grafos

Dado que el caso de uso involucra múltiples relaciones entre usuarios, contenidos y plataformas, también se evaluó conceptualmente el uso de bases de datos de grafos (por ejemplo, Neo4j). Este enfoque sería especialmente útil si se incorporan mecanismos de recomendación o análisis de comunidad.

Modelo grafo sugerido:

- **Nodos**: Usuario, Contenido, Plataforma.
- **Relaciones**:
    - (:Usuario)-[:VIO]->(:Contenido)
    - (:Contenido)-[:PERTENECE_A]->(:Cliente)
    - (:Contenido)-[:SIMILAR_A]->(:Contenido) (para recomendaciones)

La mayor limitante en este caso es que por ser datos reales, tratamos con los usuarios de forma anonimizada. Lo cual implicaría que no tendríamos más información de los usuarios que su identificador (hashvalue). Esto seria especialmente interesante para campañas de marketing dirigidas, por ejemplo. Presentamos esta opción simplemente a modo de reflejar las distintas opciones que el dataset y modelo inicial nos presentan, pero por temas de mantener datos anónimos y que estaremos trabajando con muestras de los datos completos se trabajara exclusivamente sobre el modelo de bases de datos documentales (no relacional).

Modelado

Estructura de colecciones

Video_views (colección principal)

```
{
  "user_id": "d7fea5...",
  "session_id": "a1b2c3...",
  "date": {
    "date_code": 20240511,
    "full_date": "2024-05-11"
  },
  "seconds_elapsed": 1200,

  "client": {
    "client_id": "tg",
    "client_name": "Toon Goggles"
  },
  "device": {
    "device_type_id": "androidtv",
    "device_type_name": "Android TV"
  },
  "location": {
    "country_code": "IN",
    "country_name": "India"
  },
  "partner": {
    "partner_id": "jio",
    "partner_name": "Jio STB"
  },
  "platform": {
    "platform_id": "html5",
    "platform_name": "HTML5"
  },
  "media": {
    "media_id": 280657,
    "media_name": "The Tractor",
    "category_name": "animation",
    "video_type": "Episode"
  }
}
```


- Esta colección es el núcleo de análisis. Permite consultas agregadas por usuario, cliente, contenido, categoría, etc. Contiene los datos transaccionales y es el eje de todas las consultas analíticas.
- Incluye los datos embebidos de todas las dimensiones + la fecha con código y formato legible.
- Optimizado para consultas por fecha, contenido, usuario, dispositivo, etc.

Justificación

Descripción: Cada documento representa un evento de visualización de contenido por parte de un usuario en una sesión determinada.

¿Por qué es la colección principal?

- Es la tabla de hechos en el modelo relacional (fact_vod_details).

- Contiene todas las relaciones de uso entre usuarios, contenidos, plataformas y fechas.
- Es el punto de entrada para casi todas las consultas analíticas.

¿Por qué se embeben las dimensiones (client, device, etc.)?

- Tienen pocos atributos y no cambian frecuentemente (son slow changing dimensions en el modelo relacional).
- Embebidos mejoran la performance evitando joins ($lookup).
- Facilita consultas agregadas sobre una sola colección (por ejemplo: "cuántas visualizaciones hubo por categoría de contenido, por cliente y por día").

¿Por qué se incluye date como subdocumento?

- Las consultas por tiempo son muy comunes (última semana, mes, día).
- Se evita tener que hacer transformaciones de fechas o joins con una dim_date.

Users

```
{
  "user_id": "d7fea5...",
  "countries": ["IN", "BD"],
  "devices": ["Mobile", "SmartTV"]
}
```

Datos agregados de usuario, identificados por uuid_hashvalue. Permite segmentar por país o dispositivo, para hacer perfilado o clustering, por ejemplo.

Justificación

Descripción: Representa usuarios identificados por su user_id (uuid_hashvalue). Contiene agregados útiles como historial de países o dispositivos.

¿Por qué usar una colección aparte?

- No todos los análisis se hacen a nivel de evento individual. Algunas técnicas (como clustering de usuarios, recomendaciones o perfiles) requieren vista consolidada del usuario.
- Permite enriquecer el sistema con segmentación o cohortes.

¿Por qué no embebido en viewings?

- Un mismo usuario puede tener cientos o miles de eventos. No es eficiente duplicar sus datos.

clients (apps o marcas)

```
{
  "client_id": "tg",
  "client_name": "Toon Goggles",
  "partners": [
    { "partner_id": "FireTV", "partner_name": "Amazon" },
    { "partner _id": "androidtv", " partner _name": "Android TV" }
  ]
}
```

Representa las aplicaciones o marcas. Incluye los nombres y partners disponibles por cliente.

Justificación

Descripción: Representa las aplicaciones o marcas (clientes) desde donde los usuarios acceden al contenido.
¿Por qué tener esta colección?

- Aunque los datos de cliente están embebidos en cada viewing, esta colección sirve para mantener una fuente canónica de nombres, plataformas, etc.
- Puede usarse en interfaces administrativas o dashboards.

¿Por qué embebés plataformas en cada cliente?

- Cada cliente tiene un conjunto limitado de plataformas. Es interesante para analizar a que plataformas se esta distribuyendo su aplicación y hacer análisis periódicos de rendimiento por plataforma, ya que en casos particulares, comparten ganancias los clientes con las plataformas según las visualizaciones.

media


```
{
  "media_id": 280657,
  "media_name": "The Tractor",
  "category_name": "animation",
  "video_type": "Episode"
}
```


Contiene metadatos de los contenidos: nombre, categoría, tipo.

Justificación

Descripción: Contiene metadatos de los contenidos disponibles (nombre, categoría, tipo).
¿Por qué tener una colección separada?

- Aunque los metadatos están embebidos en viewings, esta colección sirve para:
  - Proveer una fuente centralizada de datos del catálogo.
  - Evitar duplicación excesiva si se necesita actualizar información de un contenido específico.
  - Soportar otras funcionalidades como dashboards de catálogo, navegación de contenido, etc. (Los clientes revisan su catalogo con gran frecuencia dependiendo de su performance, es importante que puedan acceder a sus catálogos de forma fácil y rápida).

Principios que guiaron estas decisiones

- Desnormalización controlada: Se embeben atributos estables que se consultan frecuentemente.

- Evitar joins costosos: Al tener todos los datos relevantes en un solo documento (viewings), se evita la necesidad de $lookup.

- Escalabilidad: Datos crecientes como visualizaciones están separados de datos estáticos como media, client.

- Flexibilidad para análisis: Permite hacer agregaciones complejas por cliente, categoría, país, usuario, fecha, etc., sin sacrificar rendimiento.