Sistema de Gestión de Estudios Médicos Distribuido

Segundo Proyecto – Bases de Datos II

Víctor Gabriel Mejías Salas
Jonathan Sancho
Angie Aguilar Alemán
José Arrienta

Fecha Inicio: Martes 07 de octubre 2025
Fecha de Entrega: Martes 21 de octubre 2025

Índice

1. Introducción

El presente documento describe el desarrollo e implementación de un sistema de bases de datos distribuidas para la gestión de estudios médicos, combinando tres motores de bases de datos: PostgreSQL, Microsoft SQL Server y MongoDB. El objetivo principal es garantizar interoperabilidad entre sistemas, integridad de la información y soporte a aplicaciones web y móviles, enfocándose en la infraestructura de datos y los roles de usuario.

2. Arquitectura General

2.1. Servidor PostgreSQL – Núcleo Transaccional

PostgreSQL actúa como el motor principal transaccional, almacenando datos operativos de pacientes, citas, resultados de estudios, diagnósticos y órdenes médicas. Se implementó segmentación horizontal para distribuir los pacientes por sucursal, permitiendo consultas eficientes y operaciones de inserción, actualización y eliminación frecuentes.

2.1.1. Segmentación Horizontal

- pacientes_san_jose
- pacientes_alajuela
- pacientes_cartago

2.1.2. Comunicación Distribuida

Se utilizan **Foreign Data Wrappers (FDW)** para conectarse con:

1. **SQL Server** (`tds_fdw`) para sincronizar datos de facturación y convenios.
2. **MongoDB** (`mongo_fdw`) para exportar resúmenes de estudios clínicos y generar reportes analíticos.

2.2. Servidor SQL Server – Integración Externa

SQL Server centraliza la información de hospitales, aseguradoras y sistemas públicos. Su segmentación es vertical, almacenando solo columnas relevantes para interoperabilidad:

- id_paciente
- id_aseguradora
- codigo_externo
- estado_facturacion

Se utiliza como fuente de datos federada para PostgreSQL mediante FDW.

2.3. Servidor MongoDB – Capa Analítica

MongoDB almacena datos desnormalizados provenientes de PostgreSQL. Su objetivo es:

- Reducir carga transaccional en PostgreSQL.
- Facilitar consultas agregadas y reportes médicos.
- Almacenar documentos por tipo de estudio, médico o rango de fechas.

Nota técnica: debido a dificultades técnicas insalvables con la extensión `mongo_fdw` (problemas de compatibilidad con la versión de MongoDB y PostgreSQL, errores de serialización y problemas con tipos de datos JSON complejos), se decidió simular la integración de MongoDB para el proyecto, garantizando que la lógica y consultas sean demostradas correctamente. Se probó la conexión directa con varios métodos, incluyendo:

- `mongo_fdw` oficial y forks disponibles.
- Exportación a JSON y carga en PostgreSQL.
- Middleware intermedio en Python.

Todas las pruebas fallaron o complicaron excesivamente la implementación, por lo que la simulación permite demostrar la funcionalidad sin comprometer la estabilidad del proyecto.

3. Gestión de Usuarios y Seguridad

Se configuraron dos niveles de roles:

3.1. Perfiles Internos

- **usr_fdw_pg_mssql**: lectura y escritura controlada hacia SQL Server.
- **usr_fdw_pg_mongo**: lectura y escritura controlada hacia MongoDB (simulado).

3.2. Perfiles Externos (Aplicaciones)

- **usr_api_web**: acceso a vistas de datos para la web.
- **usr_api_mobile**: acceso a vistas simplificadas para móviles.

Cada rol tiene permisos mínimos necesarios y políticas de autenticación diferenciadas, siguiendo las buenas prácticas de seguridad.

4. Pruebas de Integración y Resultados

4.1. Consulta de estado de facturación de pacientes

Se realizó una consulta desde PostgreSQL hacia SQL Server usando FDW:

Cuadro 1: Estado de facturación de pacientes desde SQL Server

ID Paciente	Nombre	Facturación	Estado
1	José Pérez	50000	Pagado
2	Ana Gómez	35000	Pendiente
3	Luis Fernández	42000	Pagado

Explicación: Esta prueba demuestra que PostgreSQL puede consultar correctamente los datos verticalmente segmentados de SQL Server a través del FDW.

4.2. Simulación de resultados clínicos en MongoDB

Dado que la integración real no fue posible, se creó una tabla simulada en PostgreSQL que refleja cómo se almacenarían los documentos en MongoDB:

Cuadro 2: Resultados simulados de estudios clínicos (MongoDB)

ID Estudio	Paciente	Tipo	Resultado	Fecha
12345	José Pérez	Ultrasonido	Normal	2025-10-07
12346	Ana Gómez	Radiografía	Fractura	2025-10-08
12347	Luis Fernández	Laboratorio	Alto Colesterol	2025-10-08

Explicación: Se validó la lógica de inserción, consulta y generación de reportes simulando la capa analítica de MongoDB, demostrando cómo se integraría con PostgreSQL.

5. Decisiones de Diseño

- **Segmentación Horizontal vs Vertical:** Horizontal para datos de pacientes (alta frecuencia de consultas e inserciones), vertical para facturación externa (solo columnas relevantes).
- **FDW:** Selección de tds_fdw para SQL Server por estabilidad y soporte. Mongo_fdw simulado para evitar errores críticos.
- **Roles y Seguridad:** Roles diferenciados para proteger datos y permitir acceso controlado a aplicaciones externas.
- **Simulación MongoDB:** Se documenta claramente para demostrar conocimiento y funcionalidad del flujo distribuido.

6. Conclusión

El proyecto demuestra la viabilidad de un sistema distribuido de bases de datos médicas combinando PostgreSQL, SQL Server y MongoDB (simulado). Se implementaron consultas federadas, segmentación de datos, roles de seguridad y simulaciones que garantizan que la arquitectura es funcional y segura.

A. Anexo A: Scripts PostgreSQL y FDW

```sql
-- Ejemplo de FDW hacia SQL Server
CREATE SERVER sqlserver_ext
    FOREIGN DATA WRAPPER tds_fdw
    OPTIONS (servername 'localhost', port '1433', database '
    IntegracionExterna');

CREATE FOREIGN TABLE paciente_integracion_fdw (
    id_paciente INT,
    id_aseguradora INT,
    codigo_externo VARCHAR(50),
    estado_facturacion VARCHAR(20)
)
SERVER sqlserver_ext
OPTIONS (schema_name 'dbo', table_name 'paciente_integracion');

CREATE FOREIGN TABLE citas_fdw (
    id_cita INT,
    id_paciente INT,
    fecha DATE,
    tipo_estudio VARCHAR(50)
)
SERVER sqlserver_ext
OPTIONS (schema_name 'dbo', table_name 'citas');

CREATE FOREIGN TABLE facturas_externas_fdw (
    id_factura INT,
    id_paciente INT,
    monto DECIMAL,
    fecha DATE
)
SERVER sqlserver_ext
OPTIONS (schema_name 'dbo', table_name 'facturas_externas');
```

Listing 1: Creación de FDW y tablas

B. Anexo B: Scripts de Roles y Vistas

```sql
-- Roles Internos
CREATE ROLE usr_fdw_pg_mssql LOGIN PASSWORD 'Fdwmssql123';
GRANT USAGE ON FOREIGN SERVER sqlserver_ext TO usr_fdw_pg_mssql;
GRANT SELECT, INSERT, UPDATE ON paciente_integracion_fdw TO
    usr_fdw_pg_mssql;
GRANT SELECT, INSERT, UPDATE ON citas_fdw TO usr_fdw_pg_mssql;
GRANT SELECT, INSERT, UPDATE ON facturas_externas_fdw TO
    usr_fdw_pg_mssql;

CREATE ROLE usr_fdw_pg_mongo LOGIN PASSWORD 'Fdwmongo123';
GRANT USAGE ON FOREIGN SERVER mongo_ext TO usr_fdw_pg_mongo;
GRANT SELECT, INSERT, UPDATE ON resultados_mongo_fdw TO usr_fdw_pg_mongo
    ;

-- Roles Externos
CREATE ROLE usr_api_web LOGIN PASSWORD 'Apiweb123';
```

```
CREATE VIEW vista_web_pacientes AS SELECT * FROM
   paciente_integracion_fdw;
GRANT SELECT ON vista_web_pacientes TO usr_api_web;

CREATE ROLE usr_api_mobile LOGIN PASSWORD 'Apimobile123';
CREATE VIEW vista_mobile_citas AS SELECT * FROM citas_fdw;
GRANT SELECT ON vista_mobile_citas TO usr_api_mobile;
```

Listing 2: Configuración de roles internos y externos

C. Anexo C: Simulación MongoDB

```
CREATE TABLE resultados_mongo_fdw (
    id_estudio INT,
    paciente VARCHAR(100),
    tipo VARCHAR(50),
    resultado VARCHAR(100),
    fecha DATE
);

INSERT INTO resultados_mongo_fdw VALUES
(12345,'Jos  P rez','Ultrasonido','Normal','2025-10-07'),
(12346,'Ana G mez','Radiograf a','Fractura','2025-10-08'),
(12347,'Luis Fern ndez','Laboratorio','Alto Colesterol','2025-10-08');
```

Listing 3: Tabla simulada para MongoDB