Sistema de Agendamiento de Citas ElectroHuila: Arquitectura de Base de Datos Oracle con Entity Framework Core y Arquitectura Limpia

Juan Esteban Huertas
SENA
Colombia, Colombia
jehuertas89@soy.sena.edu.co

Resumen

Este artículo presenta el diseño e implementación de la arquitectura de base de datos del sistema de agendamiento de citas para ElectroHuila, una empresa de servicios públicos que gestiona solicitudes de Peticiones, Quejas y Reclamos (PQR). El sistema emplea Oracle Database como gestor principal, implementado mediante Entity Framework Core 9.0 bajo los principios de Arquitectura Limpia y Domain-Driven Design. Se describe un esquema relacional de 22 tablas normalizadas a Tercera Forma Normal (3NF), optimizado para consultas de alto rendimiento mediante índices estratégicos y patrones de acceso a datos. La arquitectura contempla características avanzadas como soft deletes, auditoría temporal, conversión de tipos booleanos para Oracle, y convenciones de nomenclatura en mayúsculas. El diseño soporta múltiples gestores de bases de datos (Oracle, SQL Server, PostgreSQL, MySQL) mediante abstracciones del ORM, facilitando la portabilidad y escalabilidad del sistema. Los resultados demuestran una estructura robusta, mantenible y escalable que satisface los requerimientos de integridad referencial, seguridad de datos y rendimiento transaccional del dominio de agendamiento de citas.

Keywords

base de datos Oracle, Entity Framework Core, arquitectura limpia, diseño de esquemas, normalización, agendamiento de citas, sistemas PQR

1. Introducción

El sistema de agendamiento de citas ElectroHuila representa un caso de estudio significativo en el diseño de bases de datos para sistemas de gestión de Peticiones, Quejas y Reclamos (PQR) en empresas de servicios públicos. La complejidad inherente a este dominio requiere una arquitectura de datos robusta que soporte múltiples dimensiones: gestión de citas, usuarios con diferentes roles y permisos, sucursales distribuidas geográficamente, categorización de servicios, y trazabilidad completa de las solicitudes ciudadanas.

El diseño de la base de datos enfrentó varios desafíos técnicos fundamentales. Primero, la necesidad de modelar relaciones complejas entre 22 entidades del dominio, manteniendo la integridad referencial y la consistencia transaccional. Segundo, la implementación sobre Oracle Database, que requiere adaptaciones específicas como la conversión de tipos booleanos a NUMBER(1) y el uso de convenciones de nomenclatura en mayúsculas. Tercero, la aplicación de principios de Arquitectura Limpia y Domain-Driven Design, que demandan una separación clara entre el modelo de dominio y la persistencia física.

El esquema relacional diseñado contempla las siguientes áreas funcionales principales:

Gestión de Identidad y Acceso: Tablas para usuarios, roles, permisos, y asignaciones rol-permiso, implementando un modelo de control de acceso basado en roles (RBAC).

Gestión de Citas: Entidades para citas, estados de citas, y cancelaciones, con campos de auditoría temporal y soft deletes para trazabilidad histórica.

Gestión de Clientes: Información de clientes, tipos de clientes, y relaciones con sus solicitudes PQR.

Organización Territorial: Sucursales, departamentos, ciudades y sus interrelaciones jerárquicas.

Servicios y Categorías: Clasificación de servicios ofrecidos, categorías PQR, y tipos de PQR.

La implementación se realizó mediante Entity Framework Core 9.0, aprovechando su capacidad de abstracción para soportar múltiples proveedores de bases de datos (Oracle, SQL Server, PostgreSQL, MySQL), mientras se mantiene un código de dominio agnóstico a la tecnología de persistencia. Este enfoque garantiza portabilidad y facilita la evolución futura del sistema.

2. Implementación

La implementación de la arquitectura de base de datos del sistema ElectroHuila se estructuró en cuatro fases fundamentales, aplicando principios de diseño de bases de datos relacionales y patrones de arquitectura empresarial.

2.1. Diseño del Esquema Relacional

El esquema de base de datos se diseñó siguiendo un proceso riguroso de análisis del dominio y normalización. Las 22 tablas principales se organizaron en grupos funcionales cohesivos:

Módulo de Seguridad (5 tablas): USERS, ROLES, PERMISSIONS, USER_ROLES, ROLE_PERMISSIONS. Implementa un sistema RBAC completo con relaciones muchos-a-muchos entre usuarios-roles y roles-permisos. Cada usuario puede tener múltiples roles, y cada rol múltiples permisos granulares.

Módulo de Citas (3 tablas): APPOINTMENTS, APPOINTMENT_STATUSES, APPOINTMENT_CANCELLATIONS. La tabla APPOINTMENTS actúa como entidad central del sistema, relacionándose con clientes, sucursales y servicios. Se incluyen campos de auditoría (CreatedAt, UpdatedAt, DeletedAt) y un campo booleano IsDeleted para implementar soft deletes.

Módulo de Clientes (2 tablas): CLIENTS, CLIENT_TYPES. Gestiona la información de los ciudadanos que solicitan citas, con clasificación por tipo de cliente (persona natural, empresa, entidad pública).

Módulo Territorial (3 tablas): BRANCHES, DEPARTMENTS, CITIES. Modela la estructura geográfica de las sucursales de ElectroHuila, con relaciones jerárquicas departamento-ciudad-sucursal.

Módulo de Servicios (3 tablas): SERVICES, PQR_CATEGORIES, PQR_TYPES. Clasifica los servicios disponibles y las categorías de PQR (Peticiones, Quejas, Reclamos).

Tablas de Configuración (6 tablas): GENERAL_SETTINGS, NOTIFICATIONS, AUDIT_LOGS, entre otras. Soportan funcionalidades transversales del sistema.

El proceso de normalización alcanzó la Tercera Forma Normal (3NF), eliminando dependencias transitivas y garantizando la integridad de datos. Sin embargo, se aplicaron desnormalizaciones estratégicas en campos de uso frecuente (como nombres de cliente en la tabla de citas) para optimizar el rendimiento de consultas de lectura.

2.2. Configuración de Entity Framework Core

La implementación del ORM se realizó mediante configuraciones Fluent API en clases separadas, siguiendo el patrón de configuración por entidad. Aspectos clave incluyen:

Conversión de tipos booleanos para Oracle: Se implementó una conversión automática de propiedades bool a NUMBER(1), dado que Oracle no soporta nativamente el tipo BOOLEAN en versiones anteriores a 23c. Esto se logró mediante value converters personalizados.

Convenciones de nomenclatura: Se configuró el modelo para usar nombres de tabla y columna en mayúsculas, cumpliendo con las convenciones estándar de Oracle Database (APPOINTMENTS, USER_ID, IS_DELETED).

Índices de rendimiento: Se definieron índices en claves de negocio frecuentemente consultadas: Email en USERS, DocumentNumber en CLIENTS, AppointmentDate en APPOINTMENTS, y claves foráneas de todas las relaciones.

Configuración de relaciones: Se establecieron relaciones uno-a-muchos y muchos-a-muchos con cascada de eliminación configurada según las reglas de negocio. Por ejemplo, eliminar un cliente no elimina sus citas históricas (soft delete), pero eliminar un rol elimina sus asignaciones de permisos.

2.3. Implementación del Patrón Repository

Se implementó el patrón Repository genérico con especializaciones por entidad, proporcionando una abstracción entre la lógica de negocio y el acceso a datos. Los repositorios incluyen:

Métodos de consulta con AsNoTracking para operaciones de solo lectura, mejorando el rendimiento.

Especificaciones (Specification pattern) para construir consultas complejas de forma reutilizable.

Soporte para paginación, filtrado y ordenamiento mediante expresiones lambda.

Manejo transaccional mediante Unit of Work, garantizando atomicidad en operaciones que afectan múltiples tablas.

2.4. Estrategias de Optimización

Se aplicaron varias técnicas de optimización de rendimiento:

Consultas proyectadas: Uso de Select para recuperar solo los campos necesarios, reduciendo el tráfico de red.

Carga eager vs lazy: Configuración selectiva de Include para evitar el problema N+1 en relaciones.

Consultas compiladas: Pre-compilación de consultas frecuentes para reducir overhead de análisis.

Particionamiento lógico: Uso de DeletedAt y IsDeleted para separar registros activos de históricos sin comprometer la integridad referencial.

3. Discusión

La arquitectura de base de datos implementada para el sistema ElectroHuila presenta varias decisiones de diseño significativas que merecen análisis detallado, considerando sus ventajas, limitaciones y alternativas.

3.1. Decisiones de Diseño y Trade-offs

Normalización vs. Rendimiento: La decisión de normalizar a 3NF proporciona beneficios claros en integridad de datos y reducción de redundancia. Sin embargo, esto implica un mayor número de JOINs en consultas complejas. Para mitigar este impacto, se aplicaron desnormalizaciones controladas en campos de alta frecuencia de lectura, como nombres de cliente en la tabla de citas. Esta decisión reduce el número de JOINs en las consultas más comunes (listado de citas) a expensas de mayor complejidad en las actualizaciones.

Soft Deletes vs. Hard Deletes: La implementación de soft deletes mediante los campos IsDeleted y DeletedAt permite mantener un historial completo de datos, esencial para auditorías y análisis retrospectivos. El costo es un incremento en el tamaño de la base de datos y la necesidad de filtrar registros eliminados en cada consulta. Se implementaron query filters globales en Entity Framework Core para automatizar este filtrado, reduciendo la posibilidad de errores.

Timestamps de Auditoría: Los campos CreatedAt, UpdatedAt y DeletedAt en todas las tablas principales proporcionan trazabilidad temporal completa. Esta decisión incrementa el ancho de cada fila en aproximadamente 24 bytes, pero es fundamental para cumplir con requerimientos de auditoría del sector público.

3.2. Oracle Database vs. Alternativas

La elección de Oracle Database como gestor principal se basó en varios factores:

Ventajas: Rendimiento probado en sistemas empresariales de alta concurrencia, características avanzadas de seguridad (Virtual Private Database, Data Redaction), capacidades de particionamiento para escalabilidad horizontal, y soporte técnico empresarial.

Desafíos: Costo de licenciamiento significativo, curva de aprendizaje más pronunciada, particularidades como la ausencia de tipo BOOLEAN nativo (resuelta mediante conversión a NUMBER(1)), y complejidad en la configuración de alta disponibilidad.

Comparación con PostgreSQL: PostgreSQL ofrece características similares sin costo de licencia, soporte nativo de tipos de datos

más amplio (incluyendo BOOLEAN, JSON, arrays), y una comunidad activa. Sin embargo, Oracle proporciona mejor rendimiento en operaciones de escritura intensiva y transacciones complejas, según benchmarks de TPC-C.

Comparación con SQL Server: SQL Server presenta mejor integración con ecosistemas Microsoft y herramientas de desarrollo .NET. La elección de Oracle se justifica por su independencia de plataforma y mayor presencia en el sector público colombiano.

3.3. Entity Framework Core: Ventajas y Limitaciones

El uso de Entity Framework Core como ORM proporciona abstracción de base de datos y productividad en desarrollo, pero introduce consideraciones importantes:

Ventajas: Code-first migrations facilitan el versionamiento del esquema, LINQ proporciona type safety en consultas, y el patrón Unit of Work simplifica el manejo transaccional. La portabilidad multi-proveedor resultó exitosa en pruebas con cuatro DBMS diferentes.

Limitaciones: El overhead de abstracción genera consultas SQL menos optimizadas que las escritas manualmente. En consultas críticas de rendimiento, puede ser necesario usar SQL directo mediante FromSqlRaw. El seguimiento de cambios (change tracking) consume memoria significativa en operaciones masivas, mitigado mediante AsNoTracking.

3.4. Escalabilidad y Consideraciones Futuras

El diseño actual soporta adecuadamente la carga esperada inicial (estimada en 1,000 citas diarias), pero requiere consideraciones para escalabilidad:

Particionamiento horizontal: Las tablas de citas y auditoría son candidatas para particionamiento por rango de fechas, separando datos históricos de activos.

Réplicas de lectura: Implementar réplicas de solo lectura para consultas de reportes y análisis, descargando la base de datos principal.

Caché distribuido: Datos de catálogo (departamentos, ciudades, servicios) pueden almacenarse en Redis para reducir consultas a la base de datos.

3.5. Seguridad y Cumplimiento Normativo

El modelo RBAC implementado proporciona control de acceso granular, cumpliendo con principios de least privilege. Sin embargo, para cumplir completamente con la Ley 1581 de 2012 (Protección de Datos Personales en Colombia), se requieren implementaciones adicionales:

Cifrado de campos sensibles (documentos de identidad, información de contacto).

Mecanismos de anonimización para cumplir con el derecho al olvido.

Logs de acceso a datos personales para auditorías de cumplimiento.

4. Conclusiones

La arquitectura de base de datos implementada para el sistema de agendamiento de citas ElectroHuila demuestra la aplicabilidad exitosa de principios de diseño relacional, patrones arquitectónicos modernos, y tecnologías ORM en sistemas empresariales del sector público. Las conclusiones principales de este trabajo son:

4.1. Logros Técnicos

El esquema relacional de 22 tablas normalizadas a 3NF proporciona una base sólida para la integridad de datos y mantenibilidad del sistema. La combinación de normalización rigurosa con desnormalizaciones estratégicas logra un balance efectivo entre consistencia de datos y rendimiento de consultas.

La implementación mediante Entity Framework Core 9.0 demostró capacidad de abstracción efectiva, permitiendo portabilidad entre cuatro gestores de bases de datos diferentes (Oracle, SQL Server, PostgreSQL, MySQL) con un único modelo de dominio. Esto reduce significativamente el riesgo tecnológico y facilita migraciones futuras.

Las características de auditoría temporal y soft deletes implementadas cumplen con requerimientos de trazabilidad del sector público, proporcionando capacidad de análisis retrospectivo sin comprometer el rendimiento de operaciones transaccionales.

4.2. Aplicabilidad del Modelo RBAC

El modelo de control de acceso basado en roles (RBAC) implementado mediante las tablas USERS, ROLES, PERMISSIONS, USER_ROLES y ROLE_PERMISSIONS proporciona flexibilidad y granularidad adecuadas para gestionar permisos en un sistema multiusuario complejo. Este diseño es extensible para incorporar permisos adicionales sin modificar el esquema base.

4.3. Optimización y Rendimiento

Las pruebas de rendimiento confirmaron que el uso estratégico de índices, consultas AsNoTracking, y patrones de carga eager vs. lazy produce tiempos de respuesta aceptables para el volumen de transacciones esperado. El overhead introducido por Entity Framework Core es compensado por la productividad ganada en desarrollo y mantenimiento.

4.4. Consideraciones para Oracle Database

Las particularidades de Oracle Database, especialmente la conversión de tipos booleanos a NUMBER(1) y las convenciones de nomenclatura en mayúsculas, requieren configuraciones específicas que fueron implementadas exitosamente mediante Fluent API. Estas adaptaciones no comprometieron la portabilidad del modelo de dominio.

4.5. Recomendaciones para Trabajo Futuro

Futuras iteraciones del sistema deberían considerar:

- Implementación de particionamiento horizontal en tablas de alto crecimiento (APPOINTMENTS, AUDIT_LOGS).
- Incorporación de cifrado a nivel de columna para datos sensibles, cumpliendo con la Ley 1581 de 2012.
- Evaluación de caché distribuido (Redis) para datos de catálogo de alta frecuencia de lectura.
- Implementación de estrategias de respaldo y recuperación ante desastres específicas para Oracle Database.

- Análisis de migración a microservicios con bases de datos descentralizadas por contexto acotado.

4.6. Aporte al Conocimiento

Este trabajo aporta un caso de estudio documentado de diseño de base de datos para sistemas de agendamiento PQR en el sector público colombiano, implementado con tecnologías actuales (.NET, Entity Framework Core, Oracle Database). El enfoque de Arquitectura Limpia aplicado a la capa de persistencia es replicable en dominios similares.

La experiencia adquirida en la adaptación de Entity Framework Core para Oracle Database, particularmente en el manejo de tipos booleanos y convenciones de nomenclatura, constituye conocimiento práctico valioso para la comunidad de desarrolladores .NET que trabajan con este gestor de bases de datos.