Creación de los Objetos de la Base de Datos del Proyecto "Censo Rural"

GA6-220501096-AA1-EV03

Isidro J Gallardo Navarro

Ficha:3070299

2025

Tecnología en Análisis y Desarrollo de Software.

ADSO

Introducción

El presente documento técnico desarrolla la creación de los objetos de la base de datos para el proyecto "Censo Rural", cumpliendo con los requisitos establecidos en la evidencia GA6-220501096-AA1-EV03. El objetivo principal es implementar una estructura de base de datos robusta que garantice la integridad, seguridad y funcionalidad del sistema de recolección de datos personales en zonas rurales apartadas. La base de datos está diseñada utilizando SQLite3, un motor de base de datos relacional ligero y eficiente, adecuado para aplicaciones móviles y entornos con conectividad limitada.

El diseño de la base de datos se fundamenta en los principios de integridad referencial, unicidad de datos y seguridad de la información personal, considerando que el sistema manejará datos sensibles protegidos por la legislación colombiana de protección de datos personales (Ley 1581 de 2012). Las sentencias DDL (Data Definition Language) presentadas en este documento definen las estructuras de almacenamiento, restricciones de integridad y mecanismos de seguridad necesarios para el funcionamiento óptimo del sistema.

1. Definición de Esquemas de Seguridad e Integridad

El diseño de la base de datos debe establecer esquemas de seguridad robustos para mantener la integridad de la información recolectada en el censo rural. Los esquemas de seguridad implementados se fundamentan en tres pilares esenciales que garantizan la confiabilidad y protección de los datos.

1.1 Integridad Referencial

La integridad referencial se implementa mediante la definición de claves primarias y claves foráneas en todas las tablas del sistema. Este mecanismo asegura que las relaciones entre entidades se mantengan consistentes, evitando registros huérfanos y garantizando la trazabilidad completa de la información.

Claves Primarias (PRIMARY KEY): Cada tabla del sistema cuenta con una clave primaria autoincrementable que identifica de manera única cada registro. Las claves primarias garantizan que no existan duplicados y proporcionan un identificador estable para las relaciones entre tablas. En SQLite3, el uso de INTEGER PRIMARY KEY AUTOINCREMENT asegura que los identificadores sean únicos y secuenciales, facilitando la sincronización entre dispositivos offline y el servidor central.

Claves Foráneas (FOREIGN KEY): Las relaciones entre tablas se establecen mediante claves foráneas que referencian las claves primarias de otras tablas. Por ejemplo, la tabla REGISTRO_CENSO incluye claves foráneas hacia USUARIO (identificando al encuestador que realizó la captura) y hacia HABITANTE_CENSADO (identificando al habitante censado). Estas restricciones previenen la inserción de datos inválidos y mantienen la coherencia del sistema, asegurando que no se pueda crear un registro de censo sin un encuestador válido o sin un habitante existente en la base de datos.

Es importante destacar que en SQLite3 las claves foráneas deben activarse explícitamente mediante el comando PRAGMA foreign_keys = ON; al inicio de cada sesión de base de datos, ya que esta funcionalidad está desactivada por defecto por razones de compatibilidad con versiones anteriores.

1.2 Unicidad de Datos

La restricción de unicidad se aplica a campos críticos que deben ser únicos en todo el sistema para evitar duplicados e inconsistencias en la información. La implementación de esta restricción es fundamental para garantizar la calidad de los datos del censo.

Número de Documento: El campo numeroDocumento en la tabla HABITANTE_CENSADO cuenta con una restricción UNIQUE, lo que impide que se registren dos habitantes con el mismo número de documento. Esta restricción es esencial para evitar la duplicación de personas censadas y mantener la precisión de las estadísticas del censo. La unicidad del número de documento asegura que cada habitante aparezca una sola vez en el sistema, independientemente de cuántos registros de censo tenga asociados a lo largo del tiempo.

Email de Usuario: En la tabla USUARIO, el campo email también cuenta con restricción UNIQUE, garantizando que cada cuenta de usuario tenga un correo electrónico único. Esto facilita la recuperación de contraseñas y previene la creación de cuentas duplicadas, reforzando la seguridad del sistema de autenticación.

1.3 Seguridad de Datos Personales

La protección de la información sensible del censo se implementa mediante múltiples mecanismos de seguridad a nivel de base de datos y aplicación.

Almacenamiento Seguro de Contraseñas: El campo contrasenaHash en la tabla USUARIO almacena únicamente el hash criptográfico de la contraseña, nunca el texto plano. Se

recomienda utilizar algoritmos de hash robustos como bcrypt o Argon2, que incluyen salt automático y son resistentes a ataques de fuerza bruta. Esta práctica garantiza que, incluso en caso de acceso no autorizado a la base de datos, las contraseñas originales permanezcan protegidas.

Encriptación de Archivos: El campo esEncriptado en la tabla ARCHIVO indica si un archivo adjunto (fotografía o documento) ha sido encriptado antes de su almacenamiento. Los archivos que contengan información sensible deben encriptarse utilizando algoritmos de encriptación simétrica como AES-256, protegiendo los datos en reposo contra accesos no autorizados.

Control de Acceso Basado en Roles: La tabla USUARIO incluye un campo rol con restricción CHECK que limita los valores posibles a 'Encuestador', 'Supervisor' y 'Administrador'. Este mecanismo de control de acceso basado en roles (RBAC) permite implementar permisos diferenciados según las responsabilidades de cada usuario, garantizando que solo personal autorizado pueda acceder a funciones críticas del sistema.

2. Creación de Sentencias DDL para Colecciones (Tablas)

Las siguientes sentencias SQL definen la estructura completa de la base de datos del proyecto "Censo Rural". Cada tabla se diseña considerando los requerimientos funcionales del sistema, las necesidades de seguridad y las mejores prácticas de diseño de bases de datos relacionales.

2.1 Tabla: CATEGORIA

La tabla CATEGORIA permite gestionar clasificaciones jerárquicas de datos por región, comunidad, vereda o grupo etario. Esta estructura facilita la organización territorial del censo y permite análisis estadísticos desagregados por niveles geográficos.

```
CREATE TABLE CATEGORIA (
    idCategoria INTEGER PRIMARY KEY AUTOINCREMENT,
    nombreCategoria VARCHAR(100) NOT NULL,
    nivelJerarquico INTEGER NOT NULL,
    descripcion TEXT,
    idCategoriaPadre INTEGER,
    fechaCreacion DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (idCategoriaPadre) REFERENCES CATEGORIA(idCategoria)
);
```

Índice para mejorar el rendimiento de consultas jerárquicas
CREATE INDEX idx_categoria_padre ON CATEGORIA(idCategoriaPadre);

Índice para búsquedas por nivel jerárquico
CREATE INDEX idx_categoria_nivel ON CATEGORIA(nivelJerarquico);

Descripción de campos:

- idCategoria: Identificador único de la categoría, clave primaria autoincrementable
- nombreCategoria: Nombre descriptivo de la categoría (ej. "Antioquia", "Zona Cafetera", "Adultos Mayores")
- nivelJerarquico: Nivel en la jerarquía (1 = Departamento, 2 = Municipio, 3 = Vereda, 4 = Grupo Etario)
- descripcion: Información adicional sobre la categoría
- idCategoriaPadre: Referencia a la categoría padre, permite estructuras jerárquicas (ej. una vereda pertenece a un municipio)
- fechaCreacion: Timestamp de creación del registro

Justificación funcional: La estructura jerárquica con autorreferencia permite modelar la división administrativa de Colombia desde departamentos hasta veredas, facilitando reportes agregados por niveles geográficos. Los índices creados optimizan las consultas de categorías por nivel y la navegación de jerarquías.

2.2 Tabla: USUARIO

La tabla USUARIO es la entidad base para la gestión de usuarios, roles y autenticación segura del sistema. Centraliza la información de todos los usuarios que interactúan con el sistema, independientemente de su rol.

```
CREATE TABLE USUARIO (
   idUsuario INTEGER PRIMARY KEY AUTOINCREMENT,
   email VARCHAR(100) UNIQUE NOT NULL,
   contrasenaHash TEXT NOT NULL,
   nombreCompleto VARCHAR(200) NOT NULL,
   numeroIdentificacion VARCHAR(30),
   rol VARCHAR(50) NOT NULL CHECK (rol IN ('Encuestador', 'Supervisor', 'Administrador')),
    estado VARCHAR(20) NOT NULL DEFAULT 'Activo' CHECK (estado IN ('Activo', 'Inactivo',
```

```
'Suspendido')),
    fechaCreacion DATETIME DEFAULT CURRENT_TIMESTAMP,
    ultimoAcceso DATETIME,
    intentosFallidos INTEGER DEFAULT 0
);

Índice único para email (garantiza unicidad y optimiza autenticación)
CREATE UNIQUE INDEX idx_usuario_email ON USUARIO(email);

Índice para consultas por rol
CREATE INDEX idx_usuario_rol ON USUARIO(rol);

Índice para consultas por estado
CREATE INDEX idx_usuario_estado ON USUARIO(estado);
```

Descripción de campos:

- idUsuario: Identificador único del usuario, clave primaria
- email: Correo electrónico único, utilizado como nombre de usuario para autenticación
- contrasenaHash: Hash criptográfico de la contraseña (bcrypt o Argon2)
- nombreCompleto: Nombre completo del usuario
- numeroIdentificacion: Número de documento de identidad del usuario
- rol: Rol del usuario en el sistema, con restricción CHECK para valores válidos
- estado: Estado de la cuenta (Activo, Inactivo, Suspendido)
- fechaCreacion: Fecha de creación de la cuenta
- ultimoAcceso: Timestamp del último inicio de sesión exitoso
- intentosFallidos: Contador de intentos de inicio de sesión fallidos (para bloqueo automático)

Justificación de seguridad: La restricción UNIQUE en el email previene cuentas duplicadas. La restricción CHECK en el campo rol garantiza que solo se asignen roles válidos, implementando control de acceso a nivel de base de datos. El campo intentosFallidos permite implementar políticas de bloqueo automático después de múltiples intentos fallidos, protegiendo contra ataques de fuerza bruta.

2.3 Tabla: ENCUESTADOR

La tabla ENCUESTADOR especializa la entidad USUARIO para almacenar información específica de los encuestadores que realizan trabajo de campo.

```
CREATE TABLE ENCUESTADOR (
    idEncuestador INTEGER PRIMARY KEY,
    idZonaAsignada INTEGER,
    metaDiaria INTEGER DEFAULT 10,
    certificacionVigente BOOLEAN NOT NULL DEFAULT 0,
    fechaCertificacion DATE,
    experienciaMeses INTEGER DEFAULT 0,
    FOREIGN KEY (idEncuestador) REFERENCES USUARIO(idUsuario) ON DELETE CASCADE,
    FOREIGN KEY (idZonaAsignada) REFERENCES ZONA_GEOGRAFICA(idZona)
);

Índice para consultas por zona asignada
CREATE INDEX idx_encuestador_zona ON ENCUESTADOR(idZonaAsignada);

Índice para filtrar encuestadores certificados
CREATE INDEX idx_encuestador_certificacion ON ENCUESTADOR(certificacionVigente);
```

Descripción de campos:

- idEncuestador: Identificador del encuestador, coincide con idUsuario (relación de herencia)
- idZonaAsignada: Zona geográfica asignada al encuestador
- metaDiaria: Número de registros esperados por día
- certificacionVigente: Indica si el encuestador tiene certificación válida
- fechaCertificacion: Fecha de la última certificación
- experienciaMeses: Meses de experiencia en trabajo de campo

Justificación funcional: La separación de atributos específicos del encuestador permite extender la entidad USUARIO sin contaminar la tabla base con campos que no aplican a otros roles. La cláusula ON DELETE CASCADE asegura que si se elimina un usuario, su registro de encuestador también se elimine, manteniendo la integridad referencial.

2.4 Tabla: ZONA_GEOGRAFICA

La tabla ZONA_GEOGRAFICA define las delimitaciones territoriales utilizadas para organizar la cobertura del censo.

```
CREATE TABLE ZONA_GEOGRAFICA (
   idZona INTEGER PRIMARY KEY AUTOINCREMENT,
   nombreZona VARCHAR(150) NOT NULL,
   codigoDANE VARCHAR(20),
   departamento VARCHAR(100) NOT NULL,
   municipio VARCHAR(100) NOT NULL,
   vereda VARCHAR(100),
   limiteGeografico TEXT,
   poblacionEstimada INTEGER,
   areaKm2 REAL,
   UNIQUE(codigoDANE)
);

Índice para búsquedas por departamento y municipio
CREATE INDEX idx_zona_depto_municipio ON ZONA_GEOGRAFICA(departamento, municipio);

Índice para búsquedas por código DANE
CREATE UNIQUE INDEX idx_zona_codigo_dane ON ZONA_GEOGRAFICA(codigoDANE);
```

Descripción de campos:

- idZona: Identificador único de la zona geográfica
- nombreZona: Nombre descriptivo de la zona
- codigoDANE: Código único del DANE para la ubicación
- departamento: Departamento al que pertenece la zona
- municipio: Municipio de la zona
- vereda: Vereda específica (opcional)
- limiteGeografico: Coordenadas del polígono delimitador (formato GeoJSON)
- poblacionEstimada: Población estimada en la zona
- areaKm2: Área de la zona en kilómetros cuadrados

Justificación funcional: La tabla permite organizar la cobertura territorial del censo, facilitando la asignación de zonas a encuestadores y el cálculo de métricas de cobertura. El código DANE garantiza compatibilidad con estadísticas oficiales de Colombia.

2.5 Tabla: HABITANTE_CENSADO

La tabla HABITANTE_CENSADO almacena los datos personales verificados de cada habitante censado. Es la tabla más sensible del sistema por el tipo de información que contiene.

```
CREATE TABLE HABITANTE_CENSADO (
   idHabitante INTEGER PRIMARY KEY AUTOINCREMENT,
    tipoDocumento VARCHAR(20) NOT NULL CHECK (tipoDocumento IN ('CC', 'TI', 'CE', 'PA', 'RC')),
   numeroDocumento VARCHAR(30) UNIQUE NOT NULL,
   primerNombre VARCHAR(100) NOT NULL,
   segundoNombre VARCHAR(100),
   primerApellido VARCHAR(100) NOT NULL,
   segundoApellido VARCHAR(100),
   fechaNacimiento DATE NOT NULL,
   sexo VARCHAR(10) NOT NULL CHECK (sexo IN ('M', 'F', 'Otro', 'Prefiere no decir')),
   emailContacto VARCHAR(100),
   telefonoContacto VARCHAR(20),
   domicilio TEXT,
   idZona INTEGER,
   estadoCivil VARCHAR(20),
   nivelEducativo VARCHAR(50),
   ocupacion VARCHAR(100),
   fechaRegistro DATETIME DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (idZona) REFERENCES ZONA_GEOGRAFICA(idZona)
);

Índice único para número de documento (crítico para prevenir duplicados)
CREATE UNIQUE INDEX idx_habitante_documento ON HABITANTE_CENSADO(numeroDocumento);
```

Índice compuesto para búsquedas por nombre
CREATE INDEX idx_habitante_nombre ON HABITANTE_CENSADO(primerApellido, segundoApellido, primerNombre);

Índice para consultas por zona geográfica
CREATE INDEX idx_habitante_zona ON HABITANTE_CENSADO(idZona);

Índice para consultas demográficas por edad
CREATE INDEX idx_habitante_fecha_nacimiento ON HABITANTE_CENSADO(fechaNacimiento);

Descripción de campos:

- idHabitante: Identificador único del habitante censado
- tipoDocumento: Tipo de documento de identidad (CC, TI, CE, PA, RC)
- numeroDocumento: Número de documento único en el sistema
- primerNombre, segundoNombre, primerApellido, segundoApellido: Nombres completos del habitante
- fechaNacimiento: Fecha de nacimiento para cálculo de edad
- sexo: Sexo o género declarado
- emailContacto: Correo electrónico de contacto
- telefonoContacto: Número telefónico de contacto
- domicilio: Dirección de residencia completa
- idZona: Zona geográfica de residencia
- estadoCivil: Estado civil del habitante
- nivelEducativo: Nivel educativo alcanzado
- ocupacion: Ocupación o actividad económica
- fechaRegistro: Fecha de registro inicial en el sistema

Justificación de seguridad: La restricción UNIQUE NOT NULL en numeroDocumento es crítica para prevenir duplicados e inconsistencias en el censo. Las restricciones CHECK en tipoDocumento y sexo garantizan la calidad de los datos almacenados. Los índices optimizan búsquedas frecuentes (por documento, nombre, zona) sin comprometer el rendimiento de inserción.

2.6 Tabla: FORMULARIO_CENSO

La tabla FORMULARIO_CENSO almacena las plantillas de formularios utilizados en el censo.

```
CREATE TABLE FORMULARIO_CENSO (
   idFormulario INTEGER PRIMARY KEY AUTOINCREMENT,
   nombreFormulario VARCHAR(150) NOT NULL,
   version VARCHAR(20) NOT NULL,
   descripcion TEXT,
   estructuraJSON TEXT NOT NULL,
   activo BOOLEAN NOT NULL DEFAULT 1,
   fechaCreacion DATETIME DEFAULT CURRENT_TIMESTAMP,
   fechaModificacion DATETIME,
   creadoPor INTEGER,
   FOREIGN KEY (creadoPor) REFERENCES USUARIO(idUsuario),
   UNIQUE(nombreFormulario, version)
);

Índice para consultas de formularios activos
CREATE INDEX idx_formulario_activo ON FORMULARIO_CENSO(activo);
```

Descripción de campos:

- idFormulario: Identificador único del formulario
- nombreFormulario: Nombre descriptivo del formulario
- version: Versión del formulario (control de cambios)
- descripcion: Descripción detallada del propósito del formulario
- estructuraJSON: Definición de campos y validaciones en formato JSON
- activo: Indica si el formulario está disponible para uso
- fechaCreacion: Fecha de creación del formulario
- fechaModificacion: Fecha de última modificación
- creadoPor: Usuario que creó el formulario

Justificación funcional: La combinación UNIQUE(nombreFormulario, version) permite versionar formularios, manteniendo múltiples versiones del mismo formulario para análisis históricos. El campo estructuraJSON permite flexibilidad en la definición de preguntas sin modificar el

esquema de base de datos.

2.7 Tabla: REGISTRO_CENSO

La tabla REGISTRO_CENSO almacena la instancia transaccional de cada encuesta capturada en campo, vinculando al encuestador con el habitante censado.

```
CREATE TABLE REGISTRO_CENSO (
   idRegistro INTEGER PRIMARY KEY AUTOINCREMENT,
   idEncuestador INTEGER NOT NULL,
   idHabitante INTEGER NOT NULL,
   idFormulario INTEGER NOT NULL,
   fechaHoraCaptura DATETIME DEFAULT CURRENT_TIMESTAMP,
   latitudGPS REAL,
   longitudGPS REAL,
   altitudGPS REAL,
   precisionGPS REAL,
            estadoSincronizacion VARCHAR(20) NOT NULL DEFAULT 'Offline' CHECK (estadoSincronizacion IN ('Offline', 'Sincronizado', 'Error')),
   fechaSincronizacion DATETIME,
   datosFormularioJSON TEXT,
   observaciones TEXT,
   FOREIGN KEY (idEncuestador) REFERENCES ENCUESTADOR(idEncuestador),
   FOREIGN KEY (idHabitante) REFERENCES HABITANTE_CENSADO(idHabitante),
   FOREIGN KEY (idFormulario) REFERENCES FORMULARIO_CENSO(idFormulario)
);

Índice para consultas por encuestador
CREATE INDEX idx_registro_encuestador ON REGISTRO_CENSO(idEncuestador);

Índice para consultas por habitante (historial)
CREATE INDEX idx_registro_habitante ON REGISTRO_CENSO(idHabitante);

Índice para sincronización pendiente
CREATE INDEX idx_registro_sincronizacion ON REGISTRO_CENSO(estadoSincronizacion,
```

fechaHoraCaptura);

Índice para consultas temporales
CREATE INDEX idx_registro_fecha ON REGISTRO_CENSO(fechaHoraCaptura);

Descripción de campos:

- idRegistro: Identificador único del registro de censo
- idEncuestador: Encuestador que realizó la captura (FK)
- idHabitante: Habitante censado (FK)
- idFormulario: Formulario utilizado (FK)
- fechaHoraCaptura: Timestamp de captura de datos
- latitudGPS, longitudGPS, altitudGPS: Coordenadas geográficas de captura
- precisionGPS: Precisión de la lectura GPS en metros
- estadoSincronizacion: Estado de sincronización con el servidor central
- fechaSincronizacion: Timestamp de sincronización exitosa
- datosFormularioJSON: Respuestas del formulario en formato JSON
- observaciones: Notas adicionales del encuestador

Justificación funcional: La captura de coordenadas GPS es esencial para el contexto rural, permitiendo georreferenciar cada registro y verificar la cobertura territorial. El campo estadoSincronizacion es crítico para el funcionamiento offline de la aplicación móvil, permitiendo identificar registros pendientes de sincronización. El campo datosFormularioJSON proporciona flexibilidad para almacenar respuestas de diferentes formularios sin modificar el esquema de la tabla.

2.8 Tabla: ARCHIVO

La tabla ARCHIVO gestiona el almacenamiento seguro de fotografías y documentos adjuntos a los registros de censo.

```
CREATE TABLE ARCHIVO (
  idArchivo INTEGER PRIMARY KEY AUTOINCREMENT,
  idRegistro INTEGER NOT NULL,
  nombreArchivo VARCHAR(255) NOT NULL,
  rutaAlmacenamiento TEXT NOT NULL,
  tipoArchivo VARCHAR(50) NOT NULL,
```

```
    tamanoBytes INTEGER,
    esEncriptado BOOLEAN NOT NULL DEFAULT 0,
    hashIntegridad VARCHAR(64),
    fechaSubida DATETIME DEFAULT CURRENT_TIMESTAMP,
    descripcion TEXT,
    FOREIGN KEY (idRegistro) REFERENCES REGISTRO_CENSO(idRegistro) ON DELETE CASCADE
);

Índice para consultas por registro
CREATE INDEX idx_archivo_registro ON ARCHIVO(idRegistro);

Índice para identificar archivos encriptados
CREATE INDEX idx_archivo_encriptado ON ARCHIVO(esEncriptado);

Índice para búsquedas por tipo de archivo
CREATE INDEX idx_archivo_tipo ON ARCHIVO(tipoArchivo);
```

Descripción de campos:

- idArchivo: Identificador único del archivo
- idRegistro: Registro de censo al que pertenece el archivo (FK)
- nombreArchivo: Nombre original del archivo
- rutaAlmacenamiento: Ruta relativa o URL del archivo almacenado
- tipoArchivo: Tipo MIME del archivo (image/jpeg, application/pdf, etc.)
- tamanoBytes: Tamaño del archivo en bytes
- esEncriptado: Indica si el archivo está encriptado en reposo
- hashIntegridad: Hash SHA-256 del archivo para verificar integridad
- fechaSubida: Timestamp de carga del archivo
- descripcion: Descripción del contenido del archivo

Justificación de seguridad: El campo esEncriptado permite marcar archivos que contienen información sensible y que deben ser encriptados antes de almacenarse. El campo hashIntegridad permite verificar que los archivos no han sido modificados o corrompidos durante la transmisión o almacenamiento. La cláusula ON DELETE CASCADE asegura que si se elimina un registro de censo, sus archivos asociados también se eliminen, manteniendo la consistencia de

la base de datos.

3. Scripts de Inicialización y Configuración

Para garantizar el funcionamiento correcto de la base de datos SQLite3, se deben ejecutar los siguientes scripts de configuración al iniciar cada conexión.

3.1 Script de Activación de Claves Foráneas

```
Activar restricciones de claves foráneas (desactivadas por defecto en SQLite)
PRAGMA foreign_keys = ON;

Verificar que las claves foráneas están activadas
PRAGMA foreign_keys;

Configurar modo WAL para mejor concurrencia
PRAGMA journal_mode = WAL;

Optimizar cache para operaciones frecuentes
PRAGMA cache_size = -8000; -- 8MB de cache
```

3.2 Script de Inserción de Datos Iniciales

```
Insertar roles administrativos predeterminados
INSERT INTO USUARIO (email, contrasenaHash, nombreCompleto, rol, estado)
VALUES
('admin@censoRural.gov.co', '$2b$12$hash_example', 'Administrador del Sistema', 'Administrador', 'Activo'),
('supervisor@censoRural.gov.co', '$2b$12$hash_example', 'Supervisor General', 'Supervisor', 'Activo');

Insertar categorías geográficas base
INSERT INTO CATEGORIA (nombreCategoria, nivelJerarquico, descripcion)
VALUES
('Departamento', 1, 'Nivel administrativo departamental'),
('Municipio', 2, 'Nivel administrativo municipal'),
```

('Vereda', 3, 'Nivel administrativo veredal'),
('Grupo Etario', 4, 'Clasificación por rangos de edad');

Insertar formulario de censo base
INSERT INTO FORMULARIO_CENSO (nombreFormulario, version, descripcion, estructuraJSON, activo, creadoPor)
VALUES
('Censo Rural Básico', '1.0', 'Formulario estándar de censo rural', '{"fields": []}', 1, 1);

4. Cumplimiento de Criterios de Evaluación

La siguiente tabla resume el cumplimiento de los criterios de evaluación establecidos en la evidencia GA6-220501096-AA1-EV03.

Tabla 1. Matriz de Cumplimiento de Criterios de Evaluación

Criterio de Evaluación	Cumplimiento en el Modelo SQL DDL	Evidencia
Crea la base de datos integrando los objetos de acuerdo con la funcionalidad del software	Se han definido 8 tablas completas que cubren: gestión de usuarios y roles (USUARIO, ENCUESTADOR), datos de habitantes censados (HABITANTE_CENSADO), captura de datos en campo (REGISTRO_CENSO), clasificación territorial (CATEGORIA, ZONA_GEOGRAFICA), formularios dinámicos (FORMULARIO_CENSO) y almacenamiento de evidencias (ARCHIVO)	Secciones 2.1 a 2.8
Crea sentencias para la creación de colecciones	Se proporcionan 8 sentencias CREATE TABLE completas con definición exhaustiva de campos, tipos de datos, restricciones de integridad y claves primarias/foráneas	Secciones 2.1 a 2.8
Define esquemas de seguridad en la base de datos para mantener la integridad de la información	La integridad se garantiza mediante: (1) PRIMARY KEY en todas las tablas, (2) FOREIGN KEY con integridad referencial, (3) Restricción UNIQUE en numeroDocumento y email, (4) Restricciones CHECK en campos de rol, estado y tipo de documento, (5) Campos de seguridad: contrasenaHash, esEncriptado, hashIntegridad	Sección 1, Subsecciones 1.1, 1.2, 1.3
Implementa mecanismos de sincronización para operación offline	El campo estadoSincronizacion en REGISTRO_CENSO con restricción CHECK permite identificar registros pendientes de sincronización, esencial para trabajo en campo sin conectividad	Sección 2.7
Protege datos personales sensibles	Implementación de: (1) Almacenamiento de hash de contraseñas, (2) Campo esEncriptado para archivos sensibles, (3) Hash de integridad de archivos, (4) Control de acceso basado en roles	Sección 1.3

5. Consideraciones de Implementación y Mejores Prácticas

5.1 Optimización del Rendimiento

La base de datos incluye índices estratégicamente ubicados para optimizar las consultas más

frecuentes sin impactar significativamente el rendimiento de inserción. Los índices compuestos (ej. departamento, municipio) optimizan consultas de rango comunes en reportes geográficos.

5.2 Mantenimiento y Auditoría

Se recomienda implementar triggers de auditoría para registrar cambios en tablas sensibles como HABITANTE_CENSADO y USUARIO, manteniendo un historial de modificaciones para fines de trazabilidad y cumplimiento normativo.

5.3 Escalabilidad

El diseño permite escalar mediante particionamiento de tablas grandes (REGISTRO_CENSO, ARCHIVO) por rangos temporales o geográficos si el volumen de datos lo requiere en el futuro.

Conclusiones

El presente documento ha desarrollado exhaustivamente la creación de los objetos de la base de datos del proyecto "Censo Rural", cumpliendo con todos los requisitos establecidos en la evidencia GA6-220501096-AA1-EV03. Se han definido ocho tablas principales con sus respectivas sentencias DDL, implementando esquemas robustos de seguridad e integridad de la información.

Los aspectos más destacados de la implementación incluyen:

Integridad de Datos: Todas las tablas cuentan con claves primarias autoincrementables y claves foráneas que garantizan la integridad referencial del sistema. Las restricciones de unicidad en campos críticos como numeroDocumento y email previenen duplicados y mantienen la consistencia de la información.

Seguridad de la Información: Se han implementado múltiples capas de seguridad, incluyendo almacenamiento de hashes de contraseñas, indicadores de encriptación para archivos sensibles, control de acceso basado en roles mediante restricciones CHECK, y hashes de integridad para verificar la autenticidad de los archivos almacenados.

Funcionalidad Offline: El diseño contempla el requisito crítico de operación sin conectividad mediante el campo estadoSincronizacion en la tabla REGISTRO_CENSO, permitiendo identificar y sincronizar registros capturados en zonas rurales sin acceso a internet.

Georreferenciación: La captura de coordenadas GPS en cada registro de censo permite verificar la cobertura territorial, generar mapas de calor de densidad poblacional y validar que los

encuestadores efectivamente visitaron las zonas asignadas.

Flexibilidad y Escalabilidad: El uso de campos JSON para almacenar respuestas de formularios y estructuras de formularios permite adaptar el sistema a diferentes tipos de censos sin modificar el esquema de base de datos, facilitando la evolución del proyecto.

El modelo de datos presentado constituye una base sólida para el desarrollo del sistema "Censo Rural", balanceando adecuadamente los requerimientos de funcionalidad, seguridad, rendimiento y escalabilidad necesarios para un proyecto de censo en zonas rurales apartadas de Colombia.

Referencias

American Psychological Association. (2020). *Publication manual of the American Psychological Association* (7th ed.). https://doi.org/10.1037/0000165-000

Date, C. J. (2019). *Database design and relational theory: Normal forms and all that jazz* (2nd ed.). Apress. https://doi.org/10.1007/978-1-4842-5540-7

Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of database systems* (7th ed.). Pearson Education.

Kreibich, J. A. (2010). *Using SQLite*. O'Reilly Media.

República de Colombia. (2012). *Ley 1581 de 2012: Régimen general de protección de datos personales*. Diario Oficial No. 48.587. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database system concepts* (7th ed.). McGraw-Hill Education.

SQLite Consortium. (2024). *SQLite documentation*. https://www.sqlite.org/docs.html