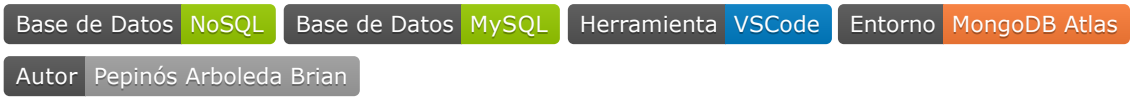
INFORME U1 – Modelado de Sistemas Operacionales (OLTP)



Autor

Nombre completo: TNTE Pepinós Arboleda Brian
Asignatura: Modelado avanzado de Base de Datos
Fecha: 04 noviembre del 2025
Repositorio de github: https://github.com/BRIANPEPINOS/Modelado-Avanzada-de-BDD/tree/main/Proyecto

1. Introducción

El presente informe corresponde al Avance 1 del proyecto TechStore, el cual tiene como propósito integrar diferentes tecnologías de bases de datos dentro de una solución de Business Intelligence (BI). En esta primera etapa se abordó el modelado de los sistemas operacionales (OLTP), los cuales constituyen la base de datos de origen que alimentará los análisis posteriores.

La arquitectura propuesta combina un entorno SQL(relacional) y un entorno NoSQL (basado en documentos). De esta manera, se busca aprovechar las fortalezas de cada tecnología según el tipo de información. Como mencionan Coronel y Morris (2022), *"no existe una sola base de datos que resuelva todas las necesidades de información en una organización"*, por lo que los modelos híbridos se han vuelto comunes en entornos empresariales modernos.

2. Estructura del proyecto

```
PROYECTO
├── SISTEMA_NOSQL/
│   ├── operaciones.mongodb
│   └── validation.mongodb
│
├── SISTEMA_SQL/
│   ├── Diagrama_entidad_relacion.png
│   └── schema_ventas.sql
│
└── INFORME_U1.md
```

3. Validación de los productos

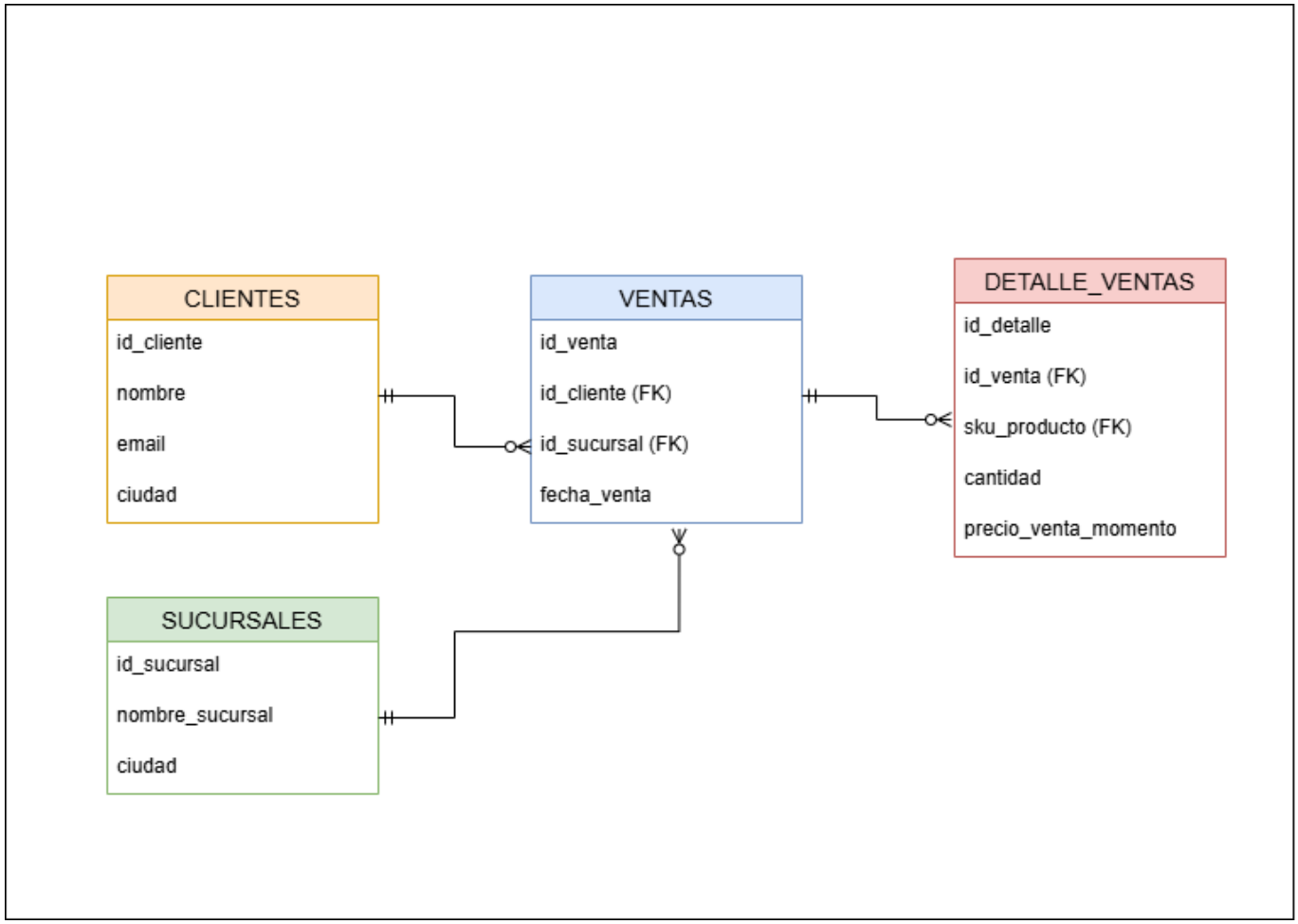
```
db.createCollection('productos', {
    validator: {
        $jsonSchema: {
            bsonType: 'object',
            required: [
                'sku',
                'nombre',
                'precio',
                'stock',
                'tipo_producto',
                'fecha_creacion',
                'especificaciones'
            ],
            properties: {
                sku: {
                    bsonType: 'string',
                    description: 'dos letras mayúsculas seguidas de cuatro números
(ej. AB1234)',
                    pattern: '^[A-Z]{2}[0-9]{4}$'
                },
                nombre: {
                    bsonType: 'string',
                    minLength: 3
                },
                precio: {
                    bsonType: 'double',
                    minimum: 0
                },
                stock: {
                    bsonType: 'int',
                    minimum: 0
                },
                tipo_producto: {
                    bsonType: 'string',
                    enum: [
                        'Smartphone',
                        'Laptop',
                        'Monitor',
                        'Accesorio',
                        'Tablet',
                        'Audio',
                        'Otro'
                    ]
                },
                fecha_creacion: {
                    bsonType: 'date'
                },
                especificaciones: {
                    bsonType: 'object',
                    additionalProperties: true,
                    minProperties: 1
                }, ultima_revision: {
                    bsonType: ['date', 'null'],
```

```
                },
            }
        }
    },

});
```

3. Diagrama entidad relacion

Figura 1. Diagrama Entidad–Relación (DER) del caso “TechStore”



4. Creacion de las tablas en MYSQL

```
CREATE TABLE clientes (
  id_cliente      CHAR(36) NOT NULL DEFAULT (UUID()) PRIMARY KEY,
  nombre          VARCHAR(120) NOT NULL,
  email           VARCHAR(180),
  ciudad          VARCHAR(80),
  CONSTRAINT email_formato_chk
    CHECK (email IS NULL OR email REGEXP '^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]
{2,}$')
) ENGINE=InnoDB;

CREATE TABLE sucursales (
```

```
  id_sucursal     CHAR(36) NOT NULL DEFAULT (UUID()) PRIMARY KEY,
  nombre_sucursal VARCHAR(120) NOT NULL,
  ciudad          VARCHAR(80)  NOT NULL
) ENGINE=InnoDB;


CREATE TABLE ventas (
  id_venta        CHAR(36) NOT NULL DEFAULT (UUID()) PRIMARY KEY,
  id_cliente      CHAR(36) NOT NULL,
  id_sucursal     CHAR(36) NOT NULL,
  fecha_venta     DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT ventas_cliente_fk  FOREIGN KEY (id_cliente)  REFERENCES
clientes(id_cliente),
  CONSTRAINT ventas_sucursal_fk FOREIGN KEY (id_sucursal) REFERENCES
sucursales(id_sucursal)
) ENGINE=InnoDB;


CREATE TABLE detalle_ventas (
  id_detalle            BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  id_venta              CHAR(36) NOT NULL,
  sku_producto          VARCHAR(6) NOT NULL,
  cantidad              INT NOT NULL,
  precio_venta_momento  DECIMAL(12,2) NOT NULL,
  CONSTRAINT detalle_ventas_venta_fk
    FOREIGN KEY (id_venta) REFERENCES ventas(id_venta) ON DELETE CASCADE,
  CONSTRAINT cantidad_chk CHECK (cantidad > 0),
  CONSTRAINT precio_chk   CHECK (precio_venta_momento >= 0),
  CONSTRAINT sku_formato_chk CHECK (sku_producto REGEXP '^[A-Z]{2}[0-9]{4}$')
) ENGINE=InnoDB;
```

3. Justificación del Modelo Dual (SQL + NoSQL)

La decisión de utilizar dos tipos de bases de datos distintos en TechStore se basa en la naturaleza de los datos y en el propósito de cada sistema. Las ventas, clientes y sucursales presentan información estructurada y estable, por lo que se implementó una base de datos relacional (MySQL) que garantice la integridad, consistencia y normalización de las transacciones.

Por otro lado, el catálogo de productos se caracteriza por tener atributos variables según la categoría (por ejemplo, un smartphone tiene “pantalla” y “RAM”, mientras que un monitor tiene “resolución” y “frecuencia”). Por esta razón, se utilizó MongoDB, una base de datos NoSQL orientada a documentos, que permite definir subdocumentos anidados y estructuras flexibles sin necesidad de alterar el esquema.

De acuerdo con Elmasri y Navathe (2020), *“el modelo de datos debe adaptarse a la naturaleza de la información y a las operaciones que se realizarán sobre ella”*. Bajo ese enfoque, el modelo relacional se centra en la estabilidad y control de las transacciones, mientras que el modelo documental busca adaptarse a la diversidad de productos y reducir la rigidez del diseño.

---

3. Informe de Calidad de Datos

Durante la implementación del modelo, se identificaron varios posibles problemas.

El primer problema identificado corresponde a la inconsistencia del SKU entre ambos sistemas. En la base de datos relacional, el campo sku_producto dentro de la tabla detalle_ventas podría referenciar un producto que no exista en la colección productos de MongoDB. Esta situación generaría errores al consolidar la información de ventas con el inventario. Para mitigar este riesgo, se estableció un formato estándar del SKU mediante una expresión regular `^[A-Z]{2}[0-9]{4}$) y se realizará una validación previa antes de registrar una venta.

El segundo problema está relacionado con la duplicidad de productos en MongoDB. Si no se aplica una restricción de unicidad, podrían existir dos documentos con el mismo código SKU, generando inconsistencias en los precios o en el stock disponible. Para evitarlo, se creó un índice único sobre el campo sku en la colección de productos.

El tercer problema se refiere a la calidad del campo email en la tabla de clientes. En muchos casos, los correos pueden quedar nulos o contener errores de formato. Por tal motivo, se añadió una restricción CHECK en MySQL con una expresión regular que valida la estructura del correo electrónico, evitando registros inválidos.

Por último, se identificó un cuarto problema relacionado con la actualización inconsistente del stock en tiempo real. En escenarios donde múltiples ventas se registran simultáneamente, es posible que el stock de un producto no se sincronice correctamente entre MongoDB y MySQL .

---

4. Conclusiones

La implementación de un modelo dual entre SQL y NoSQL en el proyecto *TechStore* permitió comprender cómo diferentes tipos de datos pueden coexistir en un mismo ecosistema empresarial. El modelo relacional asegura el control de las transacciones, mientras que el modelo NoSQL aporta flexibilidad para almacenar productos con atributos variables.

Además, el análisis de calidad de datos permitió anticipar problemas comunes como la duplicidad de registros, las referencias inconsistentes. Gracias a la aplicación de restricciones, validadores e índices únicos, se fortaleció la confiabilidad de los datos desde el origen.

---

5. Referencias

- Elmasri, R., & Navathe, S. (2020). *Fundamentals of Database Systems* (7th ed.). Pearson Educatio
- Coronel, C., & Morris, S. (2022). *Database Systems: Design, Implementation, and Management* (14th ed.). Cengage Learning.