Diseño de Arquitectura de Base de Datos para Mouse Kerramientas

1. Introducción

Este documento presenta el diseño detallado de la arquitectura de base de datos para Mouse Kerramientas, una plataforma dedicada al alquiler de herramientas tanto para usuarios individuales como empresas MYPES. El sistema está diseñado para gestionar el inventario de herramientas, clientes, alquileres, pagos y mantenimientos de forma eficiente, incorporando además funcionalidades como reseñas, calificaciones, chat y notificaciones.

La arquitectura propuesta implementa un modelo híbrido que maximiza el uso de SQL para los datos estructurados y transaccionales, minimizando la redundancia, y reserva NoSQL solamente para aquellos datos genuinamente no estructurados o que requieren alta flexibilidad.

2. Justificación de la Arquitectura Híbrida Optimizada

La decisión de implementar una arquitectura híbrida con predominio SQL se basa en:

1. **Minimización de redundancia**: Mantener la mayor parte de los datos en el sistema SQL relacional permite aprovechar la normalización para reducir la duplicación de datos.
2. **Integridad referencial**: Las relaciones entre entidades principales (herramientas, clientes, alquileres, especificaciones, mensajes) se benefician de la integridad referencial garantizada por las claves foráneas de SQL.
3. **Transaccionalidad**: Las operaciones críticas (alquileres, pagos, gestión de inventario, mensajería) requieren garantías ACID que son nativas en sistemas SQL.
4. **Uso selectivo de NoSQL**: Limitado a datos genuinamente no estructurados como reseñas, notificaciones y eventos promocionales que tienen requisitos de flexibilidad y escalabilidad específicos.
5. **Optimización de consultas**: Las operaciones más frecuentes (búsqueda de herramientas, gestión de alquileres, mensajería) se benefician del potente motor de consultas SQL y sus capacidades de indexación.

3. Componentes Estructurales

3.1 Base de Datos SQL (Datos Estructurados y Transaccionales)

La base de datos SQL gestionará:

- Catálogo de herramientas
- Especificaciones técnicas de herramientas
- Información de clientes

- Transacciones de alquiler
- Items de alquiler
- Registros de pagos
- Mensajería (conversaciones y mensajes)
- Categorías de productos

3.2 Base de Datos NoSQL (Datos Flexibles)

La base de datos NoSQL (MongoDB) se limita a gestionar:

- Reseñas y calificaciones
- Notificaciones del sistema
- Eventos y promociones

3.3 Capa de Integración

Un middleware de integración ligero que:

- Gestiona las referencias entre sistemas SQL y NoSQL
- Mantiene la consistencia entre sistemas
- Proporciona una API unificada para la aplicación

4. Modelo de Datos SQL

Tablas Principales

Categorías

```sql
CREATE TABLE categorias (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    imagen_url VARCHAR(255),
    activo BOOLEAN DEFAULT TRUE,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Herramientas

```sql
CREATE TABLE herramientas (
    id INT PRIMARY KEY AUTO_INCREMENT,
    codigo VARCHAR(20) UNIQUE NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    categoria_id INT NOT NULL,
    precio_diario DECIMAL(10,2) NOT NULL,
    estado VARCHAR(50) CHECK (estado IN ('Nuevo', 'Usado-Bueno', 'Usado-Regular', 'En Mantenimi
    imagen_url VARCHAR(255),
    stock_total INT NOT NULL DEFAULT 1,
    stock_disponible INT NOT NULL DEFAULT 1,
    deposito_garantia DECIMAL(10,2) DEFAULT 0,
    requiere_deposito BOOLEAN DEFAULT FALSE,
    calificacion_promedio DECIMAL(3,2) DEFAULT 0,
    cantidad_resenas INT DEFAULT 0,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (categoria_id) REFERENCES categorias(id)
);
```

Especificaciones Técnicas

```sql
CREATE TABLE especificaciones_tecnicas (
    id INT PRIMARY KEY AUTO_INCREMENT,
    herramienta_id INT NOT NULL,
    marca VARCHAR(100),
    modelo VARCHAR(100),
    potencia VARCHAR(50),
    peso DECIMAL(8,2),
    dimensiones VARCHAR(100),
    caracteristicas_adicionales TEXT,
    fecha_fabricacion DATE,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (herramienta_id) REFERENCES herramientas(id) ON DELETE CASCADE
);
```

Clientes

sql

```sql
CREATE TABLE clientes (
    id INT PRIMARY KEY AUTO_INCREMENT,
    tipo VARCHAR(20) CHECK (tipo IN ('Individual', 'Empresa')),
    nombre VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100),
    razon_social VARCHAR(150),
    documento_identidad VARCHAR(20),
    ruc VARCHAR(20),
    direccion TEXT,
    telefono VARCHAR(20),
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    activo BOOLEAN DEFAULT TRUE,
    ubicacion_lat DECIMAL(10,8),
    ubicacion_lng DECIMAL(11,8),
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Preferencias de Cliente

sql

```sql
CREATE TABLE preferencias_cliente (
    id INT PRIMARY KEY AUTO_INCREMENT,
    cliente_id INT NOT NULL,
    categoria_id INT,
    metodo_pago_preferido VARCHAR(50),
    notificacion_email BOOLEAN DEFAULT TRUE,
    notificacion_sms BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (cliente_id) REFERENCES clientes(id) ON DELETE CASCADE,
    FOREIGN KEY (categoria_id) REFERENCES categorias(id)
);
```

Alquileres

sql

```sql
CREATE TABLE alquileres (
    id INT PRIMARY KEY AUTO_INCREMENT,
    codigo_alquiler VARCHAR(20) UNIQUE NOT NULL,
    cliente_id INT NOT NULL,
    fecha_solicitud TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_inicio DATE NOT NULL,
    fecha_fin_prevista DATE NOT NULL,
    fecha_devolucion DATE,
    monto_total DECIMAL(10,2) NOT NULL,
    monto_deposito DECIMAL(10,2) DEFAULT 0,
    estado VARCHAR(50) CHECK (estado IN ('Solicitado', 'Aceptado', 'Rechazado', 'En Curso', 'Fi
    observaciones TEXT,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    fecha_actualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
);
```

Items de Alquiler

sql

```sql
CREATE TABLE alquiler_items (
    id INT PRIMARY KEY AUTO_INCREMENT,
    alquiler_id INT NOT NULL,
    herramienta_id INT NOT NULL,
    cantidad INT NOT NULL DEFAULT 1,
    precio_unitario DECIMAL(10,2) NOT NULL,
    subtotal DECIMAL(10,2) NOT NULL,
    estado_devolucion VARCHAR(50) DEFAULT 'Pendiente' CHECK (estado_devolucion IN ('Pendiente',
    observaciones TEXT,
    FOREIGN KEY (alquiler_id) REFERENCES alquileres(id) ON DELETE CASCADE,
    FOREIGN KEY (herramienta_id) REFERENCES herramientas(id)
);
```

Pagos

```sql
CREATE TABLE pagos (
    id INT PRIMARY KEY AUTO_INCREMENT,
    alquiler_id INT NOT NULL,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    monto DECIMAL(10,2) NOT NULL,
    metodo_pago VARCHAR(50) NOT NULL,
    referencia_pago VARCHAR(100),
    es_adelanto BOOLEAN DEFAULT FALSE,
    estado VARCHAR(50) DEFAULT 'Completado',
    FOREIGN KEY (alquiler_id) REFERENCES alquileres(id) ON DELETE CASCADE
);
```

Conversaciones

```sql
CREATE TABLE conversaciones (
    id INT PRIMARY KEY AUTO_INCREMENT,
    cliente_id INT NOT NULL,
    alquiler_id INT,
    asunto VARCHAR(200),
    estado VARCHAR(50) DEFAULT 'Activo' CHECK (estado IN ('Activo', 'Cerrado', 'Archivado')),
    ultimo_mensaje TIMESTAMP,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (cliente_id) REFERENCES clientes(id),
    FOREIGN KEY (alquiler_id) REFERENCES alquileres(id) ON DELETE SET NULL
);
```

Mensajes

```sql
CREATE TABLE mensajes (
    id INT PRIMARY KEY AUTO_INCREMENT,
    conversacion_id INT NOT NULL,
    emisor_tipo VARCHAR(20) NOT NULL CHECK (emisor_tipo IN ('Cliente', 'Sistema')),
    emisor_id INT NOT NULL,
    contenido TEXT NOT NULL,
    leido BOOLEAN DEFAULT FALSE,
    fecha_envio TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    metadatos JSON,
    FOREIGN KEY (conversacion_id) REFERENCES conversaciones(id) ON DELETE CASCADE
);
```

5. Estructura NoSQL (MongoDB)

Colecciones Principales

Reseñas y Calificaciones

json

```
{
  "_id": "ObjectId",
  "herramienta_sql_id": "INT", // Referencia a ID en SQL
  "cliente_sql_id": "INT", // Referencia a ID en SQL
  "alquiler_sql_id": "INT", // Referencia a ID en SQL
  "calificacion": "Number", // 1-5
  "comentario": "String",
  "fecha": "Date",
  "respuesta": {
    "texto": "String",
    "fecha": "Date"
  },
  "likes": "Number",
  "reportado": "Boolean",
  "fecha_creacion": "Date",
  "fecha_actualizacion": "Date"
}
```

Notificaciones

json

```
{
  "_id": "ObjectId",
  "usuario_sql_id": "INT", // Referencia a ID en SQL
  "tipo": "String", // alquiler, pago, resena, etc.
  "titulo": "String",
  "mensaje": "String",
  "referencia_id": "String", // ID de la entidad relacionada
  "referencia_tipo": "String", // alquiler, herramienta, etc.
  "leido": "Boolean",
  "importante": "Boolean",
  "fecha_envio": "Date",
  "fecha_lectura": "Date",
  "metadatos": { // Campo flexible para datos adicionales
    "clave1": "valor1",
    "clave2": "valor2"
  }
}
```

Eventos y Promociones

```json
{
  "_id": "ObjectId",
  "titulo": "String",
  "descripcion": "String",
  "tipo": "String", // descuento, oferta especial, evento
  "descuento_porcentaje": "Number",
  "herramientas_aplicables_sql_ids": ["INT"], // IDs SQL
  "categorias_aplicables_sql_ids": ["INT"], // IDs SQL
  "fecha_inicio": "Date",
  "fecha_fin": "Date",
  "codigo_promocion": "String",
  "limite_usos": "Number",
  "usos_actuales": "Number",
  "condiciones": ["String"],
  "imagen_url": "String",
  "activo": "Boolean",
  "prioridad": "Number", // Para ordenar promociones
  "criterios_segmentacion": { // Permite segmentación flexible
    "ubicacion": {
      "ciudades": ["String"],
      "distritos": ["String"]
    },
    "tipo_cliente": ["String"],
    "historial_minimo": "Number"
  },
  "fecha_creacion": "Date",
  "fecha_actualizacion": "Date"
}
```

6. Estrategia de Integración

6.1 Sincronización de Datos

- **Referencias unidireccionales**: Todas las referencias entre sistemas van desde NoSQL hacia SQL, no al revés, lo que simplifica la sincronización.
- **Actualizaciones controladas**: Los cambios en calificaciones se propagan de forma controlada al campo `calificacion_promedio` de la tabla SQL `herramientas`.
- **Sin duplicación de datos**: No se duplican datos estructurados en NoSQL, solo se mantienen referencias a las entidades SQL.

6.2 Middleware de Integración

javascript

```javascript
// Ejemplo de función de middleware para actualizar calificación promedio
async function actualizarCalificacionHerramienta(herramientaId) {
  // Obtener todas las calificaciones de la herramienta desde MongoDB
  const resenas = await mongoDb.collection('resenas')
    .find({ herramienta_sql_id: herramientaId })
    .toArray();

  // Calcular promedio
  const totalCalificaciones = resenas.length;
  let sumaCalificaciones = 0;

  resenas.forEach(resena => {
    sumaCalificaciones += resena.calificacion;
  });

  const promedio = totalCalificaciones > 0 ? sumaCalificaciones / totalCalificaciones : 0;

  // Actualizar en SQL
  await sqlDB.query(
    'UPDATE herramientas SET calificacion_promedio = ?, cantidad_resenas = ? WHERE id = ?',
    [promedio.toFixed(2), totalCalificaciones, herramientaId]
  );

  return {
    promedio: promedio.toFixed(2),
    totalCalificaciones
  };
}
```

7. Índices y Optimización

7.1 Índices SQL

sql

```sql
-- Índices para herramientas
CREATE INDEX idx_herramientas_categoria ON herramientas(categoria_id);
CREATE INDEX idx_herramientas_estado_stock ON herramientas(estado, stock_disponible);
CREATE INDEX idx_herramientas_precio ON herramientas(precio_diario);

-- Índices para especificaciones
CREATE UNIQUE INDEX idx_especificaciones_herramienta ON especificaciones_tecnicas(herramienta_i

-- Índices para clientes
CREATE INDEX idx_clientes_ubicacion ON clientes(ubicacion_lat, ubicacion_lng);
CREATE INDEX idx_clientes_documento ON clientes(documento_identidad);

-- Índices para alquileres
CREATE INDEX idx_alquileres_cliente ON alquileres(cliente_id);
CREATE INDEX idx_alquileres_estado_fechas ON alquileres(estado, fecha_inicio, fecha_fin_previst
CREATE INDEX idx_alquileres_codigo ON alquileres(codigo_alquiler);

-- Índices para conversaciones y mensajes
CREATE INDEX idx_conversaciones_cliente ON conversaciones(cliente_id);
CREATE INDEX idx_conversaciones_alquiler ON conversaciones(alquiler_id);
CREATE INDEX idx_mensajes_conversacion_fecha ON mensajes(conversacion_id, fecha_envio);
CREATE INDEX idx_mensajes_leidos ON mensajes(conversacion_id, leido);

-- Índices para búsqueda full-text
CREATE FULLTEXT INDEX idx_herramientas_busqueda ON herramientas(nombre, descripcion);
CREATE FULLTEXT INDEX idx_mensajes_contenido ON mensajes(contenido);
```

7.2 Índices NoSQL

javascript

```javascript
// Índices para reseñas
db.resenas.createIndex({ herramienta_sql_id: 1, fecha: -1 });
db.resenas.createIndex({ cliente_sql_id: 1 });
db.resenas.createIndex({ alquiler_sql_id: 1 });

// Índices para notificaciones
db.notificaciones.createIndex({ usuario_sql_id: 1, leido: 1, fecha_envio: -1 });
db.notificaciones.createIndex({ referencia_id: 1, referencia_tipo: 1 });

// Índices para eventos y promociones
db.eventos_promociones.createIndex({ fecha_inicio: 1, fecha_fin: 1, activo: 1 });
db.eventos_promociones.createIndex({ herramientas_aplicables_sql_ids: 1 });
db.eventos_promociones.createIndex({ categorias_aplicables_sql_ids: 1 });

// Índices de texto completo
db.resenas.createIndex({ comentario: "text" });
db.eventos_promociones.createIndex({ titulo: "text", descripcion: "text" });
```

8. Consultas Optimizadas

8.1 Consultas SQL

sql

```sql
-- Búsqueda de herramientas disponibles por categoría con especificaciones
SELECT h.*, e.*
FROM herramientas h
LEFT JOIN especificaciones_tecnicas e ON h.id = e.herramienta_id
WHERE h.categoria_id = ? AND h.estado = 'Disponible' AND h.stock_disponible > 0
ORDER BY h.precio_diario ASC;

-- Historial de alquileres de un cliente con detalles
SELECT a.*, i.herramienta_id, i.cantidad, i.precio_unitario, i.subtotal, h.nombre as herramient
FROM alquileres a
JOIN alquiler_items i ON a.id = i.alquiler_id
JOIN herramientas h ON i.herramienta_id = h.id
WHERE a.cliente_id = ?
ORDER BY a.fecha_inicio DESC;

-- Mensajes de una conversación
SELECT m.*
FROM mensajes m
WHERE m.conversacion_id = ?
ORDER BY m.fecha_envio ASC;

-- Actualizar estado de mensajes a leídos
UPDATE mensajes
SET leido = TRUE
WHERE conversacion_id = ? AND emisor_tipo = 'Cliente' AND leido = FALSE;
```

8.2 Consultas NoSQL

```javascript
// Obtener reseñas de una herramienta
db.resenas.find({
  herramienta_sql_id: herramientaId
}).sort({ fecha: -1 }).limit(10);

// Obtener notificaciones no leídas de un usuario
db.notificaciones.find({
  usuario_sql_id: usuarioId,
  leido: false
}).sort({ fecha_envio: -1 });

// Promociones activas para una categoría
db.eventos_promociones.find({
  categorias_aplicables_sql_ids: categoriaId,
  fecha_inicio: { $lte: new Date() },
  fecha_fin: { $gte: new Date() },
  activo: true
}).sort({ prioridad: -1 });
```

9. Escalabilidad

9.1 Escalabilidad SQL

- **Particionamiento vertical**: Separación de tablas de alta escritura (mensajes) en servidores dedicados
- **Particionamiento horizontal**: Configuración para futura fragmentación de tablas históricas por rango de fechas
- **Índices específicos**: Optimizados para cada patrón de acceso importante
- **Réplicas de lectura**: Para equilibrar cargas en operaciones de consulta intensivas

9.2 Escalabilidad NoSQL

- **Sharding por colección**: Configuración preparada para fragmentar colecciones grandes (notificaciones)
- **TTL (Time To Live)**: Configuración para expiración automática de notificaciones antiguas
- **Compresión**: Habilitada para optimizar almacenamiento

10. Seguridad

- **Encriptación en tránsito**: SSL/TLS para todas las comunicaciones
- **Encriptación en reposo**: Para datos sensibles de clientes
- **Control de acceso**: Basado en roles para ambas bases de datos
- **Auditoría**: Registro de cambios críticos para cumplimiento normativo

- **Sanitización de entradas**: Validación rigurosa para prevenir inyección SQL/NoSQL

11. Estrategia de Respaldo

- **Respaldos diferenciales diarios**: Para ambas bases de datos
- **Respaldos completos semanales**: Con retención de 6 meses
- **Pruebas de restauración**: Programadas mensualmente
- **Replicación asíncrona**: Para recuperación ante desastres

12. Conclusiones

La arquitectura híbrida optimizada para Mouse Kerramientas ofrece:

1. **Integridad de datos máxima**: Al mantener la mayoría de los datos en SQL con integridad referencial.
2. **Redundancia mínima**: Siguiendo principios de normalización en la parte SQL.
3. **Flexibilidad selectiva**: Usando NoSQL solo donde realmente aporta valor (reseñas, notificaciones, promociones).
4. **Alto rendimiento**: Con consultas optimizadas para cada patrón de acceso.
5. **Escalabilidad planificada**: Con estrategias claras para crecimiento futuro.

Esta arquitectura permite implementar todas las funcionalidades requeridas de forma eficiente, manteniendo la complejidad controlada y facilitando el mantenimiento a largo plazo.