Actividad Práctica Integrada: Sistema Bancario

Temas: Indexación, Transacciones y Recuperación

📖 Enunciado y Contexto de la Actividad

Escenario del Sistema

Imagina que trabajas como **Administrador de Bases de Datos (DBA)** en un banco que necesita implementar un **nuevo sistema de gestión de cuentas y transacciones**. El banco ha decidido migrar de su sistema legacy a una solución moderna basada en MySQL.

Situación actual:

- El banco maneja aproximadamente 10,000 clientes activos
- Se procesan alrededor de 5,000 transacciones diarias
- El sistema actual tiene problemas de rendimiento en consultas frecuentes
- Se han reportado inconsistencias en transferencias cuando hay fallos del sistema
- No existe un sistema robusto de recuperación ante desastres

Requisitos del nuevo sistema:

1. **Rendimiento:** Las consultas de saldo y búsqueda de cuentas deben ser instantáneas
2. **Confiabilidad:** Todas las transacciones deben garantizar integridad de datos (ACID)
3. **Recuperación:** El sistema debe poder recuperarse automáticamente ante fallos
4. **Escalabilidad:** Debe soportar el crecimiento futuro del banco

Tu misión: Diseñar e implementar la base de datos del nuevo sistema aplicando los conceptos de **indexación**, **transacciones** y **recuperación** de manera integrada.

📋 Objetivo de la Actividad

Implementar y analizar un sistema bancario que integre tres conceptos fundamentales:

1. **Indexación y Asociación** (Tema 6): Optimizar consultas mediante índices
2. **Procesamiento Transaccional** (Tema 7): Garantizar propiedades ACID
3. **Sistemas de Recuperación** (Tema 8): Manejar fallos y recuperación

---

🛠️ Herramientas Necesarias

Software Requerido:

- **MySQL 8.0+**
- **Cliente SQL**: MySQL Workbench o línea de comandos (`mysql`)
- **Editor de texto**: Para crear scripts SQL

Instalación Rápida:

```
# Verificar instalación de MySQL
mysql --version

# Conectar a MySQL
mysql -u root -p

# O desde MySQL Workbench: crear nueva conexión
```

🎯 Metodología: Implementación de un Sistema Nuevo

Antes de comenzar con la implementación técnica, es fundamental seguir una metodología estructurada. A continuación se presenta el **paso a paso conceptual** para implementar un sistema de base de datos desde cero:

Fase 1: Análisis y Planificación

1.1 Identificación de Requisitos Funcionales

Objetivo: Entender qué debe hacer el sistema

Actividades:

- Reuniones con usuarios finales (cajeros, gerentes, clientes)
- Análisis de procesos de negocio actuales
- Identificación de operaciones críticas:
  - Consulta de saldos
  - Transferencias entre cuentas
  - Depósitos y retiros
  - Consulta de historial de transacciones
  - Búsqueda de clientes por DNI

Resultado: Documento de requisitos funcionales

1.2 Identificación de Requisitos No Funcionales

Objetivo: Definir características de calidad del sistema

Actividades:

- **Rendimiento:** Tiempo de respuesta < 100ms para consultas simples
- **Disponibilidad:** Sistema debe estar operativo 24/7
- **Confiabilidad:** Tolerancia a fallos con recuperación automática
- **Seguridad:** Control de acceso y auditoría de operaciones
- **Escalabilidad:** Capacidad de crecer sin cambios arquitectónicos

Resultado: Especificación de requisitos no funcionales

Fase 2: Diseño Conceptual

2.1 Identificación de Entidades

Objetivo: Identificar los objetos principales del dominio

Proceso:

1. **Análisis de sustantivos** en los requisitos funcionales
2. **Identificación de entidades principales:**
    - **Cliente:** Persona que tiene cuentas en el banco
    - **Cuenta:** Producto bancario asociado a un cliente
    - **Transacción:** Operación financiera que modifica saldos
    - **Log de Transacciones:** Registro de auditoría para recuperación
3. **Identificación de entidades relacionadas:**
    - Tipos de cuenta (Ahorro, Corriente)
    - Tipos de transacción (Depósito, Retiro, Transferencia)
    - Estados de transacción (Pendiente, Confirmada, Abortada)

Resultado: Diagrama de entidades principales

2.2 Identificación de Atributos

Objetivo: Definir las propiedades de cada entidad

Proceso por entidad:

Cliente:

- Identificadores: `cliente_id` (PK), `dni` (único)
- Datos personales: `nombre`, `apellido`, `email`, `telefono`
- Metadatos: `fecha_registro`

Cuenta:

- Identificadores: `cuenta_id` (PK), `numero_cuenta` (único)
- Relaciones: `cliente_id` (FK)
- Datos financieros: `tipo_cuenta`, `saldo`
- Metadatos: `fecha_apertura`, `activa`

Transacción:

- Identificadores: `transaccion_id` (PK)
- Relaciones: `cuenta_origen_id` (FK), `cuenta_destino_id` (FK)
- Datos financieros: `tipo_transaccion`, `monto`
- Metadatos: `fecha_transaccion`, `estado`, `descripcion`

Log de Transacciones:

- Identificadores: `log_id` (PK)
- Relaciones: `transaccion_id` (FK)
- Datos de auditoría: `operacion`, `tabla_afectada`, `registro_id`
- Datos de recuperación: `valor_anterior`, `valor_nuevo`, `timestamp_log`

Resultado: Especificación completa de atributos

2.3 Identificación de Relaciones

Objetivo: Definir cómo se relacionan las entidades

Relaciones identificadas:

- **Cliente** → **Cuenta:** 1 a N (un cliente puede tener múltiples cuentas)
- **Cuenta** → **Transacción (origen):** 1 a N (una cuenta puede tener múltiples transacciones como origen)
- **Cuenta** → **Transacción (destino):** 1 a N (una cuenta puede tener múltiples transacciones como destino)
- **Transacción** → **Log:** 1 a N (una transacción genera múltiples registros de log)

Resultado: Diagrama de relaciones (ER)

Fase 3: Diseño Lógico

3.1 Normalización

Objetivo: Eliminar redundancias y garantizar integridad

Proceso:

- **Primera Forma Normal (1NF):** Eliminar grupos repetitivos
- **Segunda Forma Normal (2NF):** Eliminar dependencias parciales
- **Tercera Forma Normal (3NF):** Eliminar dependencias transitivas

Resultado: Esquema normalizado

3.2 Definición de Restricciones

Objetivo: Garantizar integridad de datos

Restricciones identificadas:

- **Claves primarias:** Identificadores únicos
- **Claves foráneas:** Integridad referencial
- **Unicidad:** `dni`, `numero_cuenta`
- **Dominios:** Valores permitidos (tipos de cuenta, estados)
- **Chequeos:** `saldo >= 0`, `monto > 0`

Resultado: Especificación de restricciones

Fase 4: Diseño Físico

4.1 Parametrización del SGBD

Objetivo: Configurar MySQL para el caso de uso específico

Parámetros críticos a configurar:

Motor de Almacenamiento:

```
-- Usar InnoDB para soporte transaccional
ENGINE=InnoDB
```

Configuración de Transacciones:

- `autocommit`: Control manual de transacciones
- `transaction_isolation`: Nivel de aislamiento (READ COMMITTED recomendado)
- `innodb_lock_wait_timeout`: Tiempo de espera para bloqueos

Configuración de Logging:

- `log_bin`: Habilitar binlog para recuperación
- `binlog_format`: Formato del binlog (ROW recomendado)
- `innodb_log_file_size`: Tamaño del log de InnoDB
- `innodb_flush_log_at_trx_commit`: Frecuencia de escritura del log

Configuración de Rendimiento:

- `innodb_buffer_pool_size`: Memoria para caché de datos
- `max_connections`: Número máximo de conexiones simultáneas
- `query_cache_size`: Caché de consultas (MySQL 5.7, removido en 8.0)

Resultado: Archivo de configuración `my.cnf` optimizado

4.2 Estrategia de Indexación

Objetivo: Optimizar consultas frecuentes

Proceso:

1. **Análisis de consultas frecuentes:**
    - Búsqueda por número de cuenta
    - Búsqueda de cliente por DNI
    - Consulta de transacciones por cuenta y fecha
    - Consulta de saldo por cliente
2. **Identificación de columnas candidatas:**
    - Columnas en cláusulas WHERE
    - Columnas en JOIN
    - Columnas en ORDER BY

- Columnas en GROUP BY

3. **Diseño de índices:**
   - **Índices primarios:** Ya definidos (claves primarias)
   - **Índices únicos:** Para búsquedas exactas (`numero_cuenta`, `dni`)
   - **Índices compuestos:** Para consultas con múltiples condiciones
   - **Índices covering:** Para consultas que solo necesitan datos del índice

Resultado: Plan de indexación

4.3 Estrategia de Transacciones

Objetivo: Garantizar propiedades ACID

Diseño:

- **Atomicidad:** Usar `START TRANSACTION` / `COMMIT` / `ROLLBACK`
- **Consistencia:** Validaciones antes de confirmar (fondos suficientes, cuentas activas)
- **Aislamiento:** Bloqueos `FOR UPDATE` en lecturas críticas
- **Durabilidad:** Configuración de `innodb_flush_log_at_trx_commit = 1`

Resultado: Procedimientos almacenados transaccionales

4.4 Estrategia de Recuperación

Objetivo: Garantizar recuperación ante fallos

Componentes:

- **Log de transacciones personalizado:** Tabla `log_transacciones` para auditoría
- **Binlog de MySQL:** Para recuperación a nivel de sistema
- **Backups regulares:** Estrategia de backup y restore
- **Procedimientos de recuperación:** UNDO y REDO manuales

Resultado: Plan de recuperación y procedimientos

Fase 5: Implementación

5.1 Creación del Esquema

Actividades:

- Crear base de datos
- Crear tablas con restricciones
- Crear índices según el plan
- Insertar datos de prueba

5.2 Implementación de Lógica de Negocio

Actividades:

- Crear procedimientos almacenados para operaciones críticas
- Implementar validaciones de negocio
- Configurar logging de transacciones

5.3 Pruebas y Validación

Actividades:

- Pruebas de rendimiento (con y sin índices)
- Pruebas de transacciones (casos exitosos y fallidos)
- Pruebas de recuperación (simulación de fallos)

Fase 6: Optimización y Mantenimiento

6.1 Monitoreo

- Análisis de planes de ejecución
- Identificación de consultas lentas
- Monitoreo de uso de índices

6.2 Ajustes

- Crear índices adicionales si es necesario
- Eliminar índices no utilizados
- Ajustar parámetros del SGBD según rendimiento observado
