Sesión	*26*
Tema	Implementación de restricciones de integridad.
Propósito	Fortalecer las capacidades de los participantes respecto a la implementación de restricciones de integridad en el contexto de desarrollo de bases de datos, mediante la resolución de casos propuestos en una sesión práctica.
Fecha	C.04.12.2025
Hora	17:00

Implementación de Restricciones de Integridad en Bases de Datos

📚 Background Teórico

¿Qué son las Restricciones de Integridad?

Las **restricciones de integridad** son reglas que se aplican a los datos en una base de datos para garantizar su **validez, precisión y consistencia**. Estas reglas preservan la calidad y fiabilidad de la información, asegurando que los datos cumplan con las condiciones establecidas por el modelo de negocio.

Tipos de Restricciones de Integridad

a) Integridad de Entidad

- **Clave Primaria (PRIMARY KEY)**: Identifica de forma única cada registro
- **Unicidad (UNIQUE)**: Valores no duplicados en una columna
- **AUTO_INCREMENT**: Generación automática de valores secuenciales

b) Integridad Referencial

- **Clave Foránea (FOREIGN KEY)**: Relación entre tablas
- **Reglas de propagación**: CASCADE, SET NULL, RESTRICT, NO ACTION

3. Integridad de Dominio

- **Tipos de datos**: INT, VARCHAR, DATE, etc.
- **NOT NULL**: Valores obligatorios
- **CHECK**: Condiciones personalizadas
- **DEFAULT**: Valores predeterminados
- **ENUM/SET**: Valores predefinidos

4. Integridad de Usuario

- **Triggers**: Acciones automáticas ante eventos
- **Procedimientos almacenados**: Validaciones complejas

¿Cuándo y Cómo se Implementan?

Fase de definición:

- Durante el diseño de tablas (CREATE TABLE)
- Al modificar estructuras (ALTER TABLE)
- En cualquier momento posterior a la creación

Formas de implementación:

- Inline (en la definición de columna)
- Como restricciones a nivel de tabla
- Mediante ALTER TABLE posterior

Ejemplos Básicos

```sql
-- 1. INTEGRIDAD DE ENTIDAD
CREATE TABLE empleados (
    id_empleado INT PRIMARY KEY AUTO_INCREMENT,
    dni VARCHAR(10) UNIQUE NOT NULL,
    nombre VARCHAR(50) NOT NULL
);

-- 2. INTEGRIDAD REFERENCIAL
CREATE TABLE pedidos (
    id_pedido INT PRIMARY KEY,
    id_cliente INT,
    fecha DATE NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

-- 3. INTEGRIDAD DE DOMINIO
CREATE TABLE productos (
    id_producto INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    precio DECIMAL(10,2) CHECK (precio >= 0),
    categoria ENUM('Electrónica', 'Ropa', 'Alimentos') DEFAULT 'Alimentos',
    stock INT NOT NULL DEFAULT 0
);

-- 4. INTEGRIDAD CON CHECK COMPLEJO
CREATE TABLE reservas (
    id_reserva INT PRIMARY KEY,
    fecha_inicio DATE,
    fecha_fin DATE,
    CHECK (fecha_fin > fecha_inicio),
    CHECK (fecha_inicio >= CURDATE())
);
```

🏢 Caso Ejemplo: Sistema de Biblioteca Universitaria

Enunciado del Problema

Una universidad necesita gestionar su biblioteca, controlando libros, préstamos a estudiantes, multas por retrasos y disponibilidad de ejemplares. Se requiere garantizar la integridad de los datos en todo momento.

Requerimientos Funcionales

a) Gestionar información de estudiantes
b) Registrar libros y ejemplares disponibles
c) Controlar préstamos y devoluciones
d) Calcular multas automáticamente
e) Evitar préstamos a estudiantes con multas pendientes

Actores

- **Estudiante**: Solicita y devuelve libros
- **Bibliotecario**: Gestiona todo el proceso
- **Sistema**: Aplica reglas automáticamente

Casos de Uso

a) Registrar nuevo estudiante
b) Agregar libro al catálogo
c) Realizar préstamo
d) Registrar devolución
e) Calcular multa por retraso
f) Bloquear estudiante con multas

🎨 Diseño de Base de Datos

Diseño Conceptual (Modelo Entidad-Relación)

```
ESTUDIANTE (id_estudiante, nombre, email, estado)
LIBRO (id_libro, titulo, autor, isbn)
EJEMPLAR (id_ejemplar, id_libro, estado)
PRÉSTAMO (id_prestamo, id_estudiante, id_ejemplar, fecha_prestamo, fecha_devoluc
ion_estimada, fecha_devolucion_real)
MULTA (id_multa, id_estudiante, monto, estado, fecha_generacion)
```

Diseño Lógico (Modelo Relacional)

```sql
-- Tabla ESTUDIANTE
CREATE TABLE estudiantes (
    id_estudiante INT PRIMARY KEY,
    codigo_estudiante VARCHAR(10) UNIQUE NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    telefono VARCHAR(15),
    estado ENUM('ACTIVO', 'INACTIVO', 'BLOQUEADO') DEFAULT 'ACTIVO',
    fecha_registro DATE DEFAULT (CURDATE()),
    CHECK (email LIKE '%@%.%')
);

-- Tabla LIBRO
CREATE TABLE libros (
    id_libro INT PRIMARY KEY AUTO_INCREMENT,
    isbn VARCHAR(13) UNIQUE NOT NULL,
    titulo VARCHAR(200) NOT NULL,
    autor VARCHAR(150) NOT NULL,
    editorial VARCHAR(100),
    anio_publicacion YEAR,
    categoria VARCHAR(50),
    CONSTRAINT chk_isbn_length CHECK (LENGTH(isbn) IN (10, 13))
);
```

```sql
-- Tabla EJEMPLAR
CREATE TABLE ejemplares (
    id_ejemplar INT PRIMARY KEY AUTO_INCREMENT,
    id_libro INT NOT NULL,
    codigo_barras VARCHAR(20) UNIQUE NOT NULL,
    estado ENUM('DISPONIBLE', 'PRESTADO', 'MANTENIMIENTO', 'PERDIDO') DEFAULT 'D
ISPONIBLE',
    ubicacion VARCHAR(50),
    fecha_adquisicion DATE DEFAULT (CURDATE()),
    FOREIGN KEY (id_libro) REFERENCES libros(id_libro)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
-- Tabla PRÉSTAMO
CREATE TABLE prestamos (
    id_prestamo INT PRIMARY KEY AUTO_INCREMENT,
    id_estudiante INT NOT NULL,
    id_ejemplar INT NOT NULL,
    fecha_prestamo DATETIME DEFAULT CURRENT_TIMESTAMP,
    fecha_devolucion_estimada DATE NOT NULL,
    fecha_devolucion_real DATE,
    estado ENUM('ACTIVO', 'DEVUELTO', 'RETRASADO') DEFAULT 'ACTIVO',
    -- Restricciones CHECK
    CHECK (fecha_devolucion_estimada > DATE(fecha_prestamo)),
    CHECK (fecha_devolucion_real IS NULL OR fecha_devolucion_real >= DATE(fecha_
prestamo)),
    -- Claves foráneas
    FOREIGN KEY (id_estudiante) REFERENCES estudiantes(id_estudiante)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (id_ejemplar) REFERENCES ejemplares(id_ejemplar)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    -- Índice para búsquedas frecuentes
    INDEX idx_fecha_prestamo (fecha_prestamo),
    INDEX idx_estado (estado)
);
```

```sql
-- Tabla MULTA
CREATE TABLE multas (
    id_multa INT PRIMARY KEY AUTO_INCREMENT,
    id_estudiante INT NOT NULL,
    id_prestamo INT NOT NULL,
    monto DECIMAL(8,2) NOT NULL CHECK (monto >= 0),
    fecha_generacion DATE DEFAULT (CURDATE()),
    fecha_pago DATE,
    estado ENUM('PENDIENTE', 'PAGADA', 'PERDONADA') DEFAULT 'PENDIENTE',
    motivo VARCHAR(200),
    -- Integridad referencial
    FOREIGN KEY (id_estudiante) REFERENCES estudiantes(id_estudiante)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (id_prestamo) REFERENCES prestamos(id_prestamo)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    -- Restricción lógica
    CHECK (fecha_pago IS NULL OR fecha_pago >= fecha_generacion)
);
```

Diseño Físico (MariaDB/MySQL)

```sql
-- 1. CREACIÓN DE LA BASE DE DATOS

CREATE DATABASE IF NOT EXISTS biblioteca_universitaria
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

USE biblioteca_universitaria;

-- 2. TABLAS CON RESTRICCIONES (como se definió anteriormente)
-- [Aquí irían todas las sentencias CREATE TABLE anteriores]
```

```
-- 3. RESTRICCIONES ADICIONALES CON ALTER TABLE
-- Evitar que un estudiante con multas pendientes realice préstamos
DELIMITER //
CREATE TRIGGER verificar_multas_antes_prestamo
BEFORE INSERT ON prestamos
FOR EACH ROW
BEGIN
    DECLARE multas_pendientes INT;

    SELECT COUNT(*) INTO multas_pendientes
    FROM multas
    WHERE id_estudiante = NEW.id_estudiante
    AND estado = 'PENDIENTE';

    IF multas_pendientes > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El estudiante tiene multas pendientes';
    END IF;
END//
DELIMITER ;

-- 4. ACTUALIZAR ESTADO DE EJEMPLAR AL PRESTAR
DELIMITER //
CREATE TRIGGER actualizar_estado_ejemplar_prestamo
AFTER INSERT ON prestamos
FOR EACH ROW
BEGIN
    IF NEW.estado = 'ACTIVO' THEN
        UPDATE ejemplares
        SET estado = 'PRESTADO'
        WHERE id_ejemplar = NEW.id_ejemplar;
    END IF;
END//
DELIMITER ;
```

```
-- 5. GENERAR MULTA AUTOMÁTICA POR RETRASO
DELIMITER //
CREATE TRIGGER generar_multa_automatica
AFTER UPDATE ON prestamos
FOR EACH ROW
BEGIN
    DECLARE dias_retraso INT;

    IF NEW.fecha_devolucion_real IS NOT NULL
       AND OLD.fecha_devolucion_real IS NULL
       AND NEW.fecha_devolucion_real > NEW.fecha_devolucion_estimada THEN

        SET dias_retraso = DATEDIFF(NEW.fecha_devolucion_real, NEW.fecha_devoluc
ion_estimada);

        IF dias_retraso > 0 THEN
            INSERT INTO multas (id_estudiante, id_prestamo, monto, motivo)
            VALUES (
                NEW.id_estudiante,
                NEW.id_prestamo,
                dias_retraso * 5.00, -- $5 por día de retraso
                CONCAT('Retraso de ', dias_retraso, ' días')
            );
        END IF;
    END IF;
END//
DELIMITER ;

-- 6. VISTA PARA REPORTES
CREATE VIEW vista_prestamos_activos AS
SELECT
    p.id_prestamo,
    e.nombre AS estudiante,
    l.titulo AS libro,
    ej.codigo_barras,
    p.fecha_prestamo,
    p.fecha_devolucion_estimada,
    DATEDIFF(CURDATE(), p.fecha_devolucion_estimada) AS dias_retraso
```

```
FROM prestamos p
JOIN estudiantes e ON p.id_estudiante = e.id_estudiante
JOIN ejemplares ej ON p.id_ejemplar = ej.id_ejemplar
JOIN libros l ON ej.id_libro = l.id_libro
WHERE p.estado = 'ACTIVO'
AND p.fecha_devolucion_real IS NULL;

-- 7. PROCEDIMIENTO ALMACENADO PARA PRÉSTAMO
DELIMITER //
CREATE PROCEDURE realizar_prestamo(
    IN p_id_estudiante INT,
    IN p_id_ejemplar INT,
    IN p_dias_prestamo INT
)
BEGIN
    DECLARE v_estado_estudiante VARCHAR(20);
    DECLARE v_estado_ejemplar VARCHAR(20);
    DECLARE v_prestamos_activos INT;

    -- Verificar estado del estudiante
    SELECT estado INTO v_estado_estudiante
    FROM estudiantes
    WHERE id_estudiante = p_id_estudiante;

    IF v_estado_estudiante != 'ACTIVO' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Estudiante no está activo';
    END IF;

    -- Verificar estado del ejemplar
    SELECT estado INTO v_estado_ejemplar
    FROM ejemplares
    WHERE id_ejemplar = p_id_ejemplar;

    IF v_estado_ejemplar != 'DISPONIBLE' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Ejemplar no disponible';
    END IF;
```

```
    -- Verificar límite de préstamos activos (máximo 3)
    SELECT COUNT(*) INTO v_prestamos_activos
    FROM prestamos
    WHERE id_estudiante = p_id_estudiante
    AND estado = 'ACTIVO';

    IF v_prestamos_activos >= 3 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Límite de préstamos activos alcanzado';
    END IF;

    -- Insertar préstamo
    INSERT INTO prestamos (
        id_estudiante,
        id_ejemplar,
        fecha_devolucion_estimada
    ) VALUES (
        p_id_estudiante,
        p_id_ejemplar,
        DATE_ADD(CURDATE(), INTERVAL p_dias_prestamo DAY)
    );

    SELECT 'Préstamo realizado exitosamente' AS mensaje;
END//
DELIMITER ;
```

Inserción de Datos de Ejemplo

```
-- Insertar estudiantes
INSERT INTO estudiantes (id_estudiante, codigo_estudiante, nombre, email) VALUES
(1, '2023001', 'Ana García', 'ana.garcia@universidad.edu'),
(2, '2023002', 'Carlos López', 'carlos.lopez@universidad.edu');
```

```sql
-- Insertar libros
INSERT INTO libros (isbn, titulo, autor, categoria) VALUES
('9788499890944', 'Cien años de soledad', 'Gabriel García Márquez', 'Literatura'
),
('9788437604947', 'Don Quijote de la Mancha', 'Miguel de Cervantes', 'Literatura
');

-- Insertar ejemplares
INSERT INTO ejemplares (id_libro, codigo_barras, estado) VALUES
(1, 'LIB001-001', 'DISPONIBLE'),
(1, 'LIB001-002', 'DISPONIBLE'),
(2, 'LIB002-001', 'DISPONIBLE');

-- Realizar un préstamo usando el procedimiento almacenado
CALL realizar_prestamo(1, 1, 15);
```

🔍 Consultas de Validación

```sql
-- 1. Verificar integridad referencial
SELECT
    TABLE_NAME,
    COLUMN_NAME,
    CONSTRAINT_NAME,
    REFERENCED_TABLE_NAME,
    REFERENCED_COLUMN_NAME
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE TABLE_SCHEMA = 'biblioteca_universitaria'
AND REFERENCED_TABLE_NAME IS NOT NULL;

-- 2. Verificar restricciones CHECK
SELECT
    TABLE_NAME,
    CONSTRAINT_NAME,
    CHECK_CLAUSE
FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS
WHERE CONSTRAINT_SCHEMA = 'biblioteca_universitaria';
```

```sql
-- 3. Prueba de restricciones
-- Esto debería fallar por el CHECK de precio
INSERT INTO productos (id_producto, nombre, precio) VALUES (1, 'Libro', -10);

-- Esto debería fallar por la restricción UNIQUE
INSERT INTO estudiantes (id_estudiante, codigo_estudiante, nombre, email)
VALUES (3, '2023001', 'Juan Pérez', 'juan@email.com');
```

📝 Buenas Prácticas de Implementación

a) **Planificar desde el diseño**: Definir restricciones en la fase de modelado
b) **Usar nombres descriptivos**: `fk_prestamos_estudiantes`, `chk_precio_positivo`
c) **Documentar restricciones**: Comentar el propósito de cada restricción
d) **Validar rendimiento**: Índices en claves foráneas para mejor performance
e) **Manejar errores**: Capturar violaciones con mensajes claros
f) **Backup antes de cambios**: Especialmente al modificar restricciones existentes
g) **Pruebas exhaustivas**: Validar todos los casos de violación posibles

🎯 Conclusión

Las restricciones de integridad son **fundamentales** para garantizar la calidad de los datos en cualquier sistema de base de datos. Su implementación debe ser:

- **Completa**: Cubrir todos los aspectos de integridad
- **Consistente**: Sin contradicciones entre restricciones
- **Eficiente**: Sin impactar negativamente el rendimiento
- **Mantenible**: Con nombres claros y documentación

El caso de la biblioteca universitaria demuestra cómo las restricciones trabajan en conjunto para mantener la coherencia de datos en un sistema real, previniendo inconsistencias y automatizando procesos de validación.

Bibliografía Recomendada

• "Fundamentos de Bases de Datos" de Abraham Silberschatz, Henry F. Korth y S. Sudarshan

• "Sistemas de Bases de Datos: un enfoque práctico" de Thomas M. Connolly y Carolyn Begg

• "Desarrollo de Bases de Datos: casos prácticos desde el análisis a la implementación" de Dolores Cuadra, Elena Castro, Ana M. Iglesias

• "Tecnología y Diseño de Bases de Datos" de Marcos, C. Calero y B. Vela

• https://docs.oracle.com/en/database/