NORMALIZACIÓN DE BASE DE DATOS Y SCRIPTS CRUD

ANÁLISIS DE NORMALIZACIÓN

PRIMERA FORMA NORMAL (1FN)

Requisitos:

- Cada columna debe contener valores atómicos (indivisibles)
- No debe haber grupos repetitivos
- Cada columna debe tener un nombre único
- Cada registro debe ser único

CUMPLIMIENTO: Todas las tablas cumplen con 1FN:

- Todos los campos tienen valores atómicos
- No hay grupos repetitivos
- Cada campo tiene nombre único
- Cada tabla tiene Primary Key

Ejemplo - Tabla PACIENTES:

- **MAL:** `contactos: "Juan:555-1234, Maria:555-5678"` (no es atómico)
- **BIEN:** `contactoEmergencia: "Juan", telefonoEmergencia: "555-1234"` (valores separados y atómicos)

SEGUNDA FORMA NORMAL (2FN)

Requisitos:

- Debe cumplir 1FN
- Todos los atributos no clave deben depender completamente de la clave primaria
- No debe haber dependencias parciales

CUMPLIMIENTO: Todas las tablas cumplen con 2FN: **Ejemplo - Tabla CITAS:**

- PK: `idCita`
- Todos los campos (`fechaCita`, `motivoConsulta`, `estadoCita`, etc.) dependen completamente de `idCita`
- No hay campos que dependan solo de parte de la clave primaria

Ejemplo - Tabla PAGOS:

- PK: `idPago`
- `monto` depende de `idPago` (no de `idCita` ni `idPaciente` por separado)
- `metodoPago` depende de `idPago`
- Las FK (`idCita`, `idPaciente`) solo establecen relaciones, no causan dependencias parciales

TERCERA FORMA NORMAL (3FN)

Requisitos:

- Debe cumplir 2FN
- No debe haber dependencias transitivas
- Los atributos no clave no deben depender de otros atributos no clave

CUMPLIMIENTO: Las tablas cumplen con 3FN mediante la separación de entidades: **Ejemplo - MEDICOS y ESPECIALIDADES:**

- **MAL (violaría 3FN):**

MEDICOS:
- idMedico (PK)
- nombreMedico
- nombreEspecialidad ← depende de especialidadId, no de idMedico
- descripcionEspecialidad ← depende de especialidadId, no de idMedico

- **BIEN (cumple 3FN):**

MEDICOS:
- idMedico (PK)
- nombreMedico
- especialidadId (FK) ← solo referencia

ESPECIALIDADES:
- idEspecialidad (PK)

- nombreEspecialidad ← depende de idEspecialidad
- descripcion ← depende de idEspecialidad

Ejemplo - PAGOS, CITAS y PACIENTES:

- No guardamos `nombrePaciente` en PAGOS
- Solo guardamos `idPaciente` (FK) y hacemos JOIN cuando necesitamos el nombre
- Evitamos redundancia y dependencias transitivas

FORMA NORMAL DE BOYCE-CODD (FNBC)

Requisitos:

- Debe cumplir 3FN
- Para cada dependencia funcional X → Y, X debe ser superclave

✓ **CUMPLIMIENTO:** Mis tablas cumplen con FNBC:

- Todas las dependencias funcionales tienen como determinante una superclave (PK)
- No hay casos donde un atributo no clave determine a otro

DIAGRAMA DE NORMALIZACIÓN

PROCESO DE NORMALIZACIÓN - EJEMPLO TABLA CITAS

FORMA NO NORMALIZADA:

CITAS_SIN_NORMALIZAR:
- idCita
- nombrePaciente
- curpPaciente
- telefonoPaciente
- nombreMedico
- especialidadMedico
- cedulaMedico
- fechaCita
- motivoConsulta

Problemas:

- Redundancia de datos
- Anomalías de inserción, actualización y eliminación
- Desperdicio de espacio

APLICANDO 1FN:

- Todos los campos ya son atómicos
- Identificamos que hay grupos de datos que se repiten

APLICANDO 2FN:

- Separamos datos de pacientes en su propia tabla
- Separamos datos de médicos en su propia tabla

APLICANDO 3FN:

- Separamos datos de especialidades en su propia tabla
- Eliminamos dependencias transitivas

RESULTADO NORMALIZADO:

PACIENTES:
- idPaciente (PK)
- nombrePaciente
- curpPaciente
- telefonoPaciente
- ... (otros datos del paciente)

MEDICOS:
- idMedico (PK)
- nombreMedico
- cedulaMedico
- especialidadId (FK)
- ... (otros datos del médico)

ESPECIALIDADES:
- idEspecialidad (PK)
- nombreEspecialidad
- descripcion

CITAS:

- idCita (PK)
- idPaciente (FK)
- idMedico (FK)
- fechaCita
- motivoConsulta
- estadoCita

Beneficios:

- Sin redundancia
- Sin anomalías
- Integridad referencial
- Optimización de espacio

SCRIPTS CRUD COMPLETOS

1. ESPECIALIDADES

CREATE (Crear)

```
INSERT INTO especialidades (nombreEspecialidad, descripcion)
VALUES ('Cardiologia', 'Especialidad del corazon');
```

READ (Leer)

```
-- Obtener todas las especialidades
SELECT * FROM especialidades ORDER BY nombreEspecialidad;

-- Obtener una especialidad específica
SELECT * FROM especialidades WHERE idEspecialidad = 1;

-- Buscar especialidades por nombre
SELECT * FROM especialidades
WHERE nombreEspecialidad LIKE '%cardio%';
```

UPDATE (Actualizar)

```
UPDATE especialidades
SET nombreEspecialidad = 'Cardiologia Avanzada',
```

```
    descripcion = 'Especialidad avanzada del corazon y sistema
cardiovascular'
WHERE idEspecialidad = 1;
```

DELETE (Eliminar)

```
-- Primero verificar que no haya médicos asociados
SELECT COUNT(*) FROM medicos WHERE especialidadId = 1;

-- Si no hay médicos asociados, eliminar
DELETE FROM especialidades WHERE idEspecialidad = 1;
```

2. MEDICOS

CREATE (Crear)

```
INSERT INTO medicos (
    nombreCompleto,
    cedulaProfesional,
    especialidadId,
    telefono,
    correoElectronico,
    horarioAtencion,
    estatus
) VALUES (
    'Dr. Juan Perez Martinez',
    'CED12345678',
    1,
    '555-1234',
    'juan.perez@clinica.com',
    'Lun-Vie 9:00-17:00',
    1
);
```

READ (Leer)

```
-- Obtener todos los médicos con su especialidad
SELECT
    m.idMedico,
```

```
    m.nombreCompleto,
    m.cedulaProfesional,
    e.nombreEspecialidad,
    m.telefono,
    m.correoElectronico,
    m.horarioAtencion,
    m.estatus
FROM medicos m
LEFT JOIN especialidades e ON m.especialidadId = e.idEspecialidad
ORDER BY m.nombreCompleto;

-- Obtener un médico específico
SELECT
    m.*,
    e.nombreEspecialidad
FROM medicos m
LEFT JOIN especialidades e ON m.especialidadId = e.idEspecialidad
WHERE m.idMedico = 1;

-- Buscar médicos por nombre o especialidad
SELECT
    m.*,
    e.nombreEspecialidad
FROM medicos m
LEFT JOIN especialidades e ON m.especialidadId = e.idEspecialidad
WHERE m.nombreCompleto LIKE '%juan%'
   OR e.nombreEspecialidad LIKE '%cardio%';

-- Obtener solo médicos activos
SELECT * FROM medicos WHERE estatus = 1;

-- Contar médicos por especialidad
SELECT
    e.nombreEspecialidad,
    COUNT(m.idMedico) as totalMedicos
FROM especialidades e
LEFT JOIN medicos m ON e.idEspecialidad = m.especialidadId
GROUP BY e.idEspecialidad, e.nombreEspecialidad;
```

UPDATE (Actualizar)

```
UPDATE medicos
SET nombreCompleto = 'Dr. Juan Perez Martinez',
   cedulaProfesional = 'CED12345678',
   especialidadId = 2,
   telefono = '555-5678',
   correoElectronico = 'juan.perez@nuevaclinica.com',
   horarioAtencion = 'Lun-Vie 10:00-18:00',
   estatus = 1
WHERE idMedico = 1;

-- Cambiar solo el estado
UPDATE medicos SET estatus = 0 WHERE idMedico = 1;

-- Cambiar solo la especialidad
UPDATE medicos SET especialidadId = 3 WHERE idMedico = 1;
```

DELETE (Eliminar)

```
-- Verificar si tiene citas asociadas
SELECT COUNT(*) FROM citas WHERE idMedico = 1;

-- Si no tiene citas, eliminar
DELETE FROM medicos WHERE idMedico = 1;
```

3. TARIFAS (SERVICIOS)

CREATE (Crear)

```
-- Con especialidad asociada
INSERT INTO tarifas (
   descripcionServicio,
   costoBase,
   especialidadId,
   estatus
) VALUES (
   'Consulta General',
   500.00,
   1,
```

```
    1
);

-- Sin especialidad (servicio general)
INSERT INTO tarifas (
    descripcionServicio,
    costoBase,
    especialidadId,
    estatus
) VALUES (
    'Consulta de Urgencia',
    800.00,
    NULL,
    1
);
```

READ (Leer)

```
-- Obtener todas las tarifas con especialidad
SELECT
    t.idTarifa,
    t.descripcionServicio,
    t.costoBase,
    COALESCE(e.nombreEspecialidad, 'General') as especialidad,
    t.estatus
FROM tarifas t
LEFT JOIN especialidades e ON t.especialidadId = e.idEspecialidad
ORDER BY t.descripcionServicio;

-- Obtener una tarifa específica
SELECT
    t.*,
    e.nombreEspecialidad
FROM tarifas t
LEFT JOIN especialidades e ON t.especialidadId = e.idEspecialidad
WHERE t.idTarifa = 1;

-- Buscar tarifas por descripción o especialidad
SELECT
    t.*,
```

```
    e.nombreEspecialidad
FROM tarifas t
LEFT JOIN especialidades e ON t.especialidadId = e.idEspecialidad
WHERE t.descripcionServicio LIKE '%consulta%'
   OR e.nombreEspecialidad LIKE '%cardio%';

-- Obtener solo tarifas activas
SELECT * FROM tarifas WHERE estatus = 1;

-- Obtener tarifas por rango de precio
SELECT * FROM tarifas
WHERE costoBase BETWEEN 300 AND 800
ORDER BY costoBase;

-- Obtener promedio de costos por especialidad
SELECT
    COALESCE(e.nombreEspecialidad, 'General') as especialidad,
    AVG(t.costoBase) as costoPromedio,
    MIN(t.costoBase) as costoMinimo,
    MAX(t.costoBase) as costoMaximo,
    COUNT(t.idTarifa) as totalServicios
FROM tarifas t
LEFT JOIN especialidades e ON t.especialidadId = e.idEspecialidad
GROUP BY e.idEspecialidad, e.nombreEspecialidad;
```

UPDATE (Actualizar)

```
UPDATE tarifas
SET descripcionServicio = 'Consulta General Especializada',
    costoBase = 600.00,
    especialidadId = 2,
    estatus = 1
WHERE idTarifa = 1;

-- Aumentar 10% a todas las tarifas
UPDATE tarifas SET costoBase = costoBase * 1.10;

-- Cambiar solo el estado
UPDATE tarifas SET estatus = 0 WHERE idTarifa = 1;
```

DELETE (Eliminar)

```
DELETE FROM tarifas WHERE idTarifa = 1;

-- Eliminar tarifas inactivas
DELETE FROM tarifas WHERE estatus = 0;
```

4. PACIENTES

CREATE (Crear)

```
INSERT INTO pacientes (
   nombreCompleto,
   curp,
   fechaNacimiento,
   sexo,
   telefono,
   correoElectronico,
   direccion,
   contactoEmergencia,
   telefonoEmergencia,
   alergias,
   antecedentesMedicos,
   estatus
) VALUES (
   'Maria Lopez Gomez',
   'LOGM900101MDFLPR08',
   '1990-01-01',
   'F',
   '555-9876',
   'maria.lopez@email.com',
   'Calle Principal 123',
   'Pedro Lopez',
   '555-1111',
   'Penicilina, Polen',
   'Diabetes tipo 2, Hipertension',
   1
);
```

READ (Leer)

```
-- Obtener todos los pacientes
SELECT * FROM pacientes ORDER BY nombreCompleto;

-- Obtener un paciente específico
SELECT * FROM pacientes WHERE idPaciente = 1;

-- Buscar por CURP
SELECT * FROM pacientes WHERE curp = 'LOGM900101MDFLPR08';

-- Buscar por nombre, CURP o teléfono
SELECT * FROM pacientes
WHERE nombreCompleto LIKE '%maria%'
  OR curp LIKE '%LOGM%'
  OR telefono LIKE '%555%';

-- Obtener solo pacientes activos
SELECT * FROM pacientes WHERE estatus = 1;

-- Obtener pacientes por sexo
SELECT * FROM pacientes WHERE sexo = 'F';

-- Obtener pacientes con alergias
SELECT * FROM pacientes WHERE alergias IS NOT NULL AND alergias != '';

-- Contar pacientes por edad
SELECT
  CASE
    WHEN YEAR(CURDATE()) - YEAR(fechaNacimiento) < 18 THEN 'Menor'
    WHEN YEAR(CURDATE()) - YEAR(fechaNacimiento) BETWEEN 18 AND
60 THEN 'Adulto'
    ELSE 'Mayor'
  END as rangoEdad,
  COUNT(*) as total
FROM pacientes
GROUP BY rangoEdad;
```

UPDATE (Actualizar)

```
UPDATE pacientes
SET nombreCompleto = 'Maria Lopez de Martinez',
    telefono = '555-9999',
    correoElectronico = 'maria.martinez@email.com',
    direccion = 'Avenida Central 456',
    alergias = 'Penicilina, Polen, Mariscos'
WHERE idPaciente = 1;

-- Cambiar solo el estado
UPDATE pacientes SET estatus = 0 WHERE idPaciente = 1;
```

DELETE (Eliminar)

```
-- Verificar si tiene citas o pagos
SELECT
    (SELECT COUNT(*) FROM citas WHERE idPaciente = 1) as citas,
    (SELECT COUNT(*) FROM pagos WHERE idPaciente = 1) as pagos;

-- Si no tiene referencias, eliminar
DELETE FROM pacientes WHERE idPaciente = 1;
```

5. CITAS

CREATE (Crear)

```
INSERT INTO citas (
    idPaciente,
    idMedico,
    fechaCita,
    motivoConsulta,
    estadoCita,
    observaciones
) VALUES (
    1,
    2,
    '2025-01-15 10:00:00',
    'Dolor de cabeza constante',
    'Programada',
    'Primera consulta'
```

);

READ (Leer)

```
-- Obtener todas las citas con información completa
SELECT
    c.idCita,
    p.nombreCompleto as paciente,
    m.nombreCompleto as medico,
    e.nombreEspecialidad as especialidad,
    c.fechaCita,
    c.motivoConsulta,
    c.estadoCita,
    c.observaciones
FROM citas c
INNER JOIN pacientes p ON c.idPaciente = p.idPaciente
INNER JOIN medicos m ON c.idMedico = m.idMedico
LEFT JOIN especialidades e ON m.especialidadId = e.idEspecialidad
ORDER BY c.fechaCita DESC;

-- Obtener citas de hoy
SELECT
    c.*,
    p.nombreCompleto as paciente,
    m.nombreCompleto as medico
FROM citas c
INNER JOIN pacientes p ON c.idPaciente = p.idPaciente
INNER JOIN medicos m ON c.idMedico = m.idMedico
WHERE DATE(c.fechaCita) = CURDATE()
ORDER BY c.fechaCita;

-- Obtener citas por estado
SELECT * FROM citas WHERE estadoCita = 'Programada';

-- Obtener citas de un médico específico
SELECT
    c.*,
    p.nombreCompleto as paciente
FROM citas c
INNER JOIN pacientes p ON c.idPaciente = p.idPaciente
```

```
WHERE c.idMedico = 1
ORDER BY c.fechaCita DESC;

-- Obtener citas de un paciente específico
SELECT
   c.*,
   m.nombreCompleto as medico
FROM citas c
INNER JOIN medicos m ON c.idMedico = m.idMedico
WHERE c.idPaciente = 1
ORDER BY c.fechaCita DESC;

-- Contar citas por estado
SELECT
   estadoCita,
   COUNT(*) as total
FROM citas
GROUP BY estadoCita;
```

UPDATE (Actualizar)

```
UPDATE citas
SET fechaCita = '2025-01-16 11:00:00',
   motivoConsulta = 'Seguimiento - Dolor de cabeza',
   estadoCita = 'Atendida',
   observaciones = 'Paciente reporta mejoría'
WHERE idCita = 1;

-- Cambiar solo el estado
UPDATE citas SET estadoCita = 'Cancelada' WHERE idCita = 1;

-- Reagendar cita
UPDATE citas SET fechaCita = '2025-01-20 14:00:00' WHERE idCita = 1;
```

DELETE (Eliminar)

```
-- Verificar si tiene pagos asociados
SELECT COUNT(*) FROM pagos WHERE idCita = 1;
```

```
-- Si no tiene pagos, eliminar
DELETE FROM citas WHERE idCita = 1;
```

6. PAGOS

CREATE (Crear)

```
INSERT INTO pagos (
    idCita,
    idPaciente,
    monto,
    metodoPago,
    referencia,
    estatusPago
) VALUES (
    1,
    1,
    500.00,
    'Efectivo',
    '',
    'Pagado'
);

-- Pago con tarjeta
INSERT INTO pagos (
    idCita,
    idPaciente,
    monto,
    metodoPago,
    referencia,
    estatusPago
) VALUES (
    2,
    1,
    600.00,
    'Tarjeta',
    'VISA-1234',
    'Pagado'
);
```

READ (Leer)

```
-- Obtener todos los pagos con información completa
SELECT
   pg.idPago,
   pg.idCita,
   p.nombreCompleto as paciente,
   pg.monto,
   pg.metodoPago,
   pg.fechaPago,
   pg.referencia,
   pg.estatusPago
FROM pagos pg
INNER JOIN pacientes p ON pg.idPaciente = p.idPaciente
ORDER BY pg.fechaPago DESC;

-- Obtener pagos de hoy
SELECT
   pg.*,
   p.nombreCompleto as paciente
FROM pagos pg
INNER JOIN pacientes p ON pg.idPaciente = p.idPaciente
WHERE DATE(pg.fechaPago) = CURDATE();

-- Obtener pagos pendientes
SELECT
   pg.*,
   p.nombreCompleto as paciente,
   c.fechaCita
FROM pagos pg
INNER JOIN pacientes p ON pg.idPaciente = p.idPaciente
INNER JOIN citas c ON pg.idCita = c.idCita
WHERE pg.estatusPago = 'Pendiente'
ORDER BY c.fechaCita;

-- Obtener pagos por método
SELECT * FROM pagos WHERE metodoPago = 'Efectivo';

-- Calcular ingresos totales
SELECT
```

```
    SUM(monto) as ingresoTotal,
    COUNT(*) as totalPagos
FROM pagos
WHERE estatusPago = 'Pagado';

-- Ingresos por método de pago
SELECT
    metodoPago,
    SUM(monto) as total,
    COUNT(*) as cantidad
FROM pagos
WHERE estatusPago = 'Pagado'
GROUP BY metodoPago;

-- Ingresos por fecha
SELECT
    DATE(fechaPago) as fecha,
    SUM(monto) as ingresos,
    COUNT(*) as numeroPagos
FROM pagos
WHERE estatusPago = 'Pagado'
GROUP BY DATE(fechaPago)
ORDER BY fecha DESC;

-- Pagos de un paciente específico
SELECT * FROM pagos
WHERE idPaciente = 1
ORDER BY fechaPago DESC;
```

UPDATE (Actualizar)

```
UPDATE pagos
SET monto = 550.00,
    metodoPago = 'Transferencia',
    referencia = 'TRANS-789456',
    estatusPago = 'Pagado'
WHERE idPago = 1;

-- Cambiar solo el estado
UPDATE pagos SET estatusPago = 'Pagado' WHERE idPago = 1;
```

```
-- Cambiar estado a cancelado
UPDATE pagos SET estatusPago = 'Cancelado' WHERE idPago = 1;
```

DELETE (Eliminar)

```
DELETE FROM pagos WHERE idPago = 1;

-- Eliminar pagos cancelados
DELETE FROM pagos WHERE estatusPago = 'Cancelado';
```

CONSULTAS AVANZADAS Y REPORTES

Reporte Financiero Completo

```
SELECT
    DATE_FORMAT(fechaPago, '%Y-%m') as mes,
    metodoPago,
    SUM(monto) as total,
    COUNT(*) as cantidad,
    AVG(monto) as promedio
FROM pagos
WHERE estatusPago = 'Pagado'
GROUP BY DATE_FORMAT(fechaPago, '%Y-%m'), metodoPago
ORDER BY mes DESC, total DESC;
```

Citas Pendientes por Médico

```
SELECT
    m.nombreCompleto as medico,
    e.nombreEspecialidad as especialidad,
    COUNT(c.idCita) as citasProgramadas,
    DATE(MIN(c.fechaCita)) as proximaCita
FROM medicos m
LEFT JOIN citas c ON m.idMedico = c.idMedico AND c.estadoCita =
'Programada'
LEFT JOIN especialidades e ON m.especialidadId = e.idEspecialidad
WHERE m.estatus = 1
```

```
GROUP BY m.idMedico, m.nombreCompleto, e.nombreEspecialidad
ORDER BY citasProgramadas DESC;
```

Pacientes con Pagos Pendientes

```
SELECT
   p.nombreCompleto as paciente,
   p.telefono,
   COUNT(pg.idPago) as pagosPendientes,
   SUM(pg.monto) as montoTotal
FROM pacientes p
INNER JOIN pagos pg ON p.idPaciente = pg.idPaciente
WHERE pg.estatusPago = 'Pendiente'
GROUP BY p.idPaciente, p.nombreCompleto, p.telefono
ORDER BY montoTotal DESC;
```

Médicos Más Solicitados

```
SELECT
   m.nombreCompleto as medico,
   e.nombreEspecialidad as especialidad,
   COUNT(c.idCita) as totalCitas,
   COUNT(CASE WHEN c.estadoCita = 'Atendida' THEN 1 END) as
citasAtendidas,
   SUM(pg.monto) as ingresosGenerados
FROM medicos m
LEFT JOIN citas c ON m.idMedico = c.idMedico
LEFT JOIN especialidades e ON m.especialidadId = e.idEspecialidad
LEFT JOIN pagos pg ON c.idCita = pg.idCita AND pg.estatusPago = 'Pagado'
WHERE m.estatus = 1
GROUP BY m.idMedico, m.nombreCompleto, e.nombreEspecialidad
ORDER BY totalCitas DESC
LIMIT 10;
```