Reporte de Normalización de Base de Datos

Sistema de Requisiciones y Presiones de Pago

The Fuentes Corporation Mexican Filial SA de CV

Base de datos: `fuente_group_prod`

Elaborado por: Mauro Ram
Repositorio: Mauro-Ram/Requisiciones

23 de febrero de 2026

Índice

1 Resumen Ejecutivo

Se realizó un proceso completo de normalización, limpieza e integridad referencial sobre la base de datos `fuente_group_prod` del sistema de requisiciones de The Fuentes Corporation. El proceso se ejecutó en **5 fases**, sin afectar la funcionalidad del sistema PHP en producción.

1.1 Alcance del trabajo

- **Fase 1:** Diagnóstico de integridad (registros huérfanos, duplicados, datos sucios)
- **Fase 2:** Limpieza de datos + creación de 6 tablas catálogo
- **Fase 3:** Foreign Keys + UNIQUE INDEX + columnas nuevas con mapeo automático
- **Fase 4:** Triggers de sincronización, auto-cálculo y vista de reportería
- **Fase 5:** Bitácora de auditoría, catálogos adicionales, vistas KPI e índices de rendimiento

1.2 Resultados principales

Componente	**Antes**	**Después**
Foreign Keys	0 verificadas	17 activas
Tablas catálogo	0	9 tablas
Triggers automáticos	0	5 triggers
Vistas de reportería	0	3 vistas
Índices de rendimiento	Básicos	+5 índices optimizados
Registros duplicados	11	0
UNIQUE INDEX	0	1 (anti-duplicados)
Datos sucios	3 detectados	0
Mapeo estatus normalizado	0 %	100 % (9,837/9,837)
Mapeo forma de pago	0 %	100 % (9,837/9,837)
Bitácora de cambios	Desactivada	Automática por trigger

Cuadro 1: Comparativa antes y después de la normalización

2 Estado Inicial de la Base de Datos

2.1 Volumen de datos al inicio

Tabla	Registros
bancos	98
emisores	1
estadosobra	33
obras	26
presiones	1,390
requisiciones	2,570
hojasrequisicion	9,837
itemrequisicion	16,441
requisicionesligadas	18,830
provedores	1,612
users	35

Cuadro 2: Volumen de datos antes de la normalización

2.2 Problemas detectados en el diagnóstico

1. **0 registros huérfanos** — La integridad referencial estaba intacta a nivel de datos
2. **11 registros duplicados** en `requisicionesligadas` (10 grupos: 9 dobles + 1 triple)
3. **Banco ID 69** con saltos de línea (`\r\n`) en el nombre comercial
4. **Banco ID 95** completamente vacío (sin razón social ni nombre comercial)
5. **1 registro** con estatus `NUEVA` en lugar de `NUEVO` en `hojasrequisicion`
6. **Sin Foreign Keys** verificadas entre las tablas principales
7. **Sin catálogos** normalizados — estatus, formas de pago y bancos como VARCHAR libre
8. **Sin auditoría** — sistema de logs comentado en el código PHP

3 Fase 1 — Diagnóstico de Integridad

3.1 Objetivo

Verificar la integridad referencial de todas las tablas antes de crear Foreign Keys, identificar registros huérfanos, duplicados y datos sucios.

3.2 Script de diagnóstico

```
USE `fuente_group_prod`;

-- Verificar huerfanos: hojasrequisicion -> requisiciones
SELECT 'HUERFANOS hoja->requisicion' AS verificacion, COUNT(*) AS total
FROM `hojasrequisicion` h
LEFT JOIN `requisiciones` r ON r.requisicion_id =
   h.hojaRequisicion_idReq
WHERE r.requisicion_id IS NULL;

-- Verificar huerfanos: hojasrequisicion -> emisores
SELECT 'HUERFANOS hoja->emisor' AS verificacion, COUNT(*) AS total
FROM `hojasrequisicion` h
LEFT JOIN `emisores` e ON e.emisor_id = h.hojaRequisicion_empresa
WHERE e.emisor_id IS NULL;

-- Verificar huerfanos: hojasrequisicion -> provedores
SELECT 'HUERFANOS hoja->proveedor' AS verificacion, COUNT(*) AS total
FROM `hojasrequisicion` h
LEFT JOIN `provedores` p ON p.proveedor_id = h.hojaRequisicion_proveedor
WHERE p.proveedor_id IS NULL;

-- Verificar huerfanos: itemrequisicion -> hojasrequisicion
SELECT 'HUERFANOS item->hoja' AS verificacion, COUNT(*) AS total
FROM `itemrequisicion` i
LEFT JOIN `hojasrequisicion` h
  ON h.hojaRequisicion_id = i.itemRequisicion_idHoja
WHERE h.hojaRequisicion_id IS NULL;

-- Verificar huerfanos: requisiciones -> obras
SELECT 'HUERFANOS requisicion->obra' AS verificacion, COUNT(*) AS total
FROM `requisiciones` r
LEFT JOIN `obras` o ON o.obras_id = r.requisicion_Obra
WHERE o.obras_id IS NULL;

-- Verificar huerfanos: presiones -> obras
SELECT 'HUERFANOS presion->obra' AS verificacion, COUNT(*) AS total
FROM `presiones` p
LEFT JOIN `obras` o ON o.obras_id = p.presiones_obra
WHERE o.obras_id IS NULL;

-- Verificar huerfanos: obras -> estadosobra
SELECT 'HUERFANOS obra->estado' AS verificacion, COUNT(*) AS total
FROM `obras` o
LEFT JOIN `estadosobra` e ON e.ciudadesObras_id = o.obras_cuidad
WHERE e.ciudadesObras_id IS NULL;

-- Verificar huerfanos: requisicionesligadas -> presiones
```

```sql
SELECT 'HUERFANOS ligada->presion' AS verificacion, COUNT(*) AS total
FROM `requisicionesligadas` rl
LEFT JOIN `presiones` p
  ON p.presiones_id = rl.requisicionesLigada_presionID
WHERE p.presiones_id IS NULL;

-- Verificar huerfanos: requisicionesligadas -> requisiciones
SELECT 'HUERFANOS ligada->requisicion' AS verificacion, COUNT(*) AS
    total
FROM `requisicionesligadas` rl
LEFT JOIN `requisiciones` r
  ON r.requisicion_id = rl.requisicionesLigadas_requisicionID
WHERE r.requisicion_id IS NULL;

-- Verificar huerfanos: requisicionesligadas -> hojasrequisicion
SELECT 'HUERFANOS ligada->hoja' AS verificacion, COUNT(*) AS total
FROM `requisicionesligadas` rl
LEFT JOIN `hojasrequisicion` h
  ON h.hojaRequisicion_id = rl.requisicionesLigadas_hojaID
WHERE h.hojaRequisicion_id IS NULL;
```

Listing 1: Fase 1: Verificación de huérfanos en todas las relaciones

```sql
-- Verificar duplicados en requisicionesligadas
SELECT requisicionesLigada_presionID,
       requisicionesLigadas_requisicionID,
       requisicionesLigadas_hojaID,
       COUNT(*) AS veces
FROM `requisicionesligadas`
GROUP BY requisicionesLigada_presionID,
         requisicionesLigadas_requisicionID,
         requisicionesLigadas_hojaID
HAVING COUNT(*) > 1;
```

Listing 2: Fase 1: Detección de duplicados en requisicionesligadas

3.3 Resultado

Todas las verificaciones de huérfanos devolvieron **0**, confirmando que los datos estaban relacionados correctamente. Se detectaron **10 grupos de duplicados** (11 registros extra) en `requisicionesligadas`.

4 Fase 2 — Limpieza de Datos y Creación de Catálogos

4.1 Objetivo

Limpiar datos sucios, eliminar duplicados y crear tablas catálogo normalizadas para reemplazar los valores VARCHAR libres.

4.2 Limpieza de datos sucios

```
USE `fuente_group_prod`;

-- Banco 69: Quitar saltos de linea del nombre comercial
UPDATE `bancos`
SET `banco_nombreComercial` = TRIM(REPLACE(REPLACE(
    `banco_nombreComercial`, '\r', ''), '\n', ''))
WHERE `banco_id` = 69;

-- Banco 95: Completamente vacio, desactivarlo
UPDATE `bancos`
SET `banco_activo` = 0
WHERE `banco_id` = 95
  AND `banco_razonSocial` = ''
  AND `banco_nombreComercial` = '';

-- Corregir "NUEVA" a "NUEVO" en hojasrequisicion
UPDATE `hojasrequisicion`
SET `hojaRequisicion_estatus` = 'NUEVO'
WHERE `hojaRequisicion_estatus` = 'NUEVA';
```

Listing 3: Fase 2: Limpieza de bancos y corrección de typo

4.3 Eliminación de duplicados

```
-- PRIMERA PASADA (elimina 10 duplicados)
DELETE rl FROM `requisicionesligadas` rl
INNER JOIN (
    SELECT MAX(requisicionesLigada_id) AS id_duplicado
    FROM `requisicionesligadas`
    GROUP BY requisicionesLigada_presionID,
             requisicionesLigadas_requisicionID,
             requisicionesLigadas_hojaID
    HAVING COUNT(*) > 1
) dups ON rl.requisicionesLigada_id = dups.id_duplicado;

-- SEGUNDA PASADA (elimina el triple restante)
DELETE rl FROM `requisicionesligadas` rl
INNER JOIN (
    SELECT MAX(requisicionesLigada_id) AS id_duplicado
    FROM `requisicionesligadas`
    GROUP BY requisicionesLigada_presionID,
             requisicionesLigadas_requisicionID,
             requisicionesLigadas_hojaID
    HAVING COUNT(*) > 1
```

```
) dups ON rl.requisicionesLigada_id = dups.id_duplicado;
```

Listing 4: Fase 2: Eliminación de 11 registros duplicados

4.4 Catálogos creados

```
CREATE TABLE `cat_estatus_hoja` (
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `nombre` VARCHAR(30) NOT NULL,
    `descripcion` VARCHAR(100) DEFAULT NULL,
    `orden_flujo` TINYINT UNSIGNED NOT NULL,
    `permite_edicion` TINYINT(1) NOT NULL DEFAULT 0,
    `permite_pago` TINYINT(1) NOT NULL DEFAULT 0,
    `color_hex` VARCHAR(7) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_nombre` (`nombre`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_estatus_hoja`
  (`nombre`,`descripcion`,`orden_flujo`,
   `permite_edicion`,`permite_pago`,`color_hex`)
VALUES
  ('NUEVO','Recien creada',1,1,0,'#6c757d'),
  ('REVISION','En validacion por tesoreria',2,0,0,'#ffc107'),
  ('LIGADA','Vinculada a presion de pago',3,0,0,'#17a2b8'),
  ('AUTORIZADA','Aprobada para pago',4,0,1,'#007bff'),
  ('PAGADA','Pago completado',5,0,0,'#28a745'),
  ('RECHAZADA','No procede el pago',6,0,0,'#dc3545'),
  ('PENDIENTE','En espera de fondos',7,0,1,'#fd7e14');
```

Listing 5: Fase 2: Catálogo de estatus de hoja

```
CREATE TABLE `cat_forma_pago` (
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `nombre` VARCHAR(50) NOT NULL,
    `activo` TINYINT(1) NOT NULL DEFAULT 1,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_nombre` (`nombre`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_forma_pago` (`nombre`) VALUES
    ('Efectivo'), ('Transferencia'), ('Transaccion');

CREATE TABLE `cat_tipo_gasto` (
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `clave` VARCHAR(5) NOT NULL,
    `nombre` VARCHAR(50) NOT NULL,
    `activo` TINYINT(1) NOT NULL DEFAULT 1,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_clave` (`clave`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_tipo_gasto` (`clave`, `nombre`) VALUES
    ('MAT','Material'), ('EQH','Equipo/Maquinaria'),
    ('IND','Indirectos'), ('MO','Mano de Obra');

CREATE TABLE `cat_moneda` (
```

```
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `codigo` VARCHAR(3) NOT NULL,
    `nombre` VARCHAR(30) NOT NULL,
    `simbolo` VARCHAR(5) NOT NULL DEFAULT '$',
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_codigo` (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_moneda` (`codigo`,`nombre`,`simbolo`) VALUES
    ('MXN','Peso Mexicano','$'),
    ('USD','Dolar Estadounidense','USD');

CREATE TABLE `cat_alias_presion` (
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `nombre` VARCHAR(30) NOT NULL,
    `activo` TINYINT(1) NOT NULL DEFAULT 1,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_nombre` (`nombre`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_alias_presion` (`nombre`) VALUES
    ('Acarreo'), ('Indirectos'), ('Maquinaria');
```

Listing 6: Fase 2: Catálogos de forma de pago, tipo de gasto, moneda y alias

```
CREATE TABLE `cuentas_pago` (
    `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `nombre_corto` VARCHAR(30) NOT NULL,
    `descripcion` VARCHAR(100) DEFAULT NULL,
    `tipo` ENUM('TRANSFERENCIA','EFECTIVO')
           NOT NULL DEFAULT 'TRANSFERENCIA',
    `banco_id` INT(11) DEFAULT NULL,
    `numero_cuenta` VARCHAR(20) DEFAULT NULL,
    `clabe` VARCHAR(18) DEFAULT NULL,
    `emisor_id` INT(11) DEFAULT NULL,
    `activo` TINYINT(1) NOT NULL DEFAULT 1,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_nombre_corto` (`nombre_corto`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cuentas_pago` (`nombre_corto`,`descripcion`,`tipo`)
VALUES
  ('FUENTES','Cuenta principal Fuentes','TRANSFERENCIA'),
  ('STDR','Cuenta STDR','TRANSFERENCIA'),
  ('FUENTES STDR','Combinada Fuentes+STDR','TRANSFERENCIA'),
  ('GERVISUR','Caja/Cuenta Gervisur','EFECTIVO'),
  ('EFECTIVO','Pago en efectivo general','EFECTIVO'),
  ('EFECT','Efectivo abreviado','EFECTIVO'),
  ('MH','Cuenta MH','EFECTIVO'),
  ('EFECTIVO MH','Efectivo+MH','EFECTIVO'),
  ('EFECTIVO INTECRE','Efectivo via Intecre','EFECTIVO'),
  ('MAQOR BX+','Cuenta Maqor BX+','TRANSFERENCIA'),
  ('EFECTIVO GERVISUR INTECRE','Gervisur+Intecre','EFECTIVO'),
  ('EFECTIVO GERVISUR','Efectivo Gervisur','EFECTIVO'),
  ('EFECTIVO GERVISUR STDR','Gervisur+STDR','EFECTIVO'),
  ('MAQOR','Cuenta Maqor','TRANSFERENCIA'),
  ('FEUNT','Typo de FUENTES','TRANSFERENCIA');
```

Listing 7: Fase 2: Catálogo de cuentas de pago

5 Fase 3 — Foreign Keys, UNIQUE INDEX y Columnas Nuevas

5.1 Objetivo

Establecer integridad referencial formal mediante Foreign Keys, prevenir duplicados futuros con UNIQUE INDEX, y agregar columnas nuevas vinculadas a los catálogos.

5.2 UNIQUE INDEX

```
ALTER TABLE `requisicionesligadas`
    ADD UNIQUE KEY `uk_presion_req_hoja` (
        `requisicionesLigada_presionID`,
        `requisicionesLigadas_requisicionID`,
        `requisicionesLigadas_hojaID`
    );
```

Listing 8: Fase 3: UNIQUE INDEX anti-duplicados

5.3 Foreign Keys principales

```
-- FK: obras -> estadosobra
ALTER TABLE `obras`
    ADD CONSTRAINT `fk_obras_estado`
        FOREIGN KEY (`obras_cuidad`)
        REFERENCES `estadosobra` (`ciudadesObras_id`)
        ON UPDATE CASCADE ON DELETE RESTRICT;

-- FK: requisiciones -> obras
ALTER TABLE `requisiciones`
    ADD CONSTRAINT `fk_requisiciones_obra`
        FOREIGN KEY (`requisicion_Obra`)
        REFERENCES `obras` (`obras_id`)
        ON UPDATE CASCADE ON DELETE RESTRICT;

-- FK: presiones -> obras
ALTER TABLE `presiones`
    ADD CONSTRAINT `fk_presiones_obra`
        FOREIGN KEY (`presiones_obra`)
        REFERENCES `obras` (`obras_id`)
        ON UPDATE CASCADE ON DELETE RESTRICT;

-- FK: hojasrequisicion -> requisiciones
ALTER TABLE `hojasrequisicion`
    ADD CONSTRAINT `fk_hoja_requisicion`
        FOREIGN KEY (`hojaRequisicion_idReq`)
        REFERENCES `requisiciones` (`requisicion_id`)
        ON UPDATE CASCADE ON DELETE RESTRICT;

-- FK: hojasrequisicion -> emisores
ALTER TABLE `hojasrequisicion`
    ADD CONSTRAINT `fk_hoja_emisor`
        FOREIGN KEY (`hojaRequisicion_empresa`)
        REFERENCES `emisores` (`emisor_id`)
```

```
        ON UPDATE CASCADE ON DELETE RESTRICT;

-- FK: hojasrequisicion -> provedores
ALTER TABLE `hojasrequisicion`
    ADD CONSTRAINT `fk_hoja_proveedor`
        FOREIGN KEY (`hojaRequisicion_proveedor`)
        REFERENCES `provedores` (`proveedor_id`)
        ON UPDATE CASCADE ON DELETE RESTRICT;

-- FK: itemrequisicion -> hojasrequisicion
ALTER TABLE `itemrequisicion`
    ADD CONSTRAINT `fk_item_hoja`
        FOREIGN KEY (`itemRequisicion_idHoja`)
        REFERENCES `hojasrequisicion` (`hojaRequisicion_id`)
        ON UPDATE CASCADE ON DELETE CASCADE;

-- FK: requisicionesligadas -> presiones
ALTER TABLE `requisicionesligadas`
    ADD CONSTRAINT `fk_ligada_presion`
        FOREIGN KEY (`requisicionesLigada_presionID`)
        REFERENCES `presiones` (`presiones_id`)
        ON UPDATE CASCADE ON DELETE CASCADE;

-- FK: requisicionesligadas -> requisiciones
ALTER TABLE `requisicionesligadas`
    ADD CONSTRAINT `fk_ligada_requisicion`
        FOREIGN KEY (`requisicionesLigadas_requisicionID`)
        REFERENCES `requisiciones` (`requisicion_id`)
        ON UPDATE CASCADE ON DELETE CASCADE;

-- FK: requisicionesligadas -> hojasrequisicion
ALTER TABLE `requisicionesligadas`
    ADD CONSTRAINT `fk_ligada_hoja`
        FOREIGN KEY (`requisicionesLigadas_hojaID`)
        REFERENCES `hojasrequisicion` (`hojaRequisicion_id`)
        ON UPDATE CASCADE ON DELETE CASCADE;
```

Listing 9: Fase 3: 10 Foreign Keys entre tablas operativas

5.4 Columnas nuevas en hojasrequisicion

```
ALTER TABLE `hojasrequisicion`
    ADD COLUMN `id_forma_pago` TINYINT UNSIGNED DEFAULT NULL
        AFTER `hojaRequisicion_formaPago`,
    ADD COLUMN `id_estatus` TINYINT UNSIGNED DEFAULT NULL
        AFTER `hojaRequisicion_estatus`,
    ADD COLUMN `id_cuenta_pago` INT UNSIGNED DEFAULT NULL
        AFTER `hojasRequisicion_bancoPago`,
    ADD COLUMN `id_moneda` TINYINT UNSIGNED DEFAULT 1
        AFTER `hojaRequisicion_total`,
    ADD COLUMN `tipo_cambio` DECIMAL(10,4) DEFAULT NULL
        AFTER `id_moneda`,
    ADD COLUMN `fecha_creacion` DATETIME
        DEFAULT CURRENT_TIMESTAMP,
    ADD COLUMN `fecha_actualizacion` DATETIME DEFAULT NULL
        ON UPDATE CURRENT_TIMESTAMP,
    ADD COLUMN `creado_por` INT(11) DEFAULT NULL,
```

```
    ADD COLUMN `validado_por` INT(11) DEFAULT NULL,
    ADD COLUMN `autorizado_por` INT(11) DEFAULT NULL,
    ADD COLUMN `fecha_validacion` DATETIME DEFAULT NULL,
    ADD COLUMN `fecha_autorizacion` DATETIME DEFAULT NULL;
```

Listing 10: Fase 3: Columnas vinculadas a catálogos + auditoría

5.5 Mapeo automático de datos existentes

```
-- Poblar id_forma_pago
UPDATE `hojasrequisicion` h
INNER JOIN `cat_forma_pago` c
  ON c.nombre = h.hojaRequisicion_formaPago
SET h.id_forma_pago = c.id;

-- Poblar id_estatus
UPDATE `hojasrequisicion` h
INNER JOIN `cat_estatus_hoja` c
  ON c.nombre = h.hojaRequisicion_estatus
SET h.id_estatus = c.id;

-- Poblar id_cuenta_pago
UPDATE `hojasrequisicion` h
INNER JOIN `cuentas_pago` c
  ON c.nombre_corto = h.hojasRequisicion_bancoPago
SET h.id_cuenta_pago = c.id
WHERE h.hojasRequisicion_bancoPago IS NOT NULL
  AND h.hojasRequisicion_bancoPago != '';
```

Listing 11: Fase 3: Poblado de columnas nuevas desde VARCHAR

5.6 FKs de columnas nuevas y corrección de tipo

```
ALTER TABLE `hojasrequisicion`
    ADD CONSTRAINT `fk_hoja_forma_pago`
        FOREIGN KEY (`id_forma_pago`)
        REFERENCES `cat_forma_pago` (`id`)
        ON UPDATE CASCADE ON DELETE RESTRICT,
    ADD CONSTRAINT `fk_hoja_estatus`
        FOREIGN KEY (`id_estatus`)
        REFERENCES `cat_estatus_hoja` (`id`)
        ON UPDATE CASCADE ON DELETE RESTRICT,
    ADD CONSTRAINT `fk_hoja_cuenta_pago`
        FOREIGN KEY (`id_cuenta_pago`)
        REFERENCES `cuentas_pago` (`id`)
        ON UPDATE CASCADE ON DELETE RESTRICT,
    ADD CONSTRAINT `fk_hoja_moneda`
        FOREIGN KEY (`id_moneda`)
        REFERENCES `cat_moneda` (`id`)
        ON UPDATE CASCADE ON DELETE RESTRICT;

-- Correccion: DOUBLE -> DECIMAL para precision monetaria
ALTER TABLE `hojasrequisicion`
    MODIFY `hojaRequisicion_total`
    DECIMAL(14,2) NOT NULL DEFAULT 0.00;
```

Listing 12: Fase 3: FKs hacia catálogos + DOUBLE a DECIMAL

5.7 Resultado del mapeo

Columna nueva	Total	Mapeados	%
`id_forma_pago`	9,837	9,837	100 %
`id_estatus`	9,837	9,837	100 %
`id_cuenta_pago`	9,837	5,332	54 %*

Cuadro 3: Mapeo VARCHAR a ID catálogo. *4,505 sin banco original asignado.

6 Fase 4 — Triggers de Sincronización y Vista de Reportería

6.1 Objetivo

Crear triggers que mantengan sincronizadas las columnas nuevas con los VARCHAR originales que usa el PHP, y una vista unificada para reportería.

6.2 Trigger BEFORE UPDATE (sincronización + fechas)

```
CREATE TRIGGER `trg_hoja_before_update`
BEFORE UPDATE ON `hojasrequisicion`
FOR EACH ROW
BEGIN
    IF NEW.hojaRequisicion_estatus != OLD.hojaRequisicion_estatus
    THEN
        SET NEW.id_estatus = (
            SELECT id FROM cat_estatus_hoja
            WHERE nombre = NEW.hojaRequisicion_estatus
            LIMIT 1);
    END IF;

    IF NEW.hojaRequisicion_formaPago
       != OLD.hojaRequisicion_formaPago THEN
        SET NEW.id_forma_pago = (
            SELECT id FROM cat_forma_pago
            WHERE nombre = NEW.hojaRequisicion_formaPago
            LIMIT 1);
    END IF;

    IF (NEW.hojasRequisicion_bancoPago IS NOT NULL
        AND NEW.hojasRequisicion_bancoPago != '')
       AND (NEW.hojasRequisicion_bancoPago
            != IFNULL(OLD.hojasRequisicion_bancoPago, ''))
    THEN
        SET NEW.id_cuenta_pago = (
            SELECT id FROM cuentas_pago
            WHERE nombre_corto = NEW.hojasRequisicion_bancoPago
            LIMIT 1);
    END IF;

    SET NEW.fecha_actualizacion = NOW();

    IF NEW.hojaRequisicion_estatus = 'REVISION'
       AND OLD.hojaRequisicion_estatus != 'REVISION' THEN
        SET NEW.fecha_validacion = NOW();
    END IF;

    IF NEW.hojaRequisicion_estatus = 'AUTORIZADA'
       AND OLD.hojaRequisicion_estatus != 'AUTORIZADA' THEN
        SET NEW.fecha_autorizacion = NOW();
    END IF;
END;
```

Listing 13: Fase 4: Trigger de sincronización en UPDATE

6.3 Trigger BEFORE INSERT

```
CREATE TRIGGER `trg_hoja_before_insert`
BEFORE INSERT ON `hojasrequisicion`
FOR EACH ROW
BEGIN
    IF NEW.hojaRequisicion_estatus IS NOT NULL THEN
        SET NEW.id_estatus = (
            SELECT id FROM cat_estatus_hoja
            WHERE nombre = NEW.hojaRequisicion_estatus
            LIMIT 1);
    END IF;
    IF NEW.hojaRequisicion_formaPago IS NOT NULL THEN
        SET NEW.id_forma_pago = (
            SELECT id FROM cat_forma_pago
            WHERE nombre = NEW.hojaRequisicion_formaPago
            LIMIT 1);
    END IF;
    IF NEW.hojasRequisicion_bancoPago IS NOT NULL THEN
        SET NEW.id_cuenta_pago = (
            SELECT id FROM cuentas_pago
            WHERE nombre_corto = NEW.hojasRequisicion_bancoPago
            LIMIT 1);
    END IF;
    IF NEW.id_moneda IS NULL THEN
        SET NEW.id_moneda = 1;
    END IF;
    SET NEW.fecha_creacion = NOW();
END;
```

Listing 14: Fase 4: Trigger de sincronización en INSERT

6.4 Triggers de auto-cálculo de totales

```
-- Se crean 3 triggers con la misma logica:
-- trg_item_after_insert, trg_item_after_update,
-- trg_item_after_delete

CREATE TRIGGER `trg_item_after_insert`
AFTER INSERT ON `itemrequisicion`
FOR EACH ROW
BEGIN
    UPDATE `hojasrequisicion`
    SET `hojaRequisicion_total` = (
        SELECT IFNULL(SUM(
            (`itemRequisicion_cantidad`
             * `itemRequisicion_precio`)
            + `itemRequisicion_iva`
            - `itemRequisicion_retenciones`
        ), 0)
        FROM `itemrequisicion`
        WHERE `itemRequisicion_idHoja`
              = NEW.itemRequisicion_idHoja
    )
    WHERE `hojaRequisicion_id`
          = NEW.itemRequisicion_idHoja;
```

```
END;

-- trg_item_after_update: Mismo patron con NEW
-- trg_item_after_delete: Mismo patron con OLD
```

Listing 15: Fase 4: Recálculo automático del total de hoja

6.5 Vista unificada de reportería

```
CREATE VIEW `v_presion_detalle` AS
SELECT
    p.presiones_id, p.presiones_nombre,
    p.presiones_alias, p.presiones_semana,
    o.obras_id, o.obras_nombre, o.obras_clave,
    e_obra.ciudadesObras_nombre AS obra_estado,
    r.requisicion_id, r.requisicion_Clave,
    r.requisicion_Numero,
    h.hojaRequisicion_id, h.hojaRequisicion_numero,
    h.hojaRequisicion_total, h.hojarequisicion_adeudo,
    h.hojaRequisicion_estatus,
    cfp.nombre AS forma_pago_normalizada,
    ceh.nombre AS estatus_normalizado,
    ceh.color_hex AS estatus_color,
    cp.nombre_corto AS cuenta_pago_normalizada,
    cm.codigo AS moneda,
    prov.proveedor_nombre, prov.proveedor_rfc,
    em.emisor_nombre
FROM `requisicionesligadas` rl
INNER JOIN `presiones` p
  ON p.presiones_id = rl.requisicionesLigada_presionID
INNER JOIN `requisiciones` r
  ON r.requisicion_id
     = rl.requisicionesLigadas_requisicionID
INNER JOIN `hojasrequisicion` h
  ON h.hojaRequisicion_id
     = rl.requisicionesLigadas_hojaID
INNER JOIN `obras` o
  ON o.obras_id = r.requisicion_Obra
INNER JOIN `estadosobra` e_obra
  ON e_obra.ciudadesObras_id = o.obras_cuidad
INNER JOIN `provedores` prov
  ON prov.proveedor_id = h.hojaRequisicion_proveedor
INNER JOIN `emisores` em
  ON em.emisor_id = h.hojaRequisicion_empresa
LEFT JOIN `cat_forma_pago` cfp
  ON cfp.id = h.id_forma_pago
LEFT JOIN `cat_estatus_hoja` ceh
  ON ceh.id = h.id_estatus
LEFT JOIN `cuentas_pago` cp
  ON cp.id = h.id_cuenta_pago
LEFT JOIN `cat_moneda` cm
  ON cm.id = h.id_moneda;
```

Listing 16: Fase 4: Vista v_presion_detalle

7 Fase 5 — Blindaje Final: Auditoría, Catálogos e Índices

7.1 Objetivo

Completar el blindaje de la BD con bitácora automática, catálogos adicionales, vistas de KPIs e índices de rendimiento.

7.2 Tabla de bitácora automática

```
CREATE TABLE `logs_hoja` (
    `log_id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `hojaRequisicion_id` INT(11) NOT NULL,
    `campo_modificado` VARCHAR(50) NOT NULL,
    `valor_anterior` VARCHAR(255) DEFAULT NULL,
    `valor_nuevo` VARCHAR(255) DEFAULT NULL,
    `usuario_accion` VARCHAR(100) DEFAULT NULL,
    `ip_origen` VARCHAR(45) DEFAULT NULL,
    `fecha_accion` DATETIME NOT NULL
        DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`log_id`),
    KEY `idx_hoja` (`hojaRequisicion_id`),
    KEY `idx_fecha` (`fecha_accion`),
    KEY `idx_campo` (`campo_modificado`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Listing 17: Fase 5: Tabla de logs automática

7.3 Catálogos adicionales

```
CREATE TABLE `cat_unidad_medida` (
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `clave` VARCHAR(10) NOT NULL,
    `nombre` VARCHAR(30) NOT NULL,
    `activo` TINYINT(1) NOT NULL DEFAULT 1,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_clave` (`clave`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_unidad_medida` (`clave`,`nombre`) VALUES
  ('DIS','DISENO'),('PZA','PIEZAS'),
  ('BLT','BULTOS'),('PES','PESOS'),
  ('LTS','LITROS'),('SRV','SERVICIO'),
  ('MEN','MENSUALIDAD'),('RNT','RENTA'),
  ('CUB','CUBETAS'),('TON','TONELADAS'),
  ('MTS','METROS'),('M2','METROS CUADRADOS'),
  ('M3','METROS CUBICOS'),('KG','KILOGRAMOS'),
  ('JGO','JUEGO'),('PAR','PAR'),
  ('GLB','GLOBAL'),('HRS','HORAS'),
  ('DIA','DIA'),('VJE','VIAJE'),
  ('TAM','TAMBO'),('GAL','GALONES'),
  ('ROL','ROLLO'),('SAC','SACO'),
  ('CJA','CAJA');
```

Listing 18: Fase 5: Catálogo de unidades de medida

```sql
CREATE TABLE `cat_estatus_requisicion` (
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `nombre` VARCHAR(20) NOT NULL,
    `descripcion` VARCHAR(100) DEFAULT NULL,
    `permite_edicion` TINYINT(1) NOT NULL DEFAULT 0,
    `color_hex` VARCHAR(7) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_nombre` (`nombre`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_estatus_requisicion`
  (`nombre`,`descripcion`,`permite_edicion`,`color_hex`)
VALUES
  ('ABIERTO','Permite agregar hojas',1,'#dc3545'),
  ('CERRADO','No permite cambios',0,'#28a745');

CREATE TABLE `cat_estatus_presion` (
    `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `nombre` VARCHAR(20) NOT NULL,
    `descripcion` VARCHAR(100) DEFAULT NULL,
    `color_hex` VARCHAR(7) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uk_nombre` (`nombre`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `cat_estatus_presion`
  (`nombre`,`descripcion`,`color_hex`)
VALUES
  ('PENDIENTE','Sin hojas ligadas','#ffc107'),
  ('EN PROCESO','Hojas en validacion/pago','#17a2b8'),
  ('PAGADA','Todas las hojas liquidadas','#28a745');
```

Listing 19: Fase 5: Estatus de requisición y presión

7.4 Vistas de KPIs

```sql
CREATE VIEW `v_kpi_obra` AS
SELECT
    o.obras_id, o.obras_nombre,
    e.ciudadesObras_nombre AS estado,
    COUNT(DISTINCT r.requisicion_id)
      AS total_requisiciones,
    COUNT(DISTINCT h.hojaRequisicion_id)
      AS total_hojas,
    SUM(CASE WHEN h.hojaRequisicion_estatus = 'PAGADA'
        THEN h.hojaRequisicion_total ELSE 0 END)
      AS monto_pagado,
    SUM(CASE WHEN h.hojaRequisicion_estatus = 'AUTORIZADA'
        THEN h.hojarequisicion_adeudo ELSE 0 END)
      AS monto_autorizado,
    SUM(CASE WHEN h.hojaRequisicion_estatus = 'PAGADA'
        THEN 1 ELSE 0 END) AS hojas_pagadas,
    SUM(CASE WHEN h.hojaRequisicion_estatus = 'RECHAZADA'
```

```
        THEN 1 ELSE 0 END) AS hojas_rechazadas
FROM `obras` o
INNER JOIN `estadosobra` e
  ON e.ciudadesObras_id = o.obras_cuidad
LEFT JOIN `requisiciones` r
  ON r.requisicion_Obra = o.obras_id
LEFT JOIN `hojasrequisicion` h
  ON h.hojaRequisicion_idReq = r.requisicion_id
WHERE o.obras_estatus = 'ACTIVO'
GROUP BY o.obras_id, o.obras_nombre,
         e.ciudadesObras_nombre;

CREATE VIEW `v_top_proveedores` AS
SELECT
    p.proveedor_id, p.proveedor_nombre,
    p.proveedor_rfc, p.proveedor_banco,
    COUNT(h.hojaRequisicion_id) AS total_hojas,
    SUM(h.hojaRequisicion_total) AS monto_total,
    MAX(h.hojaRequisicion_fechaPago) AS ultimo_pago
FROM `provedores` p
INNER JOIN `hojasrequisicion` h
  ON h.hojaRequisicion_proveedor = p.proveedor_id
GROUP BY p.proveedor_id, p.proveedor_nombre,
         p.proveedor_rfc, p.proveedor_banco
ORDER BY monto_total DESC;
```

Listing 20: Fase 5: Vista de KPIs por obra

7.5 Índices de rendimiento

```
ALTER TABLE `hojasrequisicion`
    ADD KEY `idx_hoja_estatus`
        (`hojaRequisicion_estatus`),
    ADD KEY `idx_hoja_fecha_pago`
        (`hojaRequisicion_fechaPago`),
    ADD KEY `idx_hoja_req_estatus`
        (`hojaRequisicion_idReq`,
         `hojaRequisicion_estatus`);

ALTER TABLE `presiones`
    ADD KEY `idx_presion_obra_estatus`
        (`presiones_obra`, `presiones_estatus`);

ALTER TABLE `requisiciones`
    ADD KEY `idx_req_obra_clave`
        (`requisicion_Obra`, `requisicion_Clave`);
```

Listing 21: Fase 5: Índices para queries frecuentes del PHP

8 Inventario Final de la Base de Datos

8.1 Tablas (28 total)

Tipo	Tabla	Registros
Operativa	obras	26
Operativa	presiones	1,390
Operativa	requisiciones	2,570
Operativa	hojasrequisicion	9,837
Operativa	itemrequisicion	16,441
Operativa	requisicionesligadas	18,819
Catálogo original	bancos	98
Catálogo original	emisores	1
Catálogo original	estadosobra	33
Catálogo original	provedores	1,612
Catálogo nuevo	cat_estatus_hoja	7
Catálogo nuevo	cat_forma_pago	3
Catálogo nuevo	cat_tipo_gasto	4
Catálogo nuevo	cat_moneda	2
Catálogo nuevo	cat_alias_presion	3
Catálogo nuevo	cat_unidad_medida	25
Catálogo nuevo	cat_estatus_requisicion	2
Catálogo nuevo	cat_estatus_presion	3
Catálogo nuevo	cuentas_pago	15
Auditoría	logs_hoja	2 (prueba)

Cuadro 4: Inventario completo de tablas

8.2 Foreign Keys (17 activas)

Constraint	Tabla.Columna	Referencia
fk_obras_estado	obras.obras_cuidad	estadosobra
fk_requisiciones_obra	requisiciones.requisicion_Obra	obras
fk_presiones_obra	presiones.presiones_obra	obras
fk_hoja_requisicion	hojasrequisicion.idReq	requisiciones
fk_hoja_emisor	hojasrequisicion.empresa	emisores
fk_hoja_proveedor	hojasrequisicion.proveedor	provedores
fk_hoja_forma_pago	hojasrequisicion.id_forma_pago	cat_forma_pago
fk_hoja_estatus	hojasrequisicion.id_estatus	cat_estatus_hoja
fk_hoja_cuenta_pago	hojasrequisicion.id_cuenta_pago	cuentas_pago
fk_hoja_moneda	hojasrequisicion.id_moneda	cat_moneda
fk_item_hoja	itemrequisicion.idHoja	hojasrequisicion
fk_ligada_presion	requisicionesligadas	presiones
fk_ligada_requisicion	requisicionesligadas	requisiciones
fk_ligada_hoja	requisicionesligadas	hojasrequisicion

Cuadro 5: Foreign Keys activas

8.3 Triggers (5 activos)

Trigger	Tabla	Evento	Función
trg_hoja_before_insert	hojasrequisicion	INSERT	Sincroniza catálogos
trg_hoja_before_update	hojasrequisicion	UPDATE	Sincroniza + logs
trg_item_after_insert	itemrequisicion	INSERT	Auto-calcula total
trg_item_after_update	itemrequisicion	UPDATE	Auto-calcula total
trg_item_after_delete	itemrequisicion	DELETE	Auto-calcula total

Cuadro 6: Triggers activos

8.4 Vistas (3 activas)

Vista	Propósito
v_presion_detalle	Consulta unificada presión, hoja y proveedor
v_kpi_obra	Dashboard ejecutivo: montos por obra y estatus
v_top_proveedores	Ranking de proveedores por monto total

Cuadro 7: Vistas de reportería

8.5 KPIs en vivo (datos reales de la BD)

Obra	**Estado**	**Monto Pagado**	**Hojas**
Oficinas Tuxtla Gutiérrez	Chiapas	$70,596,648	1,056
Matías Romero	Oaxaca	$9,071,478	73
Oficina Maqor	Chiapas	$8,426,020	36
Las Flores	San Luis Potosí	$87,671	6
Tepeyac 2	San Luis Potosí	$73,216	5

Cuadro 8: Top 5 obras por monto pagado

8.6 Diagrama de relaciones

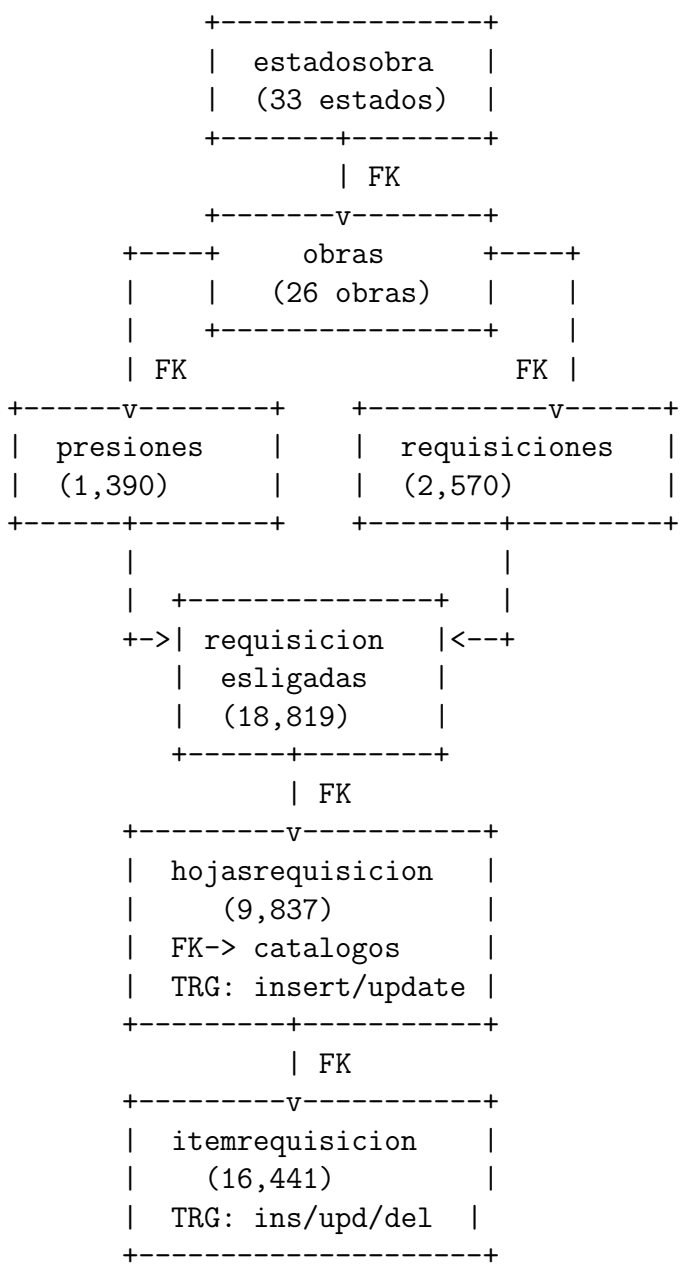

Figura 1: Diagrama de relaciones de la base de datos

9 Conclusiones

9.1 Conclusiones

La normalización se completó exitosamente en 5 fases:

- **Integridad referencial** garantizada por 17 Foreign Keys.
- **Prevención de duplicados** con UNIQUE INDEX en requisicionesligadas.
- **Normalización de datos** con 9 tablas catálogo y mapeo al 100 %.
- **Auditoría automática** con triggers y tabla de bitácora.
- **Reportería lista** con 3 vistas SQL para dashboards.
- **Cero impacto** en el código PHP actual — todo es retrocompatible.