Tablas del Modelo Lógico — Sistema de Ventas

Este documento muestra las tablas del modelo lógico en formato de tablas Markdown para facilitar la visualización.

tiendas

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_tienda	SERIAL	PK	Identificador de tienda
nombre	VARCHAR(200)		NOT NULL
direccion	VARCHAR(300)		
telefono	VARCHAR(50)		
ciudad	VARCHAR(100)		
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

puesto_empleados

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_puesto	SERIAL	PK	Identificador de puesto
nombre_puesto	VARCHAR(150)		NOT NULL
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

empleados

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_empleado	SERIAL	PK	
nombre	VARCHAR(200)		NOT NULL
apellido_paterno	VARCHAR(150)		
apellido_materno	VARCHAR(150)		
rfc	VARCHAR(20)		
fecha_contratacion	DATE		NOT NULL

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_tienda	INTEGER	FK	REFERENCES `tiendas(id_tienda)` ON DELETE SET NULL
id_puesto	INTEGER	FK	REFERENCES `puesto_empleados(id_puesto)` ON DELETE SET NULL
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

Índices: `idx_empleados_tienda (id_tienda)`, `idx_empleados_puesto (id_puesto)`

proveedores

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_proveedor	SERIAL	PK	
nombre_empresa	VARCHAR(200)		NOT NULL
contacto_nombre	VARCHAR(200)		
contacto_email	VARCHAR(150)		
contacto_telefono	VARCHAR(50)		
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

categoria_productos

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_categoria	SERIAL	PK	
nombre_categoria	VARCHAR(150)		NOT NULL
descripcion	TEXT		
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

productos

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_producto	SERIAL	PK	
sku	VARCHAR(80)	UQ	UNIQUE, índice `idx_productos_sku`

Atributo	Tipo de dato	Llave	Restricciones / Notas
nombre_producto	VARCHAR(250)		NOT NULL
descripcion	TEXT		
precio_venta	NUMERIC(12,2)		>= 0 sugerido (CHECK)
costo_compra	NUMERIC(12,2)		>= 0 sugerido (CHECK)
id_categoria	INTEGER	FK	REFERENCES `categoria_productos(id_categoria)` ON DELETE SET NULL
id_proveedor	INTEGER	FK	REFERENCES `proveedores(id_proveedor)` ON DELETE SET NULL
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

Índices: `idx_productos_sku (sku)`, `idx_productos_categoria (id_categoria)`, `idx_productos_proveedor (id_proveedor)`

inventario

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_tienda	INTEGER	PK (compuesto)	FK -> `tiendas(id_tienda)` ON DELETE CASCADE
id_producto	INTEGER	PK (compuesto)	FK -> `productos(id_producto)` ON DELETE CASCADE
cantidad	INTEGER		DEFAULT 0
fecha_ultima_actualizacion	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

Índices: PK compuesta `(id_tienda, id_producto)`, `idx_inventario_producto (id_producto)`

Nota: `fecha_ultima_actualizacion` vs `updated_at`

- `fecha_ultima_actualizacion` es una columna específica del dominio (inventario) que indica el momento en que cambió el stock (por ejemplo, la columna `cantidad`). Se usa cuando te interesa registrar la última vez que se actualizó el inventario en sí (entrada/salida de mercancía).
- `updated_at` es una columna de auditoría genérica presente en muchas tablas y mantenida por el trigger genérico `trg_set_updated_at()` (se actualiza en cualquier UPDATE de la fila). Sirve para saber cuándo cualquier campo de la fila fue modificado por última vez, no sólo el stock.

Ambas columnas pueden coexistir y tener usos distintos: `fecha_ultima_actualizacion` para eventos de inventario, y `updated_at` para auditoría general.

Ejemplo: trigger que actualiza `fecha_ultima_actualizacion` solo cuando cambia `cantidad`

El siguiente ejemplo demuestra una función y trigger que ponen `fecha_ultima_actualizacion = now()` sólo si la cantidad del inventario realmente cambió (evita actualizar la fecha cuando otras columnas se modifican):

```
CREATE OR REPLACE FUNCTION trg_inventario_set_fecha()
RETURNS TRIGGER AS $$
BEGIN
  -- Actualizar fecha_ultima_actualizacion solo si cambia la cantidad
  IF TG_OP = 'UPDATE' THEN
    IF NEW.cantidad IS DISTINCT FROM OLD.cantidad THEN
      NEW.fecha_ultima_actualizacion := now();
    END IF;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_inventario_set_fecha
  BEFORE UPDATE ON inventario
  FOR EACH ROW
  EXECUTE FUNCTION trg_inventario_set_fecha();
```

Notas:

- La columna `fecha_ultima_actualizacion` ya tiene `DEFAULT now()` para INSERTs; este trigger garantiza que en UPDATEs solo se cambie cuando varíe `cantidad`.
- El trigger genérico `trg_set_updated_at()` (que actualiza `updated_at`) puede coexistir con este trigger: ambos se ejecutarán para los UPDATEs y cada uno cumplirá su propósito.

---

clientes

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_cliente	SERIAL	PK	
nombre	VARCHAR(250)		NOT NULL
rfc	VARCHAR(13)	UQ cond.	UNIQUE WHEN NOT NULL (`ux_clientes_rfc`)
email	VARCHAR(150)		
telefono	VARCHAR(50)		
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

---

venta

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_venta	SERIAL	PK	
fecha_hora	TIMESTAMP		DEFAULT now()
monto_total	NUMERIC(14,2)		DEFAULT 0; mantenido por trigger desde `detalles_venta`
id_cliente	INTEGER	FK	REFERENCES `clientes(id_cliente)` ON DELETE SET NULL
id_empleado	INTEGER	FK	REFERENCES `empleados(id_empleado)` ON DELETE SET NULL
id_tienda	INTEGER	FK	REFERENCES `tiendas(id_tienda)` ON DELETE SET NULL
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

Índices: `idx_venta_cliente (id_cliente)`, `idx_venta_empleado (id_empleado)`, `idx_venta_tienda (id_tienda)`

detalles_venta

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_detalle_venta	SERIAL	PK	
id_venta	INTEGER	FK	REFERENCES `venta(id_venta)` ON DELETE CASCADE
id_producto	INTEGER	FK	REFERENCES `productos(id_producto)` ON DELETE RESTRICT
cantidad	INTEGER		CHECK (cantidad > 0)
precio_unitario	NUMERIC(12,2)		CHECK (precio_unitario >= 0)
subtotal	NUMERIC(14,2)		GENERATED ALWAYS AS (cantidad * precio_unitario) STORED
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

Índices: `idx_detalles_venta_venta (id_venta)`

facturacion

Atributo	Tipo de dato	Llave	Restricciones / Notas
id_factura	SERIAL	PK	Identificador de la factura
id_venta	INTEGER	FK	REFERENCES `venta(id_venta)` ON DELETE CASCADE
serie	VARCHAR(20)		Opcional

Atributo	Tipo de dato	Llave	Restricciones / Notas
folio	VARCHAR(50)		Opcional
fecha_emision	TIMESTAMP		DEFAULT now()
total	NUMERIC(14,2)		NOT NULL
metodo_pago	VARCHAR(50)		Ej: 'transferencia', 'efectivo'
estado	VARCHAR(30)		DEFAULT 'emitida'
xml_path	TEXT		Ruta o referencia al XML de la factura
pdf_path	TEXT		Ruta o referencia al PDF generado
created_at	TIMESTAMP		DEFAULT now()
updated_at	TIMESTAMP		DEFAULT now(), actualiza con trigger

Índices: `idx_facturacion_venta (id_venta)`

Triggers y funciones notables

- `trg_set_updated_at()` — función genérica que setea `NEW.updated_at = now()` en `BEFORE UPDATE`.
- Triggers `trg_*_updated_at` para las tablas: tiendas, puesto_empleados, empleados, proveedores, categoria_productos, productos, inventario, clientes, venta, detalles_venta.
- `trg_recalc_venta_monto()` — función que recalcula `venta.monto_total` (SUM(subtotal) de `detalles_venta`) y se ejecuta `AFTER INSERT/UPDATE/DELETE` en `detalles_venta`.

Notas y recomendaciones rápidas

- Ejecuta el DDL en una base de datos de desarrollo para validar triggers (requiere PostgreSQL).
- Considera agregar `created_by`/`updated_by` si necesitas auditoría por usuario.
- Añade CHECK adicionales si tienes reglas de negocio (por ejemplo: `precio_venta >= costo_compra` si aplicable).
- Para grandes volúmenes, considerar particionado de `venta` y/o `detalles_venta` por rango de fecha o por tienda.

Archivo original del diagrama: `modelo_logico.puml` Archivo DDL: `modelo_logico.sql`