|  |  |  |
| --- | --- | --- |
|  | ***Sesión*** | ***14*** |
|  | ***Tema*** | Transformación del modelo lógico al modelo físico de la base de datos. |
|  | ***Propósito*** | Dar a conocer a los participantes, los fundamentos, técnicas y procedimientos empleados para generar el modelo físico de una base de datos dada a partir del modelo lógico, mediante la correspondiente presentación y demostración de ejemplos, en una sesión expositiva-demostrativa. |
|  | ***Fecha*** | C.22.10.2025 |
|  | ***Hora*** | 19:45 |

# I Introducción

La limpieza, filtrado y normalización de datos son procesos fundamentales para garantizar la calidad y utilidad de los datos en entornos de Big Data e Inteligencia Artificial. Estos pasos permiten que los algoritmos funcionen de manera precisa, eficiente y confiable.

# II Del Modelo Lógico al Modelo Físico: Fundamentos

**¿Qué es el Modelo Físico?**

Es la representación **específica** de la base de datos para un **SGBD particular** (MySQL, PostgreSQL, Oracle, etc.). Incluye todos los detalles técnicos de implementación.

# Conceptos Principales del Modelo Físico

* 1. **Dependencia del SGBD**
     + Sintaxis específica del motor elegido
     + Tipos de datos particulares
     + Características de almacenamiento
     + Mecanismos de seguridad

# Elementos Clave del Modelo Físico

Estructuras de Almacenamiento

-- Ejemplos según SGBD

-- MySQL

TABLESPACE, ENGINE=InnoDB

-- PostgreSQL TABLESPACE, SCHEMA

-- Oracle

TABLESPACES, SEGMENTOS, EXTENSIONES

# Tipos de Datos Específicos

-- MySQL

VARCHAR(255), INT, DECIMAL(10,2), DATETIME, TEXT

-- PostgreSQL

VARCHAR, INTEGER, NUMERIC, TIMESTAMP, TEXT

-- SQL Server

NVARCHAR, INT, MONEY, DATETIME2, VARCHAR(MAX)

* 1. Componentes del Modelo Físico

# Índices y Optimización

-- Índices primarios (automáticos con PK)

-- Índices secundarios

CREATE INDEX idx\_cliente\_email ON CLIENTE(email);

-- Índices compuestos

CREATE INDEX idx\_pedido\_fecha\_cliente ON PEDIDO(fecha, id\_cliente);

# Vistas

CREATE VIEW vista\_pedidos\_clientes AS SELECT p.id\_pedido, c.nombre, p.fecha, p.total FROM PEDIDO p

JOIN CLIENTE c ON p.id\_cliente = c.id\_cliente;

# Procedimientos Almacenados y Triggers

-- Trigger para auditoría

CREATE TRIGGER audit\_cliente AFTER UPDATE ON CLIENTE FOR EACH ROW

INSERT INTO auditoria VALUES (NOW(), USER(), 'UPDATE CLIENTE');

Diferencias Clave: Modelo Lógico vs. Físico

|  |  |
| --- | --- |
| **Modelo Lógico** | **Modelo Físico** |
| Independiente del SGBD | Específico del SGBD |
| Tipos de datos genéricos | Tipos de datos específicos |
| Sin índices secundarios | Con índices de optimización |
| Sin parámetros de almacenamiento | Con parámetros de almacenamiento |
| Relaciones conceptuales | Claves foráneas con restricciones |

# Proceso de Transformación Paso 1: Elección del SGBD

* MySQL, PostgreSQL, Oracle, SQL Server, etc.
* Considerar: rendimiento, escalabilidad, costos

# Paso 2: Mapeo de Tipos de Datos Lógico: Físico (MySQL):

Texto VARCHAR(100)

Entero INT

Decimal DECIMAL(10,2) Fecha/Hora DATETIME

Booleano TINYINT(1) o BOOLEAN

# Paso 3: Definición de Esquemas y Tablespaces

*-- MySQL*

CREATE DATABASE ventas CHARACTER SET utf8mb4 COLLATE

utf8mb4\_unicode\_ci;

*-- PostgreSQL*

CREATE SCHEMA ventas;

# Paso 4: Scripts DDL Completos

*-- Ejemplo MySQL*

CREATE TABLE CLIENTE (

id\_cliente INT AUTO\_INCREMENT PRIMARY KEY, nombre VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL,

telefono VARCHAR(20),

fecha\_registro DATETIME DEFAULT CURRENT\_TIMESTAMP, INDEX idx\_email (email),

INDEX idx\_nombre (nombre)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

# Consideraciones de Rendimiento Estrategias de Indexación

* Índices en claves foráneas
* Índices en campos de búsqueda frecuente
* Índices compuestos para consultas específicas

# Particionamiento

*-- Particionamiento por rangos de fecha*

PARTITION BY RANGE (YEAR(fecha)) ( PARTITION p2023 VALUES LESS THAN (2024), PARTITION p2024 VALUES LESS THAN (2025)

);

# Configuración de Storage

* Tamaño de tablas y índices
* Configuración de buffers
* Estrategias de backup

# Ejemplo de Implementación de Modelo Físico Script Completo: Sistema de Ventas - Modelo Físico

*-- =============================================*

*-- SISTEMA DE VENTAS - IMPLEMENTACIÓN COMPLETA*

*-- Base de datos: MySQL/MariaDB*

*-- =============================================*

*-- Paso 1: Crear base de datos*

DROP DATABASE IF EXISTS sistema\_ventas; CREATE DATABASE sistema\_ventas CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci; USE sistema\_ventas;

*-- =============================================*

*-- TABLAS PRINCIPALES*

*-- =============================================*

*-- Tabla CLIENTE*

## CREATE TABLE CLIENTE (

id\_cliente INT AUTO\_INCREMENT PRIMARY KEY, nombre VARCHAR(100) NOT NULL,

apellido VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL,

telefono VARCHAR(20), direccion TEXT,

fecha\_registro DATETIME DEFAULT CURRENT\_TIMESTAMP, activo TINYINT(1) DEFAULT 1,

INDEX idx\_cliente\_email (email),

INDEX idx\_cliente\_nombre (nombre, apellido), INDEX idx\_cliente\_activo (activo)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

*-- Tabla CATEGORIA\_PRODUCTO*

## CREATE TABLE CATEGORIA\_PRODUCTO (

id\_categoria INT AUTO\_INCREMENT PRIMARY KEY, nombre VARCHAR(100) NOT NULL UNIQUE,

descripcion TEXT,

activo TINYINT(1) DEFAULT 1,

INDEX idx\_categoria\_nombre (nombre)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

*-- Tabla PRODUCTO*

## CREATE TABLE PRODUCTO (

id\_producto INT AUTO\_INCREMENT PRIMARY KEY, codigo\_sku VARCHAR(50) UNIQUE NOT NULL, nombre VARCHAR(200) NOT NULL,

descripcion TEXT, id\_categoria INT,

precio\_actual DECIMAL(10,2) NOT NULL, costo DECIMAL(10,2),

stock\_actual INT DEFAULT 0, stock\_minimo INT DEFAULT 5, imagen\_url VARCHAR(500),

fecha\_creacion DATETIME DEFAULT CURRENT\_TIMESTAMP, fecha\_actualizacion DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE

## CURRENT\_TIMESTAMP,

activo TINYINT(1) DEFAULT 1,

FOREIGN KEY (id\_categoria)

REFERENCES CATEGORIA\_PRODUCTO(id\_categoria) ON DELETE SET NULL

## ON UPDATE CASCADE,

INDEX idx\_producto\_sku (codigo\_sku), INDEX idx\_producto\_nombre (nombre),

INDEX idx\_producto\_categoria (id\_categoria), INDEX idx\_producto\_activo (activo),

INDEX idx\_producto\_stock (stock\_actual),

CHECK (precio\_actual > 0), CHECK (stock\_actual >= 0)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

*-- Tabla PEDIDO*

## CREATE TABLE PEDIDO (

id\_pedido INT AUTO\_INCREMENT PRIMARY KEY,

numero\_pedido VARCHAR(20) UNIQUE NOT NULL, fecha\_pedido DATETIME DEFAULT CURRENT\_TIMESTAMP,

fecha\_entrega DATE, id\_cliente INT NOT NULL,

subtotal DECIMAL(10,2) DEFAULT 0.00, impuestos DECIMAL(10,2) DEFAULT 0.00, total DECIMAL(10,2) DEFAULT 0.00,

estado ENUM('pendiente', 'confirmado', 'en\_proceso', 'enviado', 'entregado', 'cancelado') DEFAULT 'pendiente',

metodo\_pago ENUM('efectivo', 'tarjeta', 'transferencia', 'paypal'), notas TEXT,

FOREIGN KEY (id\_cliente) REFERENCES CLIENTE(id\_cliente) ON DELETE RESTRICT

## ON UPDATE CASCADE,

INDEX idx\_pedido\_numero (numero\_pedido), INDEX idx\_pedido\_fecha (fecha\_pedido), INDEX idx\_pedido\_cliente (id\_cliente),

INDEX idx\_pedido\_estado (estado),

INDEX idx\_pedido\_fecha\_entrega (fecha\_entrega),

CHECK (total >= 0)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

*-- Tabla DETALLE\_PEDIDO*

## CREATE TABLE DETALLE\_PEDIDO (

id\_detalle INT AUTO\_INCREMENT, id\_pedido INT NOT NULL, id\_producto INT NOT NULL, cantidad INT NOT NULL,

precio\_unitario DECIMAL(10,2) NOT NULL,

subtotal\_linea DECIMAL(10,2) GENERATED ALWAYS AS (cantidad \* precio\_unitario) STORED,

PRIMARY KEY (id\_detalle), FOREIGN KEY (id\_pedido)

REFERENCES PEDIDO(id\_pedido) ON DELETE CASCADE

## ON UPDATE CASCADE,

FOREIGN KEY (id\_producto)

REFERENCES PRODUCTO(id\_producto) ON DELETE RESTRICT

## ON UPDATE CASCADE,

UNIQUE INDEX uk\_detalle\_pedido\_producto (id\_pedido, id\_producto), INDEX idx\_detalle\_pedido (id\_pedido),

INDEX idx\_detalle\_producto (id\_producto),

CHECK (cantidad > 0), CHECK (precio\_unitario >= 0)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

*-- =============================================*

*-- TABLAS DE AUDITORÍA Y LOGS*

*-- =============================================*

## CREATE TABLE AUDITORIA\_PEDIDOS (

id\_auditoria INT AUTO\_INCREMENT PRIMARY KEY, tabla\_afectada VARCHAR(50) NOT NULL,

accion ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL,

id\_registro INT NOT NULL, datos\_anteriores JSON, datos\_nuevos JSON, usuario VARCHAR(100),

fecha\_auditoria DATETIME DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_auditoria\_tabla (tabla\_afectada), INDEX idx\_auditoria\_fecha (fecha\_auditoria), INDEX idx\_auditoria\_accion (accion)

) ENGINE=InnoDB;

## CREATE TABLE LOG\_STOCK\_PRODUCTO (

id\_log INT AUTO\_INCREMENT PRIMARY KEY,

id\_producto INT NOT NULL, stock\_anterior INT NOT NULL, stock\_nuevo INT NOT NULL, diferencia INT NOT NULL,

tipo\_movimiento ENUM('entrada', 'salida', 'ajuste') NOT NULL, motivo VARCHAR(200),

fecha\_movimiento DATETIME DEFAULT CURRENT\_TIMESTAMP, usuario VARCHAR(100),

FOREIGN KEY (id\_producto)

REFERENCES PRODUCTO(id\_producto) ON DELETE CASCADE

## ON UPDATE CASCADE,

INDEX idx\_log\_producto (id\_producto), INDEX idx\_log\_fecha (fecha\_movimiento), INDEX idx\_log\_tipo (tipo\_movimiento)

) ENGINE=InnoDB;

*-- =============================================*

*-- INSERCIÓN DE DATOS DE PRUEBA*

*-- =============================================*

*-- Insertar categorías*

INSERT INTO CATEGORIA\_PRODUCTO (nombre, descripcion) VALUES ('Electrónicos', 'Dispositivos electrónicos y gadgets'),

('Ropa', 'Prendas de vestir para hombre y mujer'), ('Hogar', 'Artículos para el hogar y decoración'), ('Deportes', 'Equipos y ropa deportiva');

*-- Insertar clientes*

INSERT INTO CLIENTE (nombre, apellido, email, telefono, direccion) VALUES

('María', 'Gonzalez', ['maria.gonzalez@email.com',](mailto:%27maria.gonzalez@email.com) '+1234567890', 'Calle Principal 123, Ciudad'), ('Carlos', 'Rodriguez', ['carlos.rodriguez@email.com',](mailto:%27carlos.rodriguez@email.com) '+1234567891', 'Avenida Central 456, Ciudad'),

('Ana', 'Martinez', ['ana.martinez@email.com',](mailto:%27ana.martinez@email.com) '+1234567892', 'Plaza Mayor 789, Ciudad'), ('Luis', 'Hernandez', ['luis.hernandez@email.com',](mailto:%27luis.hernandez@email.com) '+1234567893', 'Boulevard Norte 321, Ciudad');

*-- Insertar productos*

INSERT INTO PRODUCTO (codigo\_sku, nombre, descripcion, id\_categoria, precio\_actual, costo, stock\_actual) VALUES

('SKU-001', 'Smartphone X', 'Teléfono inteligente última generación', 1, 599.99, 450.00, 50),

('SKU-002', 'Laptop Pro', 'Laptop para trabajo y gaming', 1, 1299.99, 1000.00, 25),

('SKU-003', 'Camiseta Básica', 'Camiseta de algodón 100%', 2, 19.99, 8.50, 100),

('SKU-004', 'Zapatos Deportivos', 'Zapatos para running y ejercicio', 4, 89.99, 45.00, 75), ('SKU-005', 'Juego de Sábanas', 'Juego de sábanas de algodón egipcio', 3, 49.99, 25.00, 30), ('SKU-006', 'Auriculares Bluetooth', 'Auriculares inalámbricos con cancelación de ruido', 1, 199.99, 120.00, 40);

*-- Insertar pedidos*

INSERT INTO PEDIDO (numero\_pedido, id\_cliente, subtotal, impuestos, total, estado, metodo\_pago) VALUES

('PED-2024-001', 1, 619.98, 124.00, 743.98, 'entregado', 'tarjeta'),

('PED-2024-002', 2, 1299.99, 260.00, 1559.99, 'en\_proceso', 'transferencia'),

('PED-2024-003', 3, 109.98, 22.00, 131.98, 'pendiente', 'efectivo');

*-- Insertar detalles de pedidos*

INSERT INTO DETALLE\_PEDIDO (id\_pedido, id\_producto, cantidad, precio\_unitario) VALUES (1, 1, 1, 599.99),

(1, 3, 1, 19.99),

(2, 2, 1, 1299.99),

(3, 3, 2, 19.99),

(3, 4, 1, 89.99);

*-- =============================================*

*-- VISTAS PARA REPORTES*

*-- =============================================*

*-- Vista de ventas por cliente*

CREATE VIEW vista\_ventas\_clientes AS SELECT

c.id\_cliente,

CONCAT(c.nombre, ' ', c.apellido) as cliente\_completo, c.email,

c.telefono,

COUNT(p.id\_pedido) as total\_pedidos, SUM(p.total) as total\_ventas, MAX(p.fecha\_pedido) as ultimo\_pedido, AVG(p.total) as ticket\_promedio

FROM CLIENTE c

LEFT JOIN PEDIDO p ON c.id\_cliente = p.id\_cliente AND p.estado NOT IN ('cancelado')

GROUP BY c.id\_cliente;

*-- Vista de productos más vendidos*

CREATE VIEW vista\_productos\_populares AS SELECT

p.id\_producto, p.codigo\_sku, p.nombre as producto,

cat.nombre as categoria, SUM(dp.cantidad) as total\_vendido,

SUM(dp.subtotal\_linea) as ingresos\_totales, p.stock\_actual

FROM PRODUCTO p

JOIN CATEGORIA\_PRODUCTO cat ON p.id\_categoria = cat.id\_categoria LEFT JOIN DETALLE\_PEDIDO dp ON p.id\_producto = dp.id\_producto

LEFT JOIN PEDIDO ped ON dp.id\_pedido = ped.id\_pedido AND ped.estado NOT IN ('cancelado') GROUP BY p.id\_producto

ORDER BY total\_vendido DESC;

*-- Vista de inventario crítico*

CREATE VIEW vista\_inventario\_critico AS SELECT

p.id\_producto, p.codigo\_sku, p.nombre,

cat.nombre as categoria, p.stock\_actual, p.stock\_minimo,

(p.stock\_actual - p.stock\_minimo) as diferencia, CASE

WHEN p.stock\_actual <= p.stock\_minimo THEN 'CRÍTICO' WHEN p.stock\_actual <= (p.stock\_minimo \* 2) THEN 'BAJO' ELSE 'NORMAL'

END as estado\_stock FROM PRODUCTO p

JOIN CATEGORIA\_PRODUCTO cat ON p.id\_categoria = cat.id\_categoria WHERE p.activo = 1

ORDER BY diferencia ASC;

*-- =============================================*

*-- TRIGGERS PARA AUTOMATIZACIÓN*

*-- =============================================*

*-- Trigger para actualizar stock al agregar detalle de pedido*

## DELIMITER *//*

CREATE TRIGGER tr\_after\_insert\_detalle\_pedido AFTER INSERT ON DETALLE\_PEDIDO

## FOR EACH ROW BEGIN

*-- Actualizar stock del producto*

## UPDATE PRODUCTO

SET stock\_actual = stock\_actual - NEW.cantidad, fecha\_actualizacion = CURRENT\_TIMESTAMP

WHERE id\_producto = NEW.id\_producto;

*-- Registrar movimiento en log de stock*

INSERT INTO LOG\_STOCK\_PRODUCTO (id\_producto, stock\_anterior, stock\_nuevo, diferencia, tipo\_movimiento, motivo)

## SELECT

NEW.id\_producto, stock\_actual + NEW.cantidad, stock\_actual,

-NEW.cantidad, 'salida',

CONCAT('Venta - Pedido #', NEW.id\_pedido) FROM PRODUCTO

WHERE id\_producto = NEW.id\_producto; END*//*

## DELIMITER ;

*-- Trigger para auditoría de cambios en pedidos*

## DELIMITER *//*

CREATE TRIGGER tr\_auditoria\_pedidos AFTER UPDATE ON PEDIDO

## FOR EACH ROW BEGIN

IF OLD.estado != NEW.estado THEN

INSERT INTO AUDITORIA\_PEDIDOS (tabla\_afectada, accion, id\_registro, datos\_anteriores, datos\_nuevos, usuario)

VALUES (

'PEDIDO',

'UPDATE',

NEW.id\_pedido,

JSON\_OBJECT('estado', OLD.estado, 'total', OLD.total), JSON\_OBJECT('estado', NEW.estado, 'total', NEW.total), CURRENT\_USER()

## ); END IF;

END*//* DELIMITER ;

*-- Trigger para actualizar total del pedido*

## DELIMITER *//*

CREATE TRIGGER tr\_actualizar\_total\_pedido AFTER INSERT ON DETALLE\_PEDIDO

FOR EACH ROW

BEGIN

UPDATE PEDIDO

SET subtotal = (

SELECT SUM(subtotal\_linea) FROM DETALLE\_PEDIDO

WHERE id\_pedido = NEW.id\_pedido

),

impuestos = (

SELECT SUM(subtotal\_linea) \* 0.20 FROM DETALLE\_PEDIDO

WHERE id\_pedido = NEW.id\_pedido

),

total = (

SELECT SUM(subtotal\_linea) \* 1.20 FROM DETALLE\_PEDIDO

WHERE id\_pedido = NEW.id\_pedido

)

WHERE id\_pedido = NEW.id\_pedido; END*//*

## DELIMITER ;

*-- =============================================*

*-- PROCEDIMIENTOS ALMACENADOS*

*-- =============================================*

*-- Procedimiento para crear nuevo pedido*

## DELIMITER *//*

CREATE PROCEDURE sp\_crear\_pedido( IN p\_id\_cliente INT,

IN p\_metodo\_pago VARCHAR(20), IN p\_notas TEXT

## ) BEGIN

DECLARE v\_numero\_pedido VARCHAR(20); DECLARE v\_id\_pedido INT;

*-- Generar número de pedido único*

SET v\_numero\_pedido = CONCAT('PED-', DATE\_FORMAT(NOW(), '%Y-%m-%d-'), LPAD(FLOOR(RAND() \* 10000), 4, '0'));

*-- Insertar pedido*

INSERT INTO PEDIDO (numero\_pedido, id\_cliente, metodo\_pago, notas) VALUES (v\_numero\_pedido, p\_id\_cliente, p\_metodo\_pago, p\_notas);

*-- Obtener ID del pedido insertado*

SET v\_id\_pedido = LAST\_INSERT\_ID();

*-- Devolver información del pedido creado*

## SELECT

v\_id\_pedido as id\_pedido, v\_numero\_pedido as numero\_pedido, 'Pedido creado exitosamente' as mensaje;

## END*//* DELIMITER ;

*-- Procedimiento para agregar producto a pedido*

## DELIMITER *//*

CREATE PROCEDURE sp\_agregar\_producto\_pedido( IN p\_id\_pedido INT,

IN p\_id\_producto INT, IN p\_cantidad INT

## ) BEGIN

DECLARE v\_precio DECIMAL(10,2);

DECLARE v\_stock\_actual INT;

*-- Obtener precio actual y stock del producto*

SELECT precio\_actual, stock\_actual INTO v\_precio, v\_stock\_actual FROM PRODUCTO

WHERE id\_producto = p\_id\_producto AND activo = 1;

*-- Verificar stock disponible*

IF v\_stock\_actual >= p\_cantidad THEN

*-- Insertar detalle del pedido*

INSERT INTO DETALLE\_PEDIDO (id\_pedido, id\_producto, cantidad, precio\_unitario) VALUES (p\_id\_pedido, p\_id\_producto, p\_cantidad, v\_precio);

SELECT 'Producto agregado al pedido exitosamente' as mensaje;

## ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Stock insuficiente para este producto';

## END IF;

END*//* DELIMITER ;

*-- =============================================*

*-- CONSULTAS DE PRUEBA*

*-- =============================================*

*-- Consultar ventas por cliente*

SELECT \* FROM vista\_ventas\_clientes;

*-- Consultar productos más vendidos*

SELECT \* FROM vista\_productos\_populares;

*-- Consultar inventario crítico*

SELECT \* FROM vista\_inventario\_critico WHERE estado\_stock IN ('CRÍTICO', 'BAJO');

*-- Ver logs de auditoría*

## SELECT \* FROM AUDITORIA\_PEDIDOS

ORDER BY fecha\_auditoria DESC LIMIT 10;

*-- =============================================*

*-- USUARIOS Y PERMISOS (EJEMPLO)*

*-- =============================================*

*-- Crear usuario para aplicación*

CREATE USER 'app\_ventas'@'localhost' IDENTIFIED BY 'PasswordSeguro123!'; GRANT SELECT, INSERT, UPDATE ON sistema\_ventas.\* TO 'app\_ventas'@'localhost';

*-- Crear usuario para reportes (solo lectura)*

CREATE USER 'reportes\_ventas'@'localhost' IDENTIFIED BY 'Reportes123!'; GRANT SELECT ON sistema\_ventas.\* TO 'reportes\_ventas'@'localhost';

*-- Crear usuario administrador*

CREATE USER 'admin\_ventas'@'localhost' IDENTIFIED BY 'AdminSeguro456!'; GRANT ALL PRIVILEGES ON sistema\_ventas.\* TO 'admin\_ventas'@'localhost';

*-- =============================================*

*-- MENSAJE FINAL*

*-- =============================================*

SELECT 'Base de datos sistema\_ventas implementada exitosamente' as estado;

# Características Implementadas:

✅ Estructura Completa

* Tablas principales normalizadas
* Claves y relaciones bien definidas
* Índices de optimización

✅ Datos de Prueba

* Clientes, productos, categorías
* Pedidos y detalles de ejemplo

✅ Automatizaciones

* Triggers para stock y auditoría
* Actualización automática de totales

✅ Vistas de Reportes

* Ventas por cliente
* Productos populares
* Inventario crítico

✅ Procedimientos Almacenados

* Creación de pedidos
* Agregar productos a pedidos

✅ Seguridad

* Ejemplos de usuarios y permisos

# BIBLIOGRAFÍA

* "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/