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

2.1. Dependencia del SGBD

- Sintaxis específica del motor elegido
- Tipos de datos particulares
- Características de almacenamiento
- Mecanismos de seguridad

2.2. 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)

2.3. 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;
```

```sql
-- 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;
```

```sql
-- 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', '+1234567890', 'Calle Principal 123, Ciudad'),
('Carlos', 'Rodriguez', 'carlos.rodriguez@email.com', '+1234567891', 'Avenida Central 456,
Ciudad'),
('Ana', 'Martinez', 'ana.martinez@email.com', '+1234567892', 'Plaza Mayor 789, Ciudad'),
('Luis', 'Hernandez', 'luis.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);
```

```sql
-- 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/