ALMACENAMIENTO Y MODELIZACIÓN DE DATOS

TEMA 2. BASES DE DATOS RELACIONALES

GABRIELA ZÚÑIGA ZÁRATE

TEMA 2. BASES DE DATOS RELACIONALES

Resultados específicos del aprendizaje

El estudiante será capaz de:

- Explicar los principios fundamentales del modelo relacional y su importancia en el diseño de bases de datos.
- Identificar los elementos básicos de una base de datos relacional, incluyendo tablas, atributos y relaciones.
- Definir y aplicar claves primarias y claves foráneas para garantizar la integridad referencial de los datos.
- Diseñar estructuras de tablas relacionales coherentes a partir de requisitos de información sencillos.
- Utilizar el lenguaje SQL básico para realizar consultas de selección (SELECT) sobre una base de datos.
- Aplicar sentencias SQL de inserción, actualización y eliminación de datos (INSERT, UPDATE, DELETE).
- Interpretar los resultados de las consultas SQL para validar la correcta estructura y contenido de la base de datos.

TEMA 2. BASES DE DATOS RELACIONALES

Contenido

2.1 Modelo relacional

2.2 Tablas, claves primarias y foráneas

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.1 Modelo relacional

2.1.1 Origen del modelo relacional

Antes de la década de 1970, la mayoría de las bases de datos utilizaban modelos jerárquicos o en red, en los que los datos estaban organizados mediante estructuras complejas y los usuarios debían conocer cómo estaban almacenados físicamente para poder acceder a ellos. Este enfoque dificultaba el mantenimiento de las aplicaciones y limitaba la flexibilidad de los sistemas.

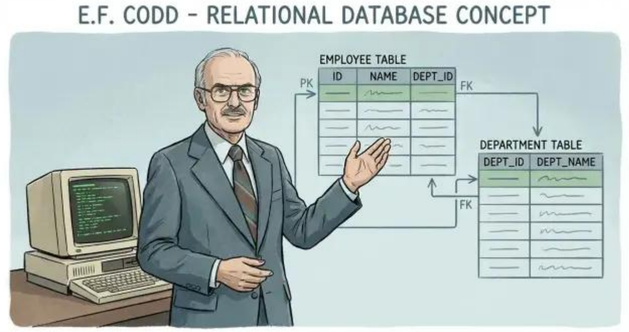
En 1970, Edgar F. Codd, investigador de IBM, publicó el artículo A Relational Model of Data for Large Shared Data Banks, considerado el punto de partida de las bases de datos relacionales modernas. En este trabajo propuso un nuevo enfoque para almacenar y gestionar la información basado en conceptos matemáticos de la teoría de conjuntos y del álgebra relacional.

Su propuesta revolucionó la gestión de datos porque permitió que los usuarios realizaran consultas sin necesidad de conocer la estructura física donde estaban almacenados los datos.

2.1 Modelo relacional

2.1.2 Edgar F. Codd y las bases de datos relacionales

Edgar Frank Codd (1923-2003) fue un matemático y científico informático británico que trabajó en IBM y es reconocido como el creador del modelo relacional, base teórica de los actuales Sistemas Gestores de Bases de Datos Relacionales (SGBDR) como MySQL, PostgreSQL, Oracle o SQL Server.



Las principales aportaciones de **Codd** fueron:

- Introducir el concepto de relación como forma de organizar los datos.
- Proponer la independencia entre los datos y las aplicaciones.
- Definir un conjunto de operaciones para manipular información.
- Sentar las bases para el desarrollo posterior del lenguaje SQL (Structured Query Language).

Gracias a estas contribuciones, Codd recibió en 1981 el Premio Turing, considerado el equivalente al Premio Nobel en informática.

2.1 Modelo relacional

2.1.3 Organización de los datos mediante relaciones (tablas)

En el modelo relacional, los datos se almacenan en relaciones, que habitualmente se representan como tablas.

Una tabla está formada por:

- Filas (tuplas o registros): representan una instancia concreta de información.
- Columnas (atributos): representan las características o propiedades de los datos.
- Valores: contenido almacenado en cada celda de la tabla.

2.1 Modelo relacional

2.1.4 Ejemplo

1. Ejemplo: Tabla Clientes

ID_Cliente	Nombre	Ciudad
1	Ana López	Barcelona
2	Carlos Ruiz	Madrid
3	Laura Pérez	Valencia

En este ejemplo:

- La tabla se denomina Clientes.
- Cada fila representa un cliente.
- Cada columna describe una característica del cliente.

2.1 Modelo relacional

2.1.5 Relaciones entre tablas

Una de las características más importantes del modelo relacional es la posibilidad de relacionar diferentes tablas.

Tabla Clientes

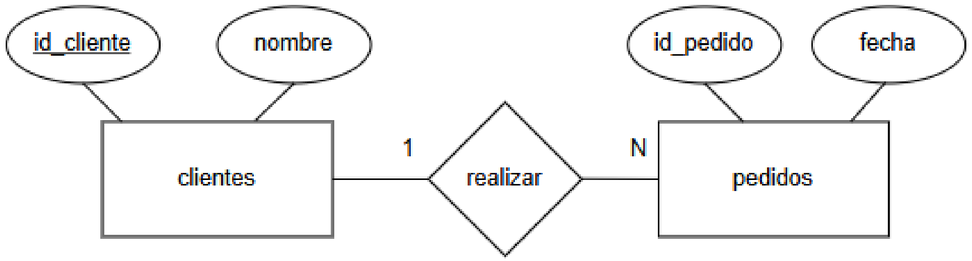
ID_Cliente	Nombre
1	Ana López
2	Carlos Ruiz

Tabla Pedidos

ID_Pedido	Fecha	ID_Cliente
101	2026-09-10	1
102	2026-09-15	2

La columna **ID_Cliente** de la tabla *Pedidos* permite asociar cada pedido con un cliente determinado.

Representación gráfica:



Representación conceptual simplificada de una relación entre las entidades CLIENTES y PEDIDOS utilizando la notación Entidad-Relación.

2.1 Modelo relacional

2.1.6 Objetivos del modelo relacional

El modelo relacional fue diseñado para resolver las limitaciones de los sistemas de datos existentes y perseguía varios objetivos fundamentales:

1. **Simplificar la gestión de datos:** Los datos se organizan mediante tablas fáciles de comprender y mantener.
2. **Garantizar la independencia de los datos:** Las aplicaciones pueden seguir funcionando, aunque cambie la forma física de almacenamiento de la información.
3. **Reducir la redundancia:** Se busca evitar la duplicación innecesaria de información, mejorando la consistencia de los datos.
4. **Garantizar la integridad:** Mediante claves y restricciones se asegura que los datos sean correctos y coherentes.
5. **Facilitar las consultas:** Los usuarios pueden recuperar información mediante consultas declarativas utilizando SQL.
6. **Favorecer la escalabilidad y el mantenimiento:** El modelo relacional permite ampliar las bases de datos y adaptarlas a nuevas necesidades de negocio sin rediseñar completamente el sistema

2.2 Tablas, claves primarias y foráneas

Una base de datos relacional es un sistema de almacenamiento que organiza la información en tablas relacionadas entre sí.

Cada tabla representa una entidad del negocio.

Ejemplos:

- Clientes
- Productos
- Empleados
- Pedidos

2.2 Tablas, claves primarias y foráneas

Ejemplo empresarial

Tabla Clientes

ClienteID	Nombre	Ciudad
1	Ana López	Barcelona
2	Carlos Pérez	Madrid
3	Marta Ruiz	Valencia

Tabla Pedidos

PedidoID	ClienteID	Fecha
1001	1	15/09/2026
1002	2	16/09/2026
1003	1	18/09/2026

Observación:

El cliente 1 ha realizado dos pedidos. La relación se establece mediante: ClienteID

2.2 Tablas, claves primarias y foráneas

2.2.1 Conceptos fundamentales: Tabla

Una tabla almacena información sobre una entidad.

Ejemplo:

Tabla CLIENTES

Propiedades de una tabla relacional

- Cada tabla tiene un nombre único.
- Cada columna tiene un nombre único dentro de la tabla.
- No existen filas duplicadas.
- Cada celda contiene un único valor (atomicidad).
- Todas las filas tienen la misma estructura.

2.2 Tablas, claves primarias y foráneas

2.2.2 Conceptos fundamentales: Registro (fila)

Cada fila representa una ocurrencia de la entidad.

ClienteID	Nombre
1	Ana López

Esta fila corresponde a un cliente concreto.

2.2 Tablas, claves primarias y foráneas

2.2.3 Conceptos fundamentales: Atributo (columna)

Describe una característica de la entidad.

Ejemplos:

- Nombre
- Dirección
- Teléfono
- Ciudad

2.2 Tablas, claves primarias y foráneas

2.2.4 Conceptos fundamentales: Dominio

Conjunto de valores permitidos para un atributo.

Ejemplos:

Edad:

- 0 - 120

Provincia:

- Barcelona
- Madrid
- Valencia

...

2.2 Tablas, claves primarias y foráneas

2.2.5 Conceptos fundamentales: Claves en bases de datos

Las claves permiten identificar registros y relacionar tablas.

Clave primaria (Primary Key)

Identifica de manera única cada registro.

Características

- No se repite.
- Nunca es nula.
- Es única.

Ejemplo:

ClienteID	Nombre
1	Ana
2	Carlos
3	Marta

ClienteID es clave primaria.

2.2 Tablas, claves primarias y foráneas

2.2.6 Conceptos fundamentales: Clave foránea (Foreign Key)

Permite conectar tablas.

Tabla clientes

ClienteID	Nombre
1	Ana
2	Carlos
3	Marta

Tabla pedidos

PedidoID	ClienteID
1001	1
1002	2

ClienteID es clave foránea.

2.2 Tablas, claves primarias y foráneas

2.2.7 Conceptos fundamentales: Relaciones entre tablas

Relación 1:1

Un registro se relaciona con uno y solo uno.

Ejemplo:

Persona ↔ Pasaporte

2.2 Tablas, claves primarias y foráneas

2.2.8 Conceptos fundamentales: Relaciones entre tablas

Relación 1:N

La más común.

Ejemplo:

Cliente → Pedidos

Un cliente puede realizar muchos pedidos.

Un pedido pertenece a un cliente.

2.2 Tablas, claves primarias y foráneas

2.2.9 Conceptos fundamentales: Relaciones entre tablas

Relación 1:N Relación de 1 a muchos

Tabla: Cliente

ClienteID	Nombre
1	Ana
2	Carlos
3	Marta



Tabla: Pedido

PedidoID	Descrip
1001	Pedido 1001
1002	Pedido 1002

Un cliente puede realizar muchos pedidos.

Un pedido pertenece a un cliente.

2.2 Tablas, claves primarias y foráneas

2.2.9 Conceptos fundamentales: Relaciones entre tablas

Relación N:M. Relación de Muchos a muchos.

Ejemplo:

Tabla: Pedidos

PedidoID	Descrip
1001	Pedido 1001
1002	Pedido 1002

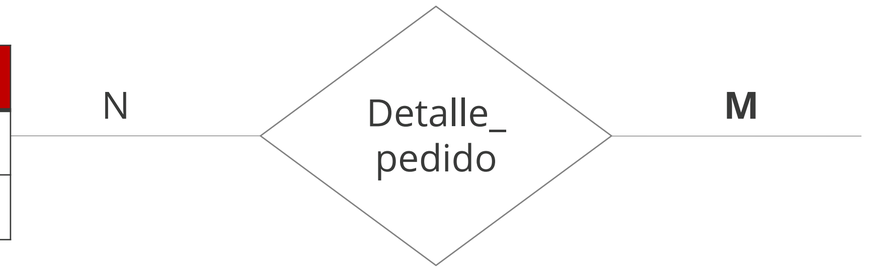


Tabla: Productos

productoID	NombreProduc
1	Ordenador
2	Teclado
3	móvil

Un pedido puede contener varios productos.

Un producto puede aparecer en varios pedidos.

2.2 Tablas, claves primarias y foráneas

2.2.10 Restricciones de integridad

Integridad de entidad

La clave primaria identifica cada registro.

No puede repetirse.

No puede ser nula.

Integridad referencial

La clave foránea debe apuntar a una clave primaria existente.

Evita registros huérfanos.

PK Unica

PedidoID = 1001

PedidoID = 1002

Esto sería incorrecto

PedidoID = 1001

PedidoID = 1001

FK Válida

ClienteID = 1

Incorrecto

ClienteID=999

(si el cliente 999 no existe)

2.2 Tablas, claves primarias y foráneas

2.2.10 Restricciones de integridad

Restricciones de dominio

Edad BETWEEN 0 AND 120

Restricción UNIQUE

Email UNIQUE

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.1 ¿Qué es SQL?

SQL (Structured Query Language) es el lenguaje estándar utilizado para interactuar con bases de datos relacionales.

Mediante SQL es posible:

- Consultar información.
- Insertar nuevos registros.
- Modificar datos existentes.
- Eliminar registros.
- Crear y administrar tablas.

Gestores de bases de datos que utilizan SQL

- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- MariaDB

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

SQL se divide en varios grupos de instrucciones

DDL (Data Definition Language): Permite definir la estructura de la base de datos. Los principales comandos son:

- CREATE
- ALTER
- DROP

DML (Data Manipulation Language): Permite trabajar con los datos almacenados. Los principales comandos son:

- INSERT
- UPDATE
- DELETE
- SELECT

En este tema vamos a trabajar fundamentalmente con dos grupos de instrucciones: DDL para crear tablas y DML para insertar, modificar y consultar datos.

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.2 Tipos de datos en SQL

Al crear una tabla es necesario indicar qué tipo de información almacenará cada columna.

Tipos numéricos

INT: Números enteros.

Ejemplos: 1, 2, 225, 3, 1000

DECIMAL(10,2):Números con decimales.

Ejemplos: 25.50 ,199.99

Tipos de texto

VARCHAR(100): Cadenas de texto de longitud variable.

Ejemplos: Ana López, Barcelona, etc.

CHAR(2): Texto de longitud fija.

Ejemplos: ES, FR, IT

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.2 Tipos de datos en SQL

Tipos de fecha

DATE: Almacena fechas.

Ejemplo: 2026-09-15

Tipo lógico

BOOLEAN

Valores: TRUE o FALSE

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.3 CREATE TABLE

CREATE TABLE Clientes (ClienteID INT PRIMARY KEY, Nombre VARCHAR(100),Ciudad VARCHAR(50));

La instrucción CREATE TABLE crea una nueva tabla llamada Clientes en la base de datos.

La tabla tendrá dos columnas:

Columna	Tipo de dato	Descripción
ClienteID	INT	Identificador numérico del cliente
Nombre	VARCHAR(100)	Nombre del cliente con hasta 100 caracteres

Explicación

CREATE TABLE crea una tabla.

ClienteID identifica a cada cliente.

PRIMARY KEY define la clave primaria.

VARCHAR almacena texto.

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.4 Inserción de datos (INSERT)

Permite añadir nuevos registros a una tabla.

Ejemplo

INSERT INTO Clientes

VALUES (1, 'Ana López', 'Barcelona');

Varias inserciones

INSERT INTO Clientes VALUES (2, 'Carlos Pérez', 'Madrid');

INSERT INTO Clientes VALUES (3, 'Marta Ruiz', 'Valencia');

ClienteID	Nombre	Ciudad
1	Ana López	Barcelona
2	Carlos Pérez	Madrid
3	Marta Ruiz	Valencia

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.5 Consulta de datos (SELECT)

Es la operación más utilizada en SQL.

Mostrar todas las

SELECT * FROM Clientes;

SELECT Nombre, Ciudad FROM Clientes;

Resultado:

Nombre	Ciudad
Ana López	Barcelona
Carlos Pérez	Madrid
Marta Ruiz	Valencia

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.6 Filtrado de datos (WHERE)

Permite recuperar únicamente los registros que cumplen una condición.

Ejemplo

SQL

SELECT * FROM Clientes WHERE Ciudad = 'Barcelona';

Resultado:

ClienteID	Nombre	Ciudad
1	Ana López	Barcelona

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.6 Filtrado de datos (WHERE).

Operadores frecuentes

Operador	Significado
=	Igual
<>	Distinto
>	Mayor
	Menor
>=	Mayor o igual
<=	Menor o igual

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.7 Ordenación de resultados (ORDER BY)

Permite ordenar la información obtenida.

Orden ascendente

SELECT * FROM Clientes ORDER BY Nombre;

Orden descendente

SELECT * FROM Clientes ORDER BY Nombre DESC;

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.8 Actualización de datos (UPDATE)

Permite modificar registros ya existentes.

Ejemplo

UPDATE Clientes SET Ciudad = 'Sevilla' WHERE ClienteID = 2;

Resultado

ClienteID	Nombre	Ciudad
1	Ana López	Barcelona
2	Carlos Pérez	Sevilla
3	Marta Ruiz	Valencia

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.9 Eliminación de datos (DELETE)

Permite borrar registros de una tabla.

Ejemplo

DELETE FROM Clientes WHERE ClienteID = 3;

Resultado:

ClienteID	Nombre	Ciudad
1	Ana López	Barcelona
2	Carlos Pérez	Sevilla

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.10 Consultas sobre varias tablas (JOIN)

Una de las ventajas del modelo relacional es combinar información procedente de varias tablas.

Tabla Clientes

ClienteID	Nombre
1	Ana López
2	Carlos Pérez

Tabla Pedidos

PedidoID	ClienteID	Importe
1001	1	250
1002	2	120
1003	1	80

Consulta

```
SELECT c.Nombre, p.PedidoID, p.Importe FROM Clientes c
INNER JOIN Pedidos p
ON c.ClienteID = p.ClienteID;
```

Nombre	PedidoID	Importe
Ana López	1001	250
Carlos Pérez	1002	120
Ana López	1003	80

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.11 Caso práctico completo

Creación de tablas

```
CREATE TABLE Clientes (
ClienteID INT PRIMARY KEY,
Nombre VARCHAR(100),
Ciudad VARCHAR(50)
);
CREATE TABLE Pedidos (
PedidoID INT PRIMARY KEY,
Fecha DATE,
Importe DECIMAL(10,2),
ClienteID INT,
FOREIGN KEY (ClienteID)
REFERENCES Clientes(ClienteID)
);
```

Inserción de datos

```
INSERT INTO Clientes
VALUES (1,'Ana López','Barcelona');
INSERT INTO Pedidos
VALUES (1001,'2026-09-15',250.00,1);
```

Selección de datos

```
SELECT c.Nombre, p.Importe
FROM Clientes c
INNER JOIN Pedidos p
ON c.ClienteID = p.ClienteID;
```

2.3 Lenguaje SQL básico (consultas, inserciones, actualizaciones)

2.3.12 Caso práctico completo

Actualización de datos

```
UPDATE Clientes
SET Ciudad='Girona'
WHERE ClienteID=1;
```

Eliminación de datos

```
DELETE FROM Pedidos
WHERE PedidoID=1001;
```