

TEMA 40

DISEÑO DE BASES DE DATOS RELACIONALES

INDICE:

1.	INTRODUCCIÓN	1
2.	FASE DE DISEÑO	1
3.	DISEÑO CONCEPTUAL: MODELO ENTIDAD-RELACIÓN.....	2
3.1.	Entidades	2
3.2.	Relaciones	2
3.3.	Cardinalidad.....	3
3.4.	Atributos.....	4
4.	DISEÑO LÓGICO: MODELO RELACIONAL.....	5
4.1.	Conversión del modelo E/R al modelo relacional.....	5
4.2.	Normalización	6
5.	DISEÑO FÍSICO.....	8
6.	CONCLUSIÓN	8
7.	BIBLIOGRAFÍA	9
8.	NORMATIVA.....	9

Realizado por Cayetano Borja Carrillo

Tiempo de escritura: 2 horas

1. INTRODUCCIÓN

Una base de datos es un conjunto de datos organizados que pueden ser manipulados y consultados cuando sea necesario. Por ejemplo, el inventario de una biblioteca se considera una base de datos ya que se trata de un documento (en papel o digital) que contiene información organizada que puede ser consultada.

Existen varios tipos de base de datos, siendo las relacionales las más habituales. Este tipo de base de datos se caracterizan porque la información se organiza en objetos que se relacionan entre ellos mediante identificadores.

En este tema se desarrollan los pasos que hay que seguir para diseñar una base de datos relacional coherente y eficaz. Se trata de un tema de gran importancia en la informática, ya que una gran cantidad de *software* utiliza las bases de datos relacionales para almacenar la información.

2. FASE DE DISEÑO

La creación de un base de datos es un proceso que parte de la necesidad de almacenar información del mundo real para que su acceso sea rápido y eficiente.

Un buen diseño de una base de datos nos va a dar una mayor velocidad de acceso a los datos, una mayor protección de incoherencias y, en general, un mejor aprovechamiento de los recursos. Por el contrario, un diseño inapropiado puede dar lugar a los siguientes problemas:

- Pérdida de dependencias funcionales: Las dependencias funcionales garantizan la integridad de los datos.
- Redundancias: Consiste en la repetición innecesaria de información, que no solo incrementará el espacio ocupado por la base de datos, sino que en el futuro puede dar lugar a incoherencias.
- Incoherencias: Una incoherencia ocurre cuando ciertos datos de la base de datos contradicen otros datos. Por ejemplo, es posible tener 2 registros con la información de un mismo cliente donde los campos “edad” no coinciden.

Para evitar estos problemas, es necesario seguir una metodología de 3 pasos a la hora de diseñar una base de datos.

- Paso 1: Diseño conceptual: Consiste en estudiar el problema y seleccionar qué elementos del mundo real se van a modelar. El modelo conceptual más extendido es el modelo entidad-relación (E/R).
- Paso 2: Diseño lógico: Consiste en describir los datos del modelo conceptual con el mayor detalle posible para que puedan ser implementados físicamente. El modelo lógico más utilizado es el modelo relacional.

- **Paso 3: Diseño físico:** Consiste en implementar la base de datos en un soporte de almacenamiento secundario como, por ejemplo, un disco duro, para que posteriormente pueda ser manipulada por un programa llamado sistema gestor de base de datos (SGBD).

3. DISEÑO CONCEPTUAL: MODELO ENTIDAD-RELACIÓN

El modelo E/R es un modelo de datos conceptual propuesto por Peter P. Chen en los años 70. Se basa en percibir la información del mundo real como un conjunto de objetos llamados entidades que se relacionan entre ellos.

A continuación, se describen los principales elementos del modelo E/R.

3.1. Entidades

Una entidad es un objeto o concepto del mundo real que interesa almacenar, por ejemplo, una persona o un libro. Existen 2 tipos de entidades:

- **Fuertes:** Su existencia no depende de ninguna otra entidad. Se representa mediante un rectángulo. Un ejemplo de entidad fuerte es “Persona”.

Persona

- **Débiles:** La entidad depende de otra entidad. Esta dependencia puede darse por identificación cuando necesita de otra entidad para identificarse o por existencia si depende totalmente de la otra entidad para existir. Por ejemplo, si se quiere almacenar la información de las habitaciones de una cadena de hoteles, conocer el número de habitación no es suficiente para identificar la habitación, sino que también, se necesita saber a qué hotel pertenece dicha habitación. Se representa mediante un rectángulo de contorno doble.

Habitación

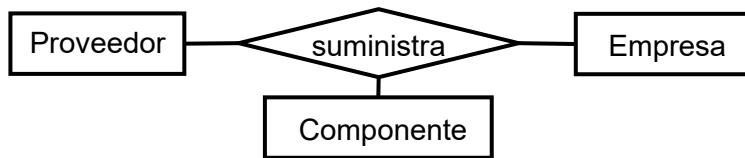
3.2. Relaciones

Las relaciones representan vínculos entre entidades. Se representan mediante un rombo unido por líneas con las entidades que tienen relación. Por ejemplo, las entidades “Hotel” y “Habitación” tienen una relación ya que un hotel tiene varias habitaciones y una habitación pertenece a un hotel.

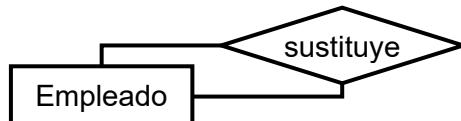


Además de las relaciones binarias (entre 2 entidades), existen los siguientes tipos de relaciones:

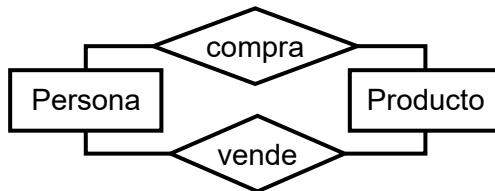
- N-arias: Relación entre “n” entidades. Si es entre 3, se llama ternarias. Ejemplo:



- Reflexiva: Se llama relación reflexiva cuando una entidad tiene una relación consigo misma. Por ejemplo, un empleado sustituye y puede ser sustituido por otro empleado.



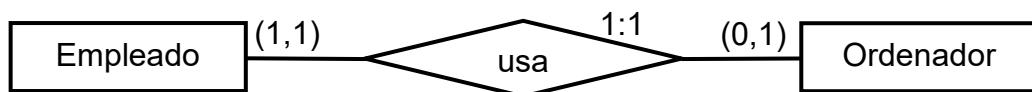
- Dobles: Se produce cuando existen 2 relaciones entre 2 entidades. Ejemplo:



3.3. Cardinalidad

La cardinalidad de las relaciones indica la cantidad de instancias de una entidad que se pueden asociar con una instancia de otra entidad. Existen 3 casos de cardinalidad:

- Uno a uno (1:1): Una instancia de una entidad tiene relación con solo una instancia de otra entidad y lo mismo ocurre al contrario. Ejemplo: En una empresa, cada empleado puede tener asignado un ordenador y cada ordenador es utilizado únicamente por el empleado al que le fue asignado.



- Uno a muchos (1:N): Una instancia de una entidad puede tener relación con varias instancias de otra entidad, pero no al contrario. Ejemplo: Una madre puede tener varios hijos, pero un hijo solo tiene una madre.



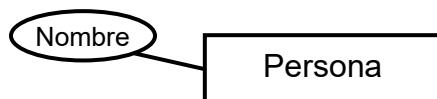
- Muchos a muchos (N:M): Una instancia de una entidad puede tener relación con varias instancias de otra entidad y viceversa. Ejemplo: Una persona puede practicar varios deportes y un deporte puede ser practicado por varias personas.



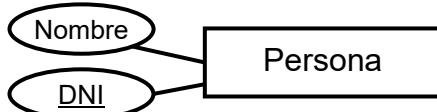
3.4. Atributos

Los atributos son las propiedades que interesan almacenar de las entidades o de las relaciones. Por ejemplo, los atributos de la entidad “Persona” pueden ser el DNI, el nombre, la edad, etc. Se distinguen los siguientes tipos de atributos:

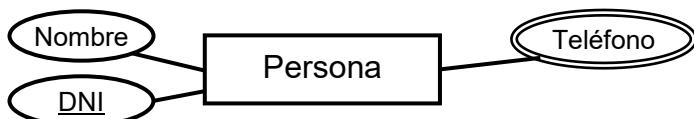
- **Atributos normales:** Son los datos que interesan almacenar de una entidad o relación. Una forma de representarlo es rodeando con un círculo el nombre del atributo unido con una línea a la entidad o relación. Ejemplo:



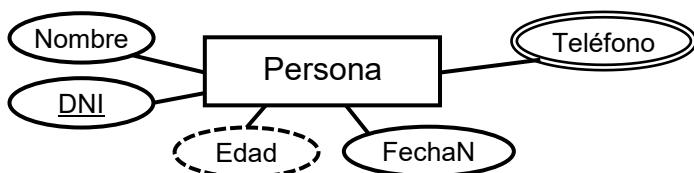
- **Clave primaria:** Es un atributo o conjunto de atributos que identifican de forma inequívoca a un registro. No pueden existir 2 registros con la misma clave, por ejemplo, 2 personas con el mismo DNI. Se puede representar subrayando el nombre del atributo. Ejemplo:



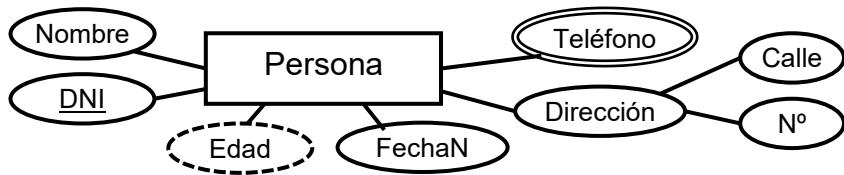
- **Atributo multivaluado:** Es un atributo que puede tener múltiples valores, por ejemplo, una persona puede tener varios números de teléfono. Se representa con un círculo de doble contorno. Ejemplo:



- **Atributo derivado:** Es un atributo cuyo valor puede derivarse de los valores de otros atributos. Por ejemplo, “edad” puede derivarse del atributo “fecha de nacimiento”. Se representa con un círculo punteado.



- **Atributo compuesto:** Es un atributo que puede ser descompuesto en varios atributos simples. Por ejemplo, el atributo “dirección” puede descomponerse en los atributos “número”, “calle”, “portal”, etc. Ejemplo:



4. DISEÑO LÓGICO: MODELO RELACIONAL

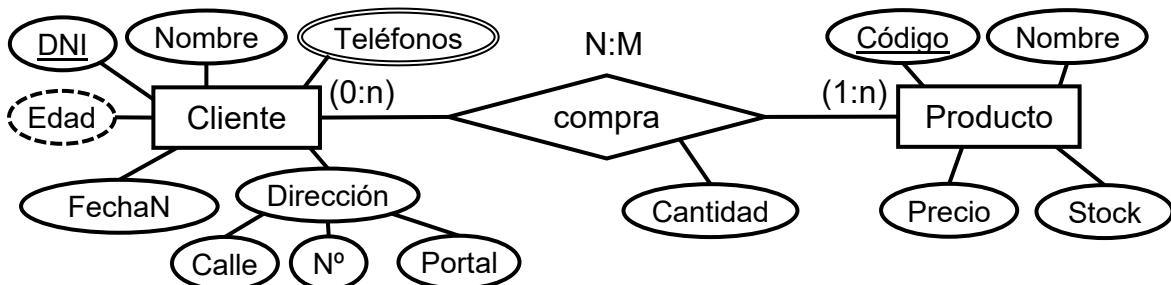
El modelo relacional se basa en la lógica de los predicados para establecer relaciones entre distintos datos. Se implementa partiendo de un diseño conceptual previo como, por ejemplo, un modelo E/R.

4.1. Conversión del modelo E/R al modelo relacional

Para convertir un modelo E/R al modelo relacional, hay que tener en cuenta los siguientes aspectos:

- Todas las entidades del modelo E/R se transforman en relaciones (tablas).
- En las relaciones 1:N, la clave primaria de la entidad con cardinalidad “1” pasa a la relación de la entidad con cardinalidad “N” como clave ajena o foránea.
- Cuando la relación es N:M, se crea una nueva tabla donde se incluirán las claves de las entidades que relaciona más los atributos de la relación. El conjunto de claves ajenas forma la clave principal.
- Si la relación es 1:1, pueden darse 3 casos:
 - o Si la cardinalidad es (0,1) en ambos extremos, se crea una tabla nueva donde se incluirán las claves de las 2 entidades como ajenas. La clave primaria será una (cualquiera) de ellas.
 - o Si la cardinalidad es (0,1) en un extremo y (1,1) en el otro, la clave de la entidad con cardinalidad (1,1) pasa a la otra entidad como ajena.
 - o Si la cardinalidad es (1,1) en ambos extremos, normalmente se pasa la clave principal de una entidad cualquiera a la otra entidad como clave ajena, aunque si se desea, se pueden pasar las claves primarias de ambas entidades como claves ajenas en las 2 tablas.
- Si existen atributos multivaluados, se crea una tabla cuya clave principal será el atributo multivaluado y la clave de la entidad a la que se refiere.
- Los atributos derivados no se representan en el modelo relacional, evitando así que se produzcan incoherencias.
- Los atributos compuestos no se reflejan en las tablas, solo se ponen los atributos simples que contiene.

Teniendo en cuenta esos puntos, podemos transformar el siguiente modelo E/R al siguiente modelo relacional:



Teléfono (DNI, Teléfono)
 ↓
 Cliente (DNI, Nombre, FechaN, Calle, Nº, Portal)
 ↑
 Compra (DNI, Código, Cantidad)
 ↓
 Producto (Código, Nombre, Precio, Stock)

4.2. Normalización

La normalización consiste en aplicar una serie de reglas a las relaciones para eliminar la redundancia de datos y proteger la integridad de los datos. Es muy importante comprobar si el modelo relacional obtenido está normalizado antes de llevarlo al diseño físico. Existen las siguientes formas normales:

Primera Forma Normal (1FN)

Una relación está en 1FN si no tiene grupos repetitivos, es decir, si sus atributos no tienen más de un valor por cada registro. Por ejemplo: Tenemos la siguiente relación sobre un cliente:

Cliente (DNI, Nombre, CodProducto, Descripción, Precio, Cantidad)

Si se observa la tabla “Cliente”, se puede apreciar que para un registro con un DNI concreto existe un único campo “Nombre”, pero puede haber varios campos “CodProducto”, “Descripción”, “Precio” y “Cantidad”. Ejemplo:

<u>DNI</u>	Nombre	Cod_Producto	Descripción	Precio	Cantidad
11111111X	Cliente1	003; 004	Tomate; Pera	1,2; 1,5	3; 2
22222222Y	Cliente2	003	Tomate	1,2	2

Para pasarlo a 1FN, se crea una nueva relación cuyos campos son los atributos multivaluados, la clave principal de la tabla original y la clave de la relación que no cumple la 1FN.

Cliente (DNI, Nombre)

Compra (DNI, CodProducto, Descripción, Precio, Cantidad)

Segunda Forma Normal (2FN)

Una relación está en 2FN si está en 1FN y, además, no tiene dependencias parciales. Una dependencia parcial se produce cuando un atributo depende funcionalmente de una parte de la clave y no de la clave completa.

Si se observa la relación “Compra” del ejemplo anterior, los atributos “Descripción” y “Precio” sólo dependen funcionalmente de la clave “CodProducto” y no de la clave completa. Para pasarlo a 2FN, se crea una tabla nueva con su clave y con los atributos que no dependen funcionalmente de la clave completa. Ejemplo:

Cliente (DNI, Nombre)

Compra (DNI, CodProducto, Cantidad)

Producto (CodProducto, Descripción, Precio)

Tercera Forma Normal (3FN)

Una relación está en 3FN si está en 2FN y, además, no tiene dependencias funcionales transitivas entre la clave primaria y sus atributos en las que no participe una clave candidata.

Una dependencia funcional “ $X \rightarrow Y$ ” es una dependencia funcional transitiva si existe un atributo “Z” donde existe una dependencia funcional “ $X \rightarrow Z$ ” y “ $Z \rightarrow Y$ ”. Para solucionar esta dependencia, se crea una relación nueva con los atributos “Y” y “Z”, siendo “Z” la clave.

Ejemplo: Tenemos la siguiente tabla de la cuenta corriente de un banco

Cuenta (NumCuenta, DNICliente, NombreCliente)

Como se puede ver, existe una dependencia funcional transitiva entre “NumCuenta” y “NombreCliente” ya que hay una dependencia funcional “ $\text{NumCuenta} \rightarrow \text{DNICliente}$ ” y otra “ $\text{DNICliente} \rightarrow \text{NombreCliente}$ ”. Se pasa a 3FN:

Cuenta (NumCuenta, DNICliente)

Cliente (DNICliente, NombreCliente)

Forma Normal de Boyce-Codd (FNBC)

Las 3 primeras formas normales fueron descritas por Codd en 1970. Estas planteaban algunos problemas, hasta que 4 años después, Boyce y Codd redefinen la tercera, llamándose “Forma Normal de Boyce-Codd” o FNBC.

Una relación está en FNBC si cada determinante es una clave candidata.

Cuarta Forma Normal (4FN)

Se dice que una relación está en 4FN si se encuentra en FNBC y, además, no existen dependencias multivaluadas no triviales.

Quinta Forma Normal (5FN)

Una relación estar en 5FN si se encuentra en 4FN y, además, toda dependencia de combinación es consecuencia de las claves candidatas.

¿Es necesario llegar hasta la 5FN? No, lo natural suele ser llegar hasta la 3FN ya que muchos autores piensan que con FNBC se puede producir pérdidas de dependencias funcionales. Además, según avanzamos en el proceso, las relaciones se van fragmentando y, por tanto, se ralentiza el acceso a los datos.

5. DISEÑO FÍSICO

El diseño físico consiste en implementar la base de datos sobre un soporte de almacenamiento secundario, por ejemplo, un disco duro. La implementación se realiza mediante un lenguaje de definición de datos (DDL) para que pueda ser manipulado por un Sistema Gestor de Bases de Datos (SGBD).

Lo primero que se tiene que hacer en el diseño físico es crear la base de datos asignándole un nombre. Ejemplo: Crear una base de datos llamada “Tienda” en el lenguaje de definición de datos MySQL.

```
CREATE DATABASE Tienda;
```

Una vez creada, se selecciona la base de datos:

```
USE DATABASE Tienda;
```

Posteriormente, se crean las tablas definidas en el modelo relacional. Ejemplo: Crear la tabla “Persona” en MySQL:

```
CREATE TABLE Persona (
    DNI VARCHAR(9) NOT NULL,
    Nombre VARCHAR(20) NOT NULL,
    Direccion VARCHAR(100) NOT NULL,
    PRIMARY KEY (DNI)
);
```

Una vez creada la base de datos, se pueden introducir valores en ella y consultarlos mediante un lenguaje de manipulación de datos (DML).

6. CONCLUSIÓN

A la hora de diseñar una base de datos, existen distintas alternativas de una misma realidad, es decir, se pueden obtener distintos esquemas y no todos ellos son equivalentes, ya que unos van a representar el problema mejor que otros.

Construir una base de datos es un proceso de diseño creativo y requiere de una gran imaginación. Asegurarse de que todos los detalles están incluidos, de que las tablas

estén correctamente normalizadas para que no haya redundancias y de utilizar las optimizaciones que el nivel físico proporciona son necesarios para si se quiere obtener una base de datos con un buen diseño y rendimiento.

7. BIBLIOGRAFÍA

- Prieto Espinosa, A. et al. (2006). *Introducción a la informática* (4^a ed.). McGraw-Hill.
- Brookshear, J. G. (2012). *Introducción a la computación* (11^a ed.). Pearson Educación.
- Silberschatz, A. et al. (2014). *Fundamentos de bases de datos* (6^a ed.). Mc Graw-Hill.
- Elmasri, R. et al. (2007). *Fundamentos de Sistemas de Bases de Datos* (5^a ed.). Pearson Educación.

8. NORMATIVA

Para el desarrollo de este tema, se ha tenido en cuenta la siguiente normativa, donde se especifican los contenidos, competencias y criterios de evaluación de los Ciclos Formativos y Bachillerato en Andalucía:

- Orden 7 de julio de 2009 (SMR). La parte correspondiente al módulo “Aplicaciones ofimáticas”.
- Orden 19 de julio de 2010 (ASIR). La parte correspondiente al módulo “Gestión de bases de datos”.
- Orden 16 de junio de 2011 (DAW/DAM). La parte correspondiente al módulo “Bases de datos”.
- Instrucción 13/2022 (Bachillerato). La parte correspondiente a la asignatura “Tecnologías de la Información y Comunicación”