5. Introducción a las bases de datos relacionales

Un **modelo de datos** es un lenguaje utilizado para la descripción de una base de datos. Con este lenguaje vamos a poder describir las estructuras de los datos (tipos de datos y relaciones entre ellos), las restricciones de integridad (condiciones que deben cumplir los datos, según las necesidades de nuestro modelo basado en la realidad) y las operaciones de manipulación de los datos (insertado, borrado, modificación de datos).

5.1. Modelo de datos relacional

El **modelo relacional** se basa en el concepto matemático de relación, es un modelo de organización y gestión de bases de datos consistente en el almacenamiento de datos en tablas compuestas por filas o **tuplas** y columnas o **campos.**

El modelo de datos relacional persigue:

- **Independencia Física**: La forma de almacenar los datos no debe influir en su manipulación. Si el almacenamiento físico cambia, los usuarios que acceden a esos datos no tienen que modificar sus aplicaciones.
- **Independencia Lógica**: Las aplicaciones que utilizan la base de datos no deben ser modificadas por que se inserten, actualicen y eliminen datos.
- **Flexibilidad**: En el sentido de poder presentar a cada usuario los datos de la forma en que éste prefiera
- **Uniformidad**: Las estructuras lógicas de los datos siempre tienen una única forma conceptual (las tablas), lo que facilita la creación y manipulación de la base de datos por parte de los usuarios.
- **Sencillez**: Las características anteriores hacen que este Modelo sea fácil de comprender y de utilizar por parte del usuario final.

Atributos: es el nombre de cada dato que se almacena en la relación (tabla). Ejemplos serían: DNI, nombre, apellidos, etc.

El nombre del atributo debe describir el significado de la información que representa. Si tenemos la tabla *Empleados*, el atributo *Sueldo* almacenará el valor en euros del sueldo que recibe cada empleado. A veces es necesario añadir una pequeña descripción para aclarar un poco más el contenido. Por ejemplo, si el sueldo es neto o bruto.

Tuplas: se refiere a cada elemento de la relación. Si una tabla guarda datos de un cliente, como su DNI o Nombre, una tupla o registro sería ese DNI y nombre concreto de un cliente.

Cada una de las filas de la tabla se corresponde con la idea de registro y tiene que cumplir que:

- Cada tupla se debe corresponder con un elemento del mundo real.
- No puede haber dos tuplas iguales (con todos los valores iguales).

Está claro que un atributo en una tupla no puede tomar cualquier valor. No sería lógico que en un atributo *Población* se guarde "250€". Estaríamos cometiendo un error, para evitar este tipo de situaciones obligaremos a que cada atributo sólo pueda tomar los valores pertenecientes a un conjunto de valores previamente establecidos, es decir, un atributo tiene asociado un **dominio de valores**.

A menudo un dominio se define a través de la declaración de un tipo para el atributo (por ejemplo, diciendo que es un número entero entre 1 y 16), pero también se pueden definir dominios más complejos y precisos. Por ejemplo, para el atributo *Sexo* de mis empleados, podemos definir un dominio en el que los valores posibles sean "M" o "F" (masculino o femenino).

Una característica fundamental de los dominios es que sean atómicos, es decir, que los valores contenidos en los atributos no se pueden separar en valores de dominios más simples.

Un dominio debe tener: Nombre, Definición lógica, Tipo de datos y Formato.

Por ejemplo, si consideramos el *Sueldo* de un *Empleado*, tendremos:

Nombre: Sueldo.

Definición lógica: Sueldo neto del empleado

Tipo de datos: número entero.

Formato: 9.999€.

5.2. Grado. Cardinalidad

Ya hemos visto que una relación es una tabla con filas y columnas. Pero ¿hasta cuántas columnas puede contener? ¿Cuántos atributos podemos guardar en una tabla?

Llamaremos **grado** al tamaño de una tabla en base a su número de atributos (columnas). Mientras mayor sea el grado, mayor será su complejidad para trabajar con ella.

¿Y cuántas tuplas (filas o registros) puede tener?

Llamaremos **cardinalidad** al número de tuplas o filas de una relación o tabla.

Vamos a verlo con un ejemplo:

Relación de grado 3, sobre los dominios A={Carlos, María}, B={Matemáticas, Lengua}, C={Aprobado, Suspenso}.

Las posibles relaciones que obtenemos al realizar el producto cartesiano AxBxC (2*2*2=8) es el siguiente:

A={Carlos, María}	B={Matemáticas, Lengua}	C={Aprobado, Suspenso}
Carlos	Matemáticas	Aprobado
Carlos	Matemáticas	Suspenso
Carlos	Lengua	Aprobado
Carlos	Lengua	Suspenso
María	Matemáticas	Aprobado
María	Matemáticas	Suspenso
María	Lengua	Aprobado
María	Lengua	Suspenso

Si cogemos un subconjunto de 5 filas de esta tabla, diremos que tiene cardinalidad 5

A={Carlos, María}	B={Matemáticas, Lengua}	C={Aprobado, Suspenso}
Carlos	Matemáticas	Aprobado
Carlos	Matemáticas	Suspenso
Carlos	Lengua	Aprobado
Carlos	Lengua	Suspenso
María	Matemáticas	Aprobado

La cardinalidad de una relación depende del momento en que ésta sea considerada, pero el grado de una relación es independiente del tiempo. Dependiendo del grado de la relación éstas se denominan: *unarias, binarias, ternarias, etc..*

5.3. Sinónimos

Los términos vistos hasta ahora tienen distintos sinónimos según la nomenclatura utilizada. Trabajaremos con tres:

- En el modelo relacional: RELACIÓN - TUPLA - ATRIBUTO - GRADO - CARDINALIDAD.
- En tablas: TABLA - FILA - COLUMNAS - NÚMERO COLUMNAS - NÚMERO FILAS.
- En términos de registros: FICHEROS - REGISTROS - CAMPOS - NÚMERO CAMPOS - NÚMERO REGISTROS.

5.4. Características de una tabla

- Cada **tabla** tiene un nombre distinto

- Cada **atributo** (columna) de la tabla toma un solo valor en cada tupla (fila)

- Cada **atributo** (columna) tiene un nombre distinto en cada tabla (pero puede ser el mismo en tablas distintas).

- No puede haber dos tuplas (filas) completamente iguales.

Carlos	Matemáticas	Aprobado
Carlos	Matemáticas	Suspenso
Carlos	Lengua	Aprobado
Carlos	Lengua	Aprobado

- El orden de las tuplas (filas) no importa.

a	20
b	30
c	70

=

a	20
c	70
b	30

- El orden de los atributos (columnas) no importa.

a	20
b	30
c	70

=

20	a
70	c
30	b

- Todos los datos de un atributo (columna) deben ser del mismo dominio.

Carlos	Matemáticas	Aprobado
Carlos	Lengua	Suspenso
Carlos	Inglés	NOTABLE
María	Matemáticas	Suspenso
María	Lengua	Suspenso

Tipos de relaciones

Existen varios tipos de relaciones, vamos a clasificarlas en:

- **Persistentes**: permiten que sus registros sean borrados o eliminados manualmente por los usuarios
  - **Base**: Es donde se encuentran todos los registros sin ningún tipo de validación ni formateo
  - **Vistas**: son tablas que sólo almacenan una definición de consulta, resultado de la cual se produce una tabla cuyos datos proceden de las bases o de otras vistas e instantáneas. Si los datos de las tablas base cambian, los de la vista que utilizan esos datos también cambiarán.

    - **Instantáneas**: son vistas (se crean de la misma forma) que sí almacenan los datos que muestran, además de la consulta que la creó. Solo modifican su resultado cuando el sistema se refresca cada cierto tiempo. Es como una fotografía de la relación, que sólo es válida durante un periodo de tiempo concreto.
- **Temporales**: son tablas que son eliminadas automáticamente por el sistema.

Tipos de datos

¿Con qué tipo de dato representamos el precio de un libro? Lo haríamos con un número decimal o un número entero. Sin embargo, si queremos representar un nombre, lo haríamos con texto.

Hasta ahora hemos visto vamos a guardar la información en forma de filas y columnas. Las columnas son los atributos o información que nos interesa incluir del mundo real que estamos modelando.

Hemos visto que esos atributos se mueven dentro de un dominio, que no es más que un conjunto de valores. Pues bien, en términos de sistemas de base de datos, se habla más de tipos de datos que de dominios. Al crear la relación (*tabla*) decidimos qué conjunto de datos deberá ser almacenado en las filas de los atributos que hemos considerado. Tenemos que asignar un tipo de dato a cada atributo.

Con la asignación de tipos de datos, también habremos seleccionado un dominio para un atributo.

Cada campo tendrá un nombre y debe tener un asociado un tipo de dato. En función del lenguaje que se utilice existen diferentes formas de nombrar los tipos de datos (C, PHP, SQL, Java,…)

Los tipos de datos más comunes con los que nos encontraremos son:

- **Texto**: almacena cadenas de caracteres (letras, símbolos o números con los que no vamos a realizar operaciones matemáticas).
- **Numérico**: almacena números con los que vamos a realizar operaciones matemáticas.
- **Fecha/hora**: almacena fechas y horas.
- **Verdadero/ Falso**: almacena datos que solo tienen dos posibilidades.
- **Autonumérico**: valor numérico secuencial que el SGBD incrementa de modo automático al añadir un registro (fila).
- **Memo**: almacena texto largo (mayor que un tipo texto).
- **Moneda**: se puede considerar un subtipo de Numérico ya que almacena números, pero con una característica especial, y es que los valores representan cantidades de dinero.
- **Objeto OLE**: almacena gráficos, imágenes o textos creados por otras aplicaciones.

Claves

Si tenemos una tabla (o relación) donde guardamos la información relativa a los **Usuarios** de un juego on-line, en en esta tabla guardaremos los siguientes atributos:

- Login del jugador que será nuestro usuario
- Password
- Nombre y Apellidos
- Dirección
- Código Postal
- Localidad
- Provincia
- País
- Fecha de nacimiento para comprobar que no es menor de edad
- Fecha de ingreso en la web
- Correo electrónico
- Créditos (dinero "ficticio") que tenga

¿Cómo diferenciamos unos usuarios de otros? ¿Cómo sabemos que no estamos recogiendo la misma información? ¿Cómo vamos a distinguir unas (tuplas o filas) de otras? Lo haremos mediante los valores de sus atributos. Para ello, buscaremos un atributo o un conjunto de atributos que identifiquen de modo único las tuplas (filas) de una relación (tabla). A ese atributo o conjunto de atributos lo llamaremos **superclaves**.

Sabemos que una característica de las tablas es que no puede haber dos tuplas (filas) completamente iguales, así que podríamos decir que toda la fila como conjunto sería una **superclave**.

Por ejemplo en la tabla de **Usuarios** podemos tener las siguientes superclaves**:**

- {Nombre, Apellidos, Login, e_mail, Fecha_nacimiento}
- {Nombre, Apellidos, Login, e_mail}
- {Login, e_mail}
- {Login}

¿Con cuál nos quedamos? Entre estas claves a las que llamaremos **candidatas**, tendremos que elegir la que mejor se adapte a nuestras necesidades. Todas las tablas deben tener al menos una clave candidata.

Cada clave candidata estará formado por uno o varios atributos, siempre que identifiquen de manera única a la fila. Cuando una clave candidata está formada por más de un atributo, se llamará **clave compuesta.**

Una clave candidata debe cumplir:

- **Unicidad**: No pueden existir dos filas con los mismos valores para esos atributos
- **Irreducibilidad:** si se elimina alguno de los atributos, deja de ser única

Si elegimos como clave candidata **Nombre,Apellidos,fechaNacimiento,** cumple la unicidad, porque es muy complicado encontrar otra fila con el mismo nombre, apellidos, fecha nacimiento, es irreducible, sería fácil encontrar otro usuario con el mismo nombre y apellidos, o con el mismo nombre y fecha de nacimiento, por lo que son necesarios los 3 atributos para formar la clave.

Para identificar las claves candidatas de una tabla no tenemos que fijarnos en los datos que tenemos en un momento. Puede ocurrir que en ese momento no haya duplicados para un atributo o conjunto de atributos, pero esto no garantiza que se puedan producir más adelante.

El único modo de identificar las claves candidatas es conociendo el significado real de los atributos (campos), ya que así podremos saber si es posible que aparezcan duplicados. Es posible desechar claves como candidatas fijándonos en los posibles valores que podemos llegar a tener. Por ejemplo, podríamos pensar que Nombre y Apellidos podrían ser una clave candidata, pero ya sabemos que cabe la posibilidad de que dos personas puedan tener el mismo Nombre y Apellidos, así que lo descartamos.

Ya tenemos las claves candidatas, ahora vamos a quedarnos con una.

La **clave primaria** de una tabla es aquella clave candidata que se escoge para identificar sus registros de modo único. Ya que una tabla no tiene filas duplicadas, siempre hay una clave candidata y, por lo tanto, la tabla siempre tiene clave primaria. En el peor de los casos, la clave primaria estará formada por todos los atributos de la tabla, pero normalmente habrá un pequeño subconjunto de los atributos que haga esta función. En otros casos, podemos crear un campo único que identifique las filas, por ejemplo un código de usuario, que podría estar constituido por valores autonuméricos.

Las claves candidatas que nos son elegidas como primarias, son las **claves alternativas**

Si en la tabla **Usuarios** escogemos *Login* como clave primaria, el *E_mail o {Nombre, Apellidos, F_Nacimiento}* serán las claves alternativas.

Claves externas, ajenas o secundarias

¿Cómo podemos relacionar una tabla con otra?

Si tenemos una tabla **Partidas,** en la que guardamos las partidas jugadas por los usuarios, debemos tener algún dato que relacione la tabla **Partidas** y la tabla **Usuarios.**

Como una *partida* es jugada por un *Usuario*, en la tabla **Partidas** deberíamos guardar algún dato que identifique al *jugador*, ¿cuál?

Sabemos que la clave primaria es la que nos identifica al usuario, por tanto la clave primaria de la tabla Usuarios que hemos dicho que es el atributo *Login* será el dato que utilicemos para relacionar la partida con el jugador y este dato será la **clave ajena** de la tabla **Partidas.**

Las claves ajenas sí pueden repetirse en la tabla. En nuestro ejemplo, un mismo jugador puede jugar varias partidas y por tanto en la tabla *Partidas*, pueden aparecer el *Login* varias veces

Las **claves ajenas** tienen por objetivo establecer una conexión con la clave primaria que referencian. Por lo tanto, los valores de una clave ajena deben estar presentes en la clave primaria correspondiente, o bien deben ser valores nulos. En caso contrario, la clave ajena representaría una referencia o conexión incorrecta.

No podemos tener una partida de un jugador que no existe. Pero sí podemos tener los datos de una partida y desconocer el jugador de ésta.

Partidas		Usuarios
idPartida		Login
FechaPartida		Password
NombrePartida		NombreyApellidos
Login		Direccion
		…
		Creditos