

Modelado de datos y SQL




Día I – Modelado de datos

- **Modelo entidad-relación**
 - Entidades
 - Atributos y tipos de datos
 - ¿Qué es una relación?
 - Relaciones entre entidades
 - Cardinalidad
- **Normalización**
 - Primera, Segunda y Tercera Forma Normal
 - Desnormalización

Día II - SQL

- **DDL (Data Definition Language)**
 - Creación de tablas
 - Modificación de tablas
 - Creación de relaciones e índices
- **DML (Data Manipulation Language)**
 - Extracción de datos con Select
 - Inserción de registros con Insert
 - Actualización de registros con Update



Día III - SQL

- DML (Data Manipulation Language)

- Consultas avanzadas haciendo uniones de tablas

- Ejercicio Práctico:

- Cargar un set de datos desnormalizado en Excel en una base de datos normalizada haciendo uso de consultas avanzadas.

Modelado de datos



Tipos de información en base a su estructura

Información desestructurada

Es información interpretable por un ser humano, por ejemplo, un texto, pero que carece de una estructura fija y estandarizada.

Información estructurada

Hablamos de información estructurada cuando está delimitada, por tanto, es fácilmente reconocible e interpretable.

Un ejemplo es el de los ficheros **CSV** (comma separated values). En este tipo de ficheros todos los valores están separados por el mismo carácter (coma o punto y coma) y todas las líneas poseen el mismo número de columnas.

Otros ejemplos de este tipo de archivo son las **Hojas de Cálculo Excel**.

Estos formatos maridan genial con las bases de datos, en cuanto a que podemos utilizarlos como formatos de intercambio (importación/exportación) pero **no son bases de datos en sí mismos**.

¿Qué es una Base de datos?

Una **Base de datos** es un almacén de información estructurada y organizada que se almacena de forma electrónica.

El software encargado de este propósito es conocido como “Sistema gestor de base de datos” **SGBD** (o “DataBase Management System” DBMS, en inglés).

Un **SGBD** será utilizado como persistencia de información en aplicaciones externas, rara vez se utiliza desde un usuario final.

Además de permitir crear y manipular los datos, proporcionan seguridad de acceso (diferentes perfiles de usuarios) e integridad en las operaciones (puede haber millones de accesos simultáneos) y el acceso a los mismos está estandarizado.

Ejemplos de **SGBD** comunes:



Modelado de datos

El modelado de datos hace referencia a la planificación de diseño previa que hay que realizar para exprimir al máximo un SGBD.

La herramienta fundamental para ello es la creación de un diagrama **Entidad/Relación**.

Este tipo de diagrama refleja las **entidades** que formarán parte de nuestro sistema, que datos almacenarán y como se **relacionarán** entre sí.

Una **entidad** es la representación abstracta en datos de objetos o conceptos del mundo real y está compuesta de los **atributos** que la definen.

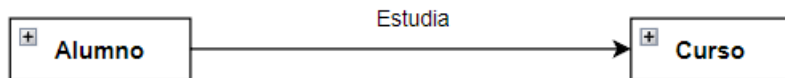
¿Qué datos básicos podrías necesitar para gestionar una academia?

Entidad/Relación I – Identificando entidades y relaciones

Cuando se empieza a analizar la solución a un problema, es común utilizar **sustantivos** para identificar **entidades** y **verbos** para identificar **relaciones**. De este modo, utilizando un lenguaje coloquial podemos obtener fácilmente las entidades:

*“Un **alumno** **estudia** un **curso**”*

Con lo que tenemos una primera aproximación:



Con esta sencilla frase hemos podido reflejar en un diagrama las **entidades** alumno y curso y su **relación**. En el **SGBD**, las entidades se crean como **tablas** y las relaciones son un tipo de **constricción**. Lo veremos en la siguiente clase cuando pasemos a SQL.

Entidad/Relación II – Atributos y tipos

Aparte de las **entidades** y las **relaciones**, un diagrama **entidad/relación** también debe reflejar los **atributos** de las entidades.

Los **atributos** de las **entidades** son los datos que definen cada uno de los registros. Por ejemplo, de un alumno podríamos registrar su **nombre**, **fecha de nacimiento**, **DNI**, **teléfono**, etc. Cuando un **atributo** es requerido se indica como NOT NULL.

Además de cada **atributo** indicamos su tipo de datos:

- Numéricos: INT, FLOAT
- Fecha: DATE, DATETIME
- Texto: CHAR, VARCHAR, TEXT

Alumno	
	nombre VARCHAR(80) NOT NULL
	DNI VARCHAR(9) NOT NULL
	fecha_nacimiento DATE NOT NULL
	telefono INT

Entidad/Relación III – Claves Primarias

Se entiende por **clave primaria** (Primary Key o PK) al **atributo** (o atributos) únicos que definen cada **entidad**.

Son **obligatorios** y **no se pueden repetir**.

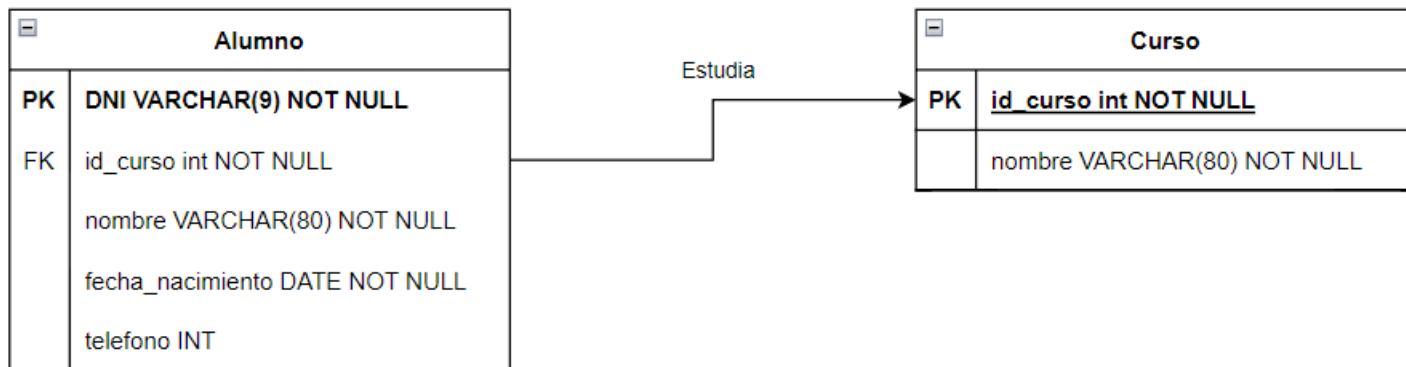
Cuando no hay un **atributo** que defina por sí mismo a una **entidad**, se suele utilizar como **clave primaria** un número entero autoincremental, es decir, el propio SGBD lleva la cuenta del identificador siguiente.

Una **clave primaria** siempre será un índice. Los **índices** los utiliza el **SGBD** para identificar los registros de una tabla y agilizar las búsquedas, así como un índice de un libro indica la página en la que empieza un capítulo para llegar a él rápidamente.

En el ejemplo anterior, un buen atributo candidato a ser **clave primaria** de **alumno** sería el **DNI**, pero en el caso de **curso** es factible utilizar un valor numérico autoincremental. Los valores numéricos ocupan menos que los valores de tipo texto (VARCHAR) y tienen mejor rendimiento, tenedlo en cuenta.

Entidad/Relación IV – Claves Foráneas

Una clave foránea (Foreign Key o FK) es la clave primaria (PK) de otra **entidad** que se propaga al hacer una **relación**. Por tanto, solo puede adquirir valores que pertenezcan a la **entidad** relacionada. Esto se conoce como **integridad referencial**.



En este caso, junto con los datos de **alumno** se almacena el valor de **id_curso** del **curso** que **estudia**.

Entidad/Relación V – Cardinalidad I

Define la naturaleza de la relación, es decir, como se propagan las claves primarias entre entidades:

- 1:1 – De uno a uno. Ambas claves se propagan. Este tipo de relación se utiliza para extender la información que una tabla puede almacenar. En este caso, ambos registros quedan relacionados entre sí.



- 1:N – De uno a muchos. La clave primaria solo se propaga en una dirección. Se utiliza cuando un solo registro de una tabla puede relacionarse con varios de otra.



- N:N – De muchos a muchos. Se utiliza cuando varias líneas de una tabla pueden relacionarse con varias líneas de otra tabla. Desencadena en una tabla nueva.

Entidad/Relación VI – Cardinalidad II

Volviendo al ejemplo del **alumno** que **estudia** un **curso**, hemos visto que la clave se propaga desde **curso** hasta **alumno**, pero no de **alumno** a **curso**. Pero ¿por qué? Veamos ambas posibilidades:

A)

DNI	nombre	apellido	id_curso
00000001A	Aitor	Menta	1
00000002B	Aitor	Tilla	1
00000003C	Elena	Nito del Bosque	2

y

id_curso	nombre
1	Desarrollo Web
2	Desarrollo Mobile

B)

DNI	nombre	apellido
00000001A	Aitor	Menta
00000002B	Aitor	Tilla
00000003C	Elena	Nito del Bosque

y

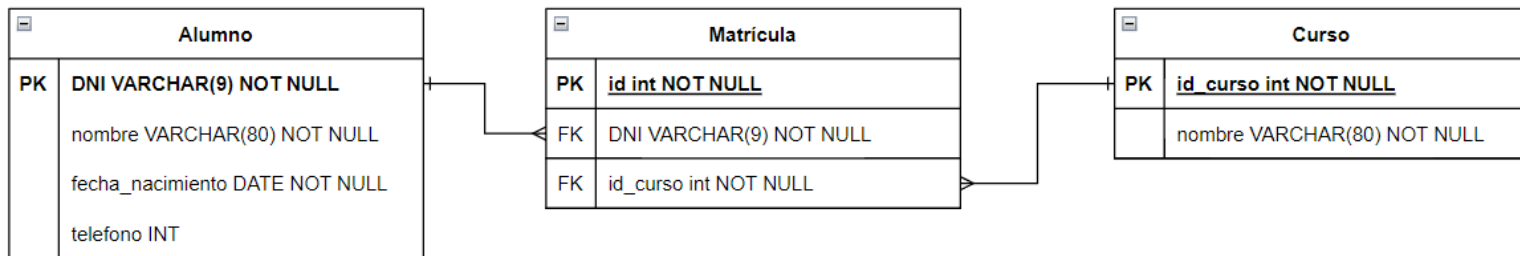
id_curso	nombre	DNI
1	Desarrollo Web	00000001A
1	Desarrollo Web	00000002B
2	Desarrollo Mobile	00000003C

Con lo que sabemos hasta ahora, ¿podría existir el supuesto B?

Entidad/Relación VII – Cardinalidad III

No sé si tendríamos mucho futuro como academia si nuestros alumnos solo pueden estudiar un curso. Vamos a cambiar la frase inicial: “Un **alumno** puede **estudiar** uno o varios **cursos**”.

Esto desencadenaría una relación de **muchos a muchos (N:N)**, por lo que habría que crear una tabla nueva* para poder almacenarlo:



** Llamo “Matrícula” al registro que hace un alumno en un curso. El uso de nombres siempre es subjetivo.*

Ejercicio I – Primer Entidad/Relación

No solo con **alumnos** que se matriculan en **cursos** pueden almacenarse los datos de una academia. Necesitamos almacenar también:

- Ampliar la entidad **alumno** para incluir: **fecha de nacimiento, población, provincia, dirección, email, si es o no apto, fecha de matriculación** y el **número de cuenta**
- Los **profesores**. De cada profesor se necesitan almacenar **DNI, nombre, teléfono, fecha de nacimiento, población, provincia, dirección, email, número de cuenta** y **asignatura** (materia, módulo, llámalo como quieras) que imparte.

Para eso vamos a utilizar la aplicación web draw.io

Normalización I

La **normalización** es el proceso por el cual optimizamos el modelo de datos aplicando normas que aseguran que la información no se repite y cada **entidad** queda definida únicamente con los datos necesarios.

Es **preferible encontrar nuevas entidades más pequeñas** con un significado propio que no **grandes tablas con muchos atributos**.

A los diferentes procesos de normalización se les llama **Formas Normales**. Cada proceso añade una capa más de análisis.

Aunque existen más, veremos las 4 primeras **Formas Normales**.

Normalización II – Primera Forma Normal

1FN

La primera forma normal se basa en cuatro premisas fundamentales:

1. Cada registro debe tener una Clave Primaria.
2. El valor de un atributo es indivisible, no hay atributos con múltiples valores.
3. El orden de los registros no importa.
4. No puede haber atributos con valores nulos.

Por ejemplo, en el ejercicio anterior añadimos la dirección de los alumnos y los profesores. En la dirección almacenaríamos, según está, la calle y el número. Con lo cual, lo ideal sería tener dos atributos en lugar de uno, calle y extensión (en extensión guardaríamos número y piso).

Tampoco podríamos tener un alumno que no tuviera teléfono o email, porque podrían adquirir valores nulos si un alumno no los tuviera. Irían, por tanto, en una nueva tabla.

Normalización III – Segunda Forma Normal

2FN

Para que una tabla se considere en 2FN debe:

1. Estar ya en 1FN
2. Cada atributo queda perfectamente definido por su Clave Primaria. Si no, debería pertenecer a otra tabla.

Esto quiere decir que no deberíamos almacenar junto a un registro atributos que no le pertenezcan. Por ejemplo, no almacenar el nombre del módulo que imparte junto a un profesor, si no que el nombre del módulo debería formar parte de una nueva entidad Módulo.

Normalización IV – Tercera Forma Normal

3FN

Para que una tabla se considere en 3FN debe:

1. Estar ya en 2FN
2. Si los atributos dependen unos de otros, deberían formar parte de una nueva tabla con significado propio.

Esto quiere decir que, aunque varios atributos dependan de la Clave Primaria por sí mismos, también dependen entre sí.

Por ejemplo, si un alumno tiene domiciliado el pago del curso y almacenásemos su número de cuenta y el número de la orden de domiciliación (SEPA en Europa), estos datos podrían ir a una nueva tabla de datos de pago.

¡Para esto están las relaciones 1:1!

Normalización V – Cuarta Forma Normal 4FN

Para que una tabla se considere en 4FN debe:

1. Estar ya en 3FN
2. Los atributos con valores repetidos deberían formar parte de una nueva tabla y ser referenciados por Clave Foránea.

En el ejemplo anterior hemos llevado las direcciones a una nueva tabla con los campos Población, Provincia y Calle, pero haciendo esto tenemos un montón de registros con el mismo valor en población y provincia (tantas veces como calles de una población haya), con lo cual, Población y Provincia deberían ser dos tablas aparte y quedar ligadas a la Calle con Claves Foráneas.

Normalización VI – ¿Alguna optimización más?

Aunque no atiendan a formas normales, podemos observar que tenemos dos tablas similares con valores idénticos: Alumno y Profesor.

Además, si tuviéramos algún caso de un Alumno que además da clase en otro curso distinto, tendríamos los mismos datos del Alumno también en la tabla Profesor.

Conviene, por tanto, hacer un análisis posterior también de las diferentes tablas en su conjunto, no solamente ir tabla por tabla aplicando normalización.

Normalización VII – Desnormalización

Solo cuando se han aplicado todas las formas de Normalización podemos aplicar el camino inverso si lo necesitamos en alguna entidad.

Desnormalizar permite tener valores duplicados en tablas y atributos de otras entidades en la misma tabla cuando nos interesa que la foto de los datos permanezca estática.

Un buen ejemplo es el caso de la facturación. Por ley, una factura una vez se presenta debe permanecer inalterable. ¿Qué pasa si emitimos una factura a un alumno y este ha cambiado de provincia? En una base de datos normalizada, tendríamos el caso de que una factura “mutaría” porque han cambiado los datos del alumno al que se emitió.

Fin de Modelado de datos

Ahora que ya tenemos las nociones básicas de Modelado de Datos, podemos llevar el modelo que hemos aprendido a hacer a un SGBD real.

Para ello se utiliza el lenguaje SQL (Structured Query Language). Este lenguaje se divide en dos partes, DDL (Data Definition Language) Y DML (Data Manipulation Language):

- Para trasladar el modelo Entidad/Relación al SGBD se utiliza DDL o Lenguaje de Definición de Datos. Permite crear la estructura de la Base de Datos.
- Para poblar esa estructura con información, se utiliza DML o Lenguaje de Manipulación de Datos. Además, nos permitirá extraer la información.

El lenguaje SQL posee conectores con prácticamente todos los lenguajes de programación como Python, Java, NodeJS, Swift, .NET, etc.



¡Muchas gracias!



KEEPCODING

Tech School

Madrid | Barcelona | Bogotá

Francisco José Molina Martínez
franciscomoma@gmail.com