**MÓDULO 2 – LENGUAJE DE CONSULTAS A UNA BASE DE DATOS.**

**1.1 Base De datos relaciones**

**Clase 30**

**Objetivo de la jornada:**

* Comprender la importancia del uso y las potencialidades de base de datos.
* Lograr Entender y aplicar base de datos relacionales

**Base de datos Relaciones**

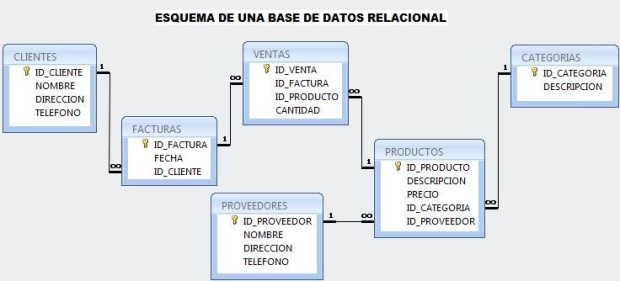
En general, el objetivo del diseño de las bases de datos relacionales es la generación de un conjunto de esquemas relacionales que nos permita almacenar la información sin redundancias innecesarias, pero que también nos permita recuperar fácilmente esa información. Un enfoque es el diseño de esquemas que se hallen en una forma normal adecuada. Para determinar si el esquema de una relación se halla en una de las formas normales deseables hace falta información adicional sobre la empresa real que se está modelando con la base de datos.

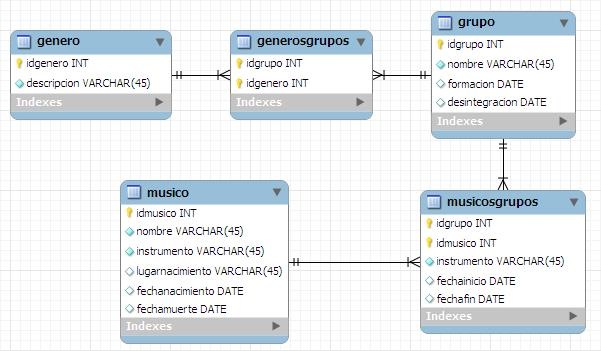
Lo primero que se debe tener muy bien documentados son tus requerimientos. Se debe saber muy bien cuál es la necesidad que se va a resolver/solucionar. Con los requerimientos se podrá ir diseñando cada uno de los componentes de su aplicación. Generalmente, los requerimientos bien documentados son todo lo que se necesita para comenzar a diseñar las bases de datos. Sin embargo, se sugiere tener un prototipo que nos permita entender mejor el flujo de trabajo de la aplicación, ya que con este prototipo se podrá entender procesos que pueden ser difíciles de plasmar o de imaginar. Además, los prototipos pueden dar más información de lo que los requerimientos ofrecen.

**Diseño Conceptual**

Durante esta fase, se plasmaran nuestras entidades y las relaciones que existirán entre ellas. Por lo general se hace a mano, es mucho más sencillo. Cada entidad se identificará con un rectángulo y dentro de este se  colocará su nombre. A cada entidad se colocarán sus respectivos atributos y se resaltara el atributo principal, aquél atributo que identificará cada registro de manera única. Y por último se crearan las relaciones que existen entre dichas entidades.

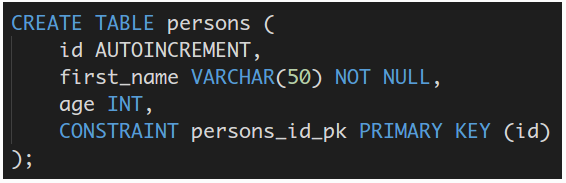
**Diseño Lógico**





**Diseño físico**

En esta última fase ya se debe revisar con detalle los tipos de datos que se utilizaran, sus dominios (qué valores va a permitir), cuales índices se deben crear para optimizar las consultas, entre otros. Aquí ya se escribe nuestro SQL para plasmar todo nuestro diseño en el motor de bases de datos elegido.



# Bases de datos relacionales vs. no relacionales: ¿qué es mejor?

Es muy común entre desarrolladores de aplicaciones encontrarse en una situación de**tener que elegir si se va a usar una base de datos relacional o no relacional**. La mayoría no se lo piensa demasiado y opta por la opción que más conocen y con la que más cómodos trabajan. Tampoco es una decisión catastrófica; en realidad, ya sea la base de datos relacional o no, se puede construir cualquier cosa.

Vale, entonces, **¿por  qué es importante saber en qué se diferencian y cuál deberíamos usar en cada caso?** Pues porque un buen diseño de base de datos con la tecnología apropiada indudablemente**aporta calidad al proyecto**. Dependiendo de la naturaleza de la aplicación, interesa que la base de datos tenga unas características u otras.



## Un poco de historia

Las **bases de datos relacionales** o de **lenguaje de consulta SQL** se empezaron a usar en los años 80 y a día de hoy siguen siendo la opción más popular. En cambio, las**bases de datos no relacionales** o de **lenguaje de consulta NoSQL** solo están empezando a ser más populares en los últimos años. Entre 2012 y 2015, hubo un crecimiento importante en el uso de este tipo de bases de datos. Y aunque desde 2016 su racha se ha quedado un poco estancada, siguen siendo también muy populares.

## Bases de datos relacionales

El principio de las bases de datos relacionales se basa en la organización de la información en trozos pequeños, que se relacionan entre ellos mediante la relación de identificadores.En el ámbito informático se habla mucho de **ACID**, cuyas siglas vienen de las palabras en inglés: **atomicidad, consistencia, aislamiento y durabilidad**. Son propiedades que las bases de datos relacionales aportan a los sistemas y les permiten ser **más robustos y menos vulnerables** ante fallos.

La base de datos relacional más usada y conocida es **MySQL** junto con **Oracle**, seguida por **SQL Server** y **PostgreSQL**, entre otras.

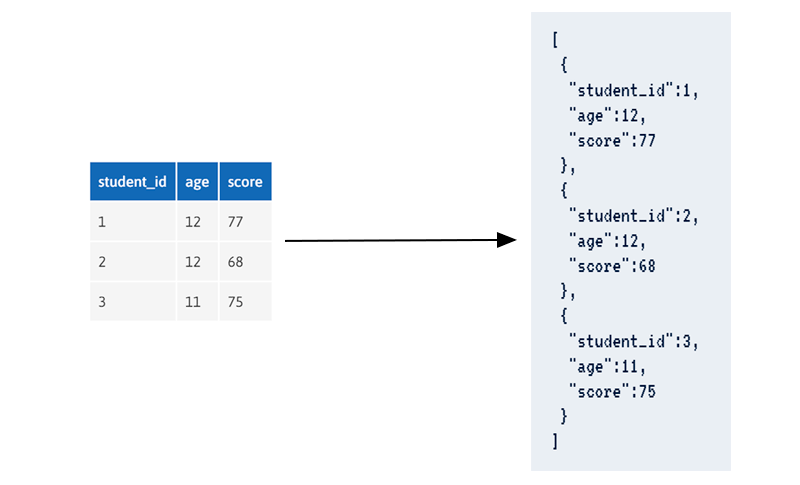
## Bases de datos no relacionales

Como su propio nombre indica, las bases de datos no relacionales son las que, a diferencia de las relacionales, no tienen un identificador que sirva de relación entre un conjunto de datos y otros. Como veremos, la información se organiza normalmente mediante documentos y es muy útil cuando**no tenemos un esquema exacto de lo que se va a almacenar**.

La indiscutible reina del reciente éxito de las bases de datos no relacionales es **MongoDB** seguida por **Redis**, **Elasticsearch** y **Cassandra**.

## Formatos

La información puede organizarse en **tablas** o en **documentos**. Cuando organizamos información en un Excel, lo hacemos en formato tabla y, cuando los médicos hacen fichas a sus pacientes, están guardando la información en documentos. Lo habitual es que las bases de datos **basadas en tablas** sean bases de datos **relacionales** y las **basadas en documentos** sean**no relacionales**, pero esto **no tiene que ser siempre así**. En realidad, una tabla puede transformarse en documentos, cada uno formado por cada fila de la tabla. Solo es una cuestión de visualización.



Los dos esquemas de la imagen contienen exactamente la misma información. Lo único que cambia aquí es el formato: cada documento de la figura de la derecha es una fila de la figura de la izquierda.

Se ve más claro en la tabla, ¿verdad? Lo que pasa es que a menudo en una base de datos no relacional **una unidad de datos puede llegar a ser demasiado compleja como para plasmarlo en una tabla**. Por ejemplo, en el documento JSON de la imagen que se muestra a continuación, al tener elementos jerárquicos, es más difícil plasmarlo en una tabla plana. Una solución sería plasmarlo en varias tablas y, por tanto, necesitar de relaciones

.

[

{

"student\_id":1,

"age":12,

"subjects":{

"mathematics":{

"scores":[7,8,7,10],

"final\_score":8

},

"biology":{

"scores":[6,6,5,7],

"final\_score":6

}

}

}

]

Esto explica por qué las bases de datos relacionales suelen servirse de tablas y las no relacionales de documentos JSON. En cualquier caso, a día de hoy, **las bases de datos más competitivas suelen permitir, de una forma u otra, operaciones de los dos tipos.** Por ejemplo, el servicio de base de datos en la nube **BigQuery** que ofrece Google es, en principio, una base de datos de lenguaje de consulta SQL, por lo que permite fácilmente relacionar varias tablas, pero, a su vez, permite insertar elementos jerárquicos JSON, más propios de bases de datos no relacionales.

**La diferencia entre el éxito y el fracaso recae, sobre todo, en el diseño del modelo**. Es decir, si se decide que el mejor enfoque es usar una base de datos relacional, no es suficiente con meter la información a lo bruto en una base de datos relacional y esperar a que se relacione sola, porque eso no va a ocurrir. De nada sirve elegir la base de datos más apropiada para nuestro sistema, si luego no se hace un buen diseño.

## Ejemplo práctico: base de datos relacional

Imaginemos que **tenemos una plataforma online que ofrece cursos de idiomas**. Los clientes contratan o se suscriben al idioma y al nivel que más les puede interesar, y, además, tienen la opción de elegir qué tipo de suscripción quieren: mensual, trimestral o anual. Y dependiendo de esta opción, se les aplicará un descuento u otro.

El**primer diseño** de base de datos que propongo es una tabla donde cada fila corresponde con un servicio contratado por un cliente. Toda la información está contenida en una sola tabla, por tanto, **no es relacional**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| fecha | cliente | idioma | nivel | suscripción | precio | descuento\_% | precio final |
| 25/06/2018 | Pedro | Inglés | Intermedio | Mensual | 7 | 0 | 7 |
| 25/06/2018 | Pedro | Chino | Principiante | Mensual | 9 | 0 | 9 |
| 01/07/2018 | Aurelia | Francés | Avanzado | Anual | 8 | 25 | 6 |
| 03/07/2018 | Federico | Inglés | Intermedio | Trimestral | 7 | 10 | 6.3 |

### Problemas que podemos encontrar con este modelo

* **No sabemos si el Pedro de la primera fila es el mismo cliente que el Pedro de la segunda fila** o si son dos clientes con el mismo nombre. Sí, podríamos incluir el e-mail para que haga de identificador único, pero es una solución cogida con pinzas.
* Si algún precio o descuento cambia, **hay que modificarlo en todas las filas en las que aparece** y, si no se hace correctamente, puede dar lugar a discrepancias. No tiene sentido que la información esté duplicada de esta manera.
* Si un cliente cambia su suscripción, **habría que cambiar tanto el campo de suscripción como el precio**. Y también puede dar lugar a discrepancias si no se hace correctamente.
* Al tener la columna de “precio final” **se está duplicando información**, ya que se puede calcular fácilmente con las columnas “precio” y “descuento\_%”. Esto también puede dar lugar a discrepancias.
* No hay manera de saber **qué idiomas y niveles hay disponibles, ni cuál es su precio** hasta que alguien lo contrate.

### ¿Cómo solucionamos todos estos problemas?

Parece que esta situación está pidiendo a gritos **un diseño de base de datos relacional, donde se recoja la información en varias tablas y no solo en una**.

Empezamos con la primera tabla; esta contendrá solamente la información del **cliente**.

|  |  |
| --- | --- |
| cliente\_id | nombre\_cliente |
| 1 | Pedro |
| 2 | Aurelia |
| 3 | Federico |

Por otro lado tenemos la tabla contenedora de las **clases disponibles**. Cada una con su nivel y precio base.

|  |  |  |  |
| --- | --- | --- | --- |
| programa\_id | idioma | nivel | precio |
| 1 | alemán | principiante | 7 |
| 2 | chino | principiante | 9 |
| 3 | francés | avanzado | 8 |
| 4 | inglés | intermedio | 7 |

En esta tabla podemos ver todos los cursos disponibles. **En el diseño principal, como nadie se había suscrito al curso de alemán, ni siquiera podíamos saber que existía.**

A este precio base luego se descontará un porcentaje, según la suscripción que los clientes elijan.

|  |  |  |
| --- | --- | --- |
| suscripcion\_id | tipo | descuento\_% |
| 1 | Mensual | 0 |
| 2 | Trimestral | 10 |
| 3 | Anual | 25 |

Y por último, tenemos **la tabla que relaciona todo**: a cada cliente con la clase o clases contratadas y el tipo de suscripción.

|  |  |  |  |
| --- | --- | --- | --- |
| id | cliente\_id | programa\_id | suscripcion\_id |
| 1 | 1 | 4 | 1 |
| 2 | 1 | 2 | 1 |
| 3 | 2 | 3 | 3 |
| 4 | 3 | 4 | 2 |

Pedro, que es el cliente con identificador 1, se había suscrito mensualmente (id=1) a inglés intermedio (id=4) y chino principiante (id=2). Por eso, en las dos filas en las que el identificador de cliente “client\_id” es 1, el identificador de programa es 4 y 2, y el identificador de suscripción es un 1.

Fijaos también que **no hemos apuntado el precio final en ningún lado**. No es necesario, ya que, conociendo el precio del programa y el descuento de la suscripción elegida, se puede calcular de inmediato. Y así evitaremos duplicidad y la posibilidad de discrepancias en nuestros datos.

¡Y ya lo tenemos! No es tan difícil, ¿no?

Antes de continuar, **os propongo un ejercicio: ¿cómo haríais para incluir la posibilidad de tener cupones de descuento?** Pensad que estos cupones son canjeables por cada curso que se contrata y cada uno puede proporcionar un descuento diferente.

## Ejemplo práctico: bases de datos no relacionales

Quizá os estéis preguntando**“si las bases de datos relacionales son tan prácticas, ¿en qué situaciones es buena idea trabajar con las no relacionales?”**.  Si algo tienen de malo las bases de datos relacionales, es que son como Sheldon Cooper,**tienen que saber de antemano qué es y cómo es lo que van a almacenar**. En cambio, las bases de datos no relacionales son más flexibles, se lo tragan todo, sin importar su estructura.

Imaginemos que **hemos mandado unas máquinas al espacio para que nos reporten qué es lo que encuentran en su viaje**. Obviamente, no sabemos a ciencia cierta qué se van a encontrar. De alguna forma, tienen una inteligencia artificial instalada que reconoce los objetos con los que se va encontrando y también tienen sensores instalados. Pero no sabemos bien qué miden, ya que cada máquina tiene sensores diferentes. Cada 24 horas envían un resumen de lo que han visto durante el día.

{

"maquina\_id":1,

"timestamp":149992693000,

"coordenadas":"75988823.567, 55375867.098, 12676444.311",

"encontrado":[

"roca",

"agua",

"roca",

"roca",

"algo que parece un animal",

"roca"

],

"temperatura":{

"min":-50,

"max":-49

},

"ruido":{

"min":72,

"max":4549

}

}

{

"maquina\_id":2,

"timestamp":1499925677000,

"coordenadas":"66635675.920, 78021134.727, 53580995.751",

"temperatura":{

"min":-50,

"max":-49

},

"humedad":{

"min":2%,

"max":5%

}

}

Cada uno de estos documentos JSON contiene la información reportada en cada envío por cada máquina. La máquina con identificador 1 está reportando datos de temperatura y ruido, mientras que la de identificador 2 reporta temperatura y humedad. No sabemos qué sensores tendrá instalados la siguiente y, mucho menos, qué y cómo reportarán las máquinas que aún no se han mandado y los ingenieros están montando.

No merece la pena ponerse a diseñar una base de datos relacional para almacenar esta información. En este caso, **lo mejor es dejar a una base de datos no relacional que se trague todo lo que las máquinas reportan, tal cual.**

Además, la **finalidad** del sistema es meramente **científica** y no se contempla la existencia de usuarios a los que se les deba la garantía de consistencia que ofrecen las bases de datos SQL. Simplemente, **se quiere almacenar todo para un futuro análisis.**

Una vez almacenados en la base de datos no relacional se podrá pedir y visualizar la información de diferentes maneras. Y si en algún momento se necesita consumir los datos de una forma más estructurada, **siempre podremos procesar y volcar la información a una base de datos relacional.** Pero es que, muy probablemente, no sea necesario.

|  |  |
| --- | --- |
| **Plan Formativo:** | **Nivel de Dificultad:** |
| **Full Stack Java Trainee** | **Básico/medio.** |
| **Módulo:** | **Tema:** |
| **LENGUAJE DE CONSULTAS A UNA BASE DE DATOS** | **Introducción a los modelos de datos.** |
| **Título del Problema** | **Guía de preguntas, Problemas propuestos,.** |
| **Intención del aprendizaje o aprendizaje esperado** | |
| * Modelar base de datos relacionales en UMl. * Conceptualizar y realizar consultas en base en modelo de datos. | |
| **Planteamiento del Problema** | |
| En base a los siguientes enunciados realice el correspondiente modelo físico y lógico relacional.1)Fábrica de Pelotas “Golazo” Solicitan nuestros servicios para resolver el almacenamiento de datos de un sistema de gestión de la producción de una fábrica de pelotas. La fábrica se compone de una serie de plantas, cada una identificada por un color. De las plantas conocemos la superficie en metros cuadrados y la lista de procesos que se llevan a cabo dentro de ellas; de estos procesos sólo conocemos su nombre y un grado de complejidad asociado. Dentro de cada planta se encuentran las máquinas. Cada máquina es de una marca y un modelo, y se identifica por un número; este número es único a lo largo de todas las plantas. Cada máquina es operada por técnicos, debemos conocer en qué rango de fechas los técnicos estuvieron asignados a esa máquina, y además en qué turno (mañana, tarde o noche). De los técnicos conocemos su DNI, nombre, apellido y fecha de nacimiento, aparte de una serie de números telefónicos de contacto. Existen situaciones normales en las que una máquina sale de servicio y debe ser reparada, lo único que nos interesa conocer aquí es cuál otra máquina está asignada para tomar el trabajo que ella no puede realizar.(Tiempo aproximado 30 minutos)2) Sistema de Ventas Se quiere diseñar una BD que permita registrar las ventas de una empresa. Específicamente, esta empresa necesita llevar un control de proveedores, clientes, productos y ventas. Un proveedor se modela con Rut, nombre, dirección, teléfono y página web. Un cliente también se modela con Rut, nombre y dirección, pero puede tener varios teléfonos de contacto. De cada dirección, nos interesa su calle, número, comuna y ciudad. Tanto para los proveedores como los clientes, Rut es un valor único (equivalente al DNI). De los productos, sabemos que tienen un identificador único, nombre, precio actual, stock y nombre del proveedor que los comercializa. Además se organizan en categorías, y cada producto se clasifica solamente en una de ellas, pero sin embargo una categoría clasifica varios productos. De ellas nos interesa saber su id, nombre y descripción. Sabemos que un producto es comercializado por varios proveedores, pero que un proveedor provee un solo producto. Por razones de contabilidad, se debe registrar la información de cada venta , las cuales tienen un número de factura (que es único), fecha, cliente, descuento y monto final. A su vez, sabemos que una venta se compone de varios productos, y por eso nos interesa el precio al momento de la venta del producto, la cantidad vendida y el monto total por él. Tenga en cuenta que un producto puede estar en varias ventas, pero que podemos tener un producto que no haya sido vendido. Adicionalmente, sabemos que cada cliente puede realizar varias ventas, y en una venta solamente participa un cliente.Modelo su correspondiente modelo físico y lógico. (Tiempo aproximado 40 min)3) Cadena de Deportes Una cadena de casas de deportes desea realizar una base de datos para manejar sus sucursales, empleados, productos y clientes. De las sucursales se sabe el número único que la identifica dentro de la cadena, el domicilio y la ciudad. De los empleados el legajo, el nombre, el dni, el domicilio (calle, número y ciudad) y los números de teléfono en los cuales puede ser contactado. Los empleados trabajan en diferentes sucursales en diferentes días de la semana y en cada sucursal tiene asignado un horario en particular, que puede no ser el mismo en diferentes sucursales. Por ejemplo, el empleado GBA trabaja los lunes de 9hs. a 18hs. en la Sucursal 1, y los martes de 10hs. a 20hs. en la Sucursal 2. En cada sucursal trabajan varios empleados. De los productos se conoce un código, una descripción, un color y un costo fijo de fabricación. A su vez, existen también las fábricas que son identificados con CUIT, nombre, país de origen, cantidad de empleados y nombre de gerente. Cada producto es fabricado en una sola fábrica, y cada fábrica solamente realiza un tipo de producto. El costo fijo de fabricación no depende de la fábrica. Cada sucursal puede vender varios productos, y a su vez, cada producto puede ser vendido por varias sucursales. Cada sucursal establece cuál es el precio del venta del producto que ofrece. Es decir, un mismo producto podría tener diferentes precios en diferentes sucursales. De los clientes se conoce el código de cliente, el dni, el nombre, la fecha de nacimiento y la ciudad en la que vive. A su vez, también sabemos que cada cliente puede utilizar varias tarjetas de crédito, que son identificadas por el nombre de la tarjeta, el número, el código de seguridad y la fecha de vencimiento. Cada cliente solamente realiza compra en una sola sucursal, y en una sucursal pueden comprar varios clientes. Cada sucursal le ofrece a sus clientes un descuento fijo por su fidelidad en las compras.Modelo su correspondiente modelo físico y lógico. (Tiempo aproximado 45 min) | |
| **Datos de apoyo al planteamiento** | |
|  | |
| **Pregunta Guía:** | |
|  | |
| **Recursos Bibliográficos.** | |
| <https://www.monografias.com/trabajos81/db-base-de-datos/db-base-de-datos.shtml> | |
| **Nombre del Autor y Fecha:** | |
| **Mario Canedo 21/11/2019** | |