**UNIDAD 2: Características de Base de Datos**

1. [UNIDAD 2: Introducción](https://learn.nextu.com/mod/page/view.php?id=9657&pid=P_WEB_DATABASE)

# **Tipos de Bases de Datos**

Existen diversos tipos de bases de datos, los cuales se han agrupado en dos grandes categorías, las **bases de datos relacionales** y las **no relacionales**. Las primeras representan la forma más tradicional de organizar los datos en estructuras, generalmente, conocidas como **tablas**. La segunda engloba todas aquellas formas de constituir datos de una manera **flexible**, que en algunos casos puede ser una alternativa más natural de resolver los problemas. En esta unidad, nos adentramos un poco más en el mundo de las bases de datos, permitiéndote así reconocer estas dos categorías en función de sus **componentes estructurales**. Usando un enfoque más **práctico**, te daremos la oportunidad de aplicar los elementos de las bases de datos SQL y NoSQL en diversos escenarios, con la finalidad de que puedas diferenciar en qué situaciones un paradigma puede resultar mejor que el otro.

# **Objetivos de aprendizaje**

1. **Usar las sentencias** básicas de SQL en la gestión de bases de datos relacionales.
2. **Diferenciar las bases de datos** SQL de las NoSQL en función de sus características elementales.
3. **Identificar la estructura** de los documentos JSON en función de los valores, objetos y arreglos.
4. Lección 1: Tipos de Base de Datos

[2.1. SQL vs NoSQL](https://learn.nextu.com/mod/lesson/view.php?id=9658&pid=P_WEB_DATABASE)

SQL – Relacionales:

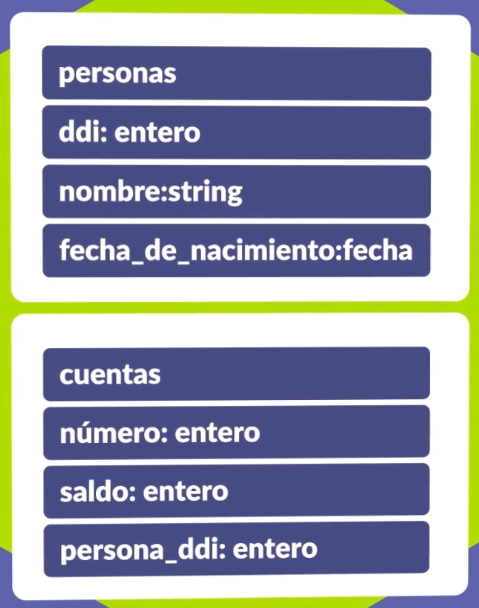
* Los datos se estructuran en un esquema, como se van a organizar las entidades, en tablas, todos los registros contendrán la misma estructura, sino se indica un valor en una columna se tomará un valor por defecto que en la mayoría de los casos será nulo.
* Lenguaje manejador de la base de datos
* Se basan en el principio de integridad que significa que los datos no deben ser redundantes, los datos siempre deben ser consistentes
* Se caracteriza por poseer una escalabilidad Vertical, lo cual quiere decir que si se desea una DB más potente solo se debe incrementar el tamaño del recurso, si necesitamos más capacidad para la base de datos significa que necesitamos más capacidad en los servidores.
* SQL más conocidos: MySQL, PostgreSQL, Oracle, Microsoft Access

NoSQL – No Relacionales

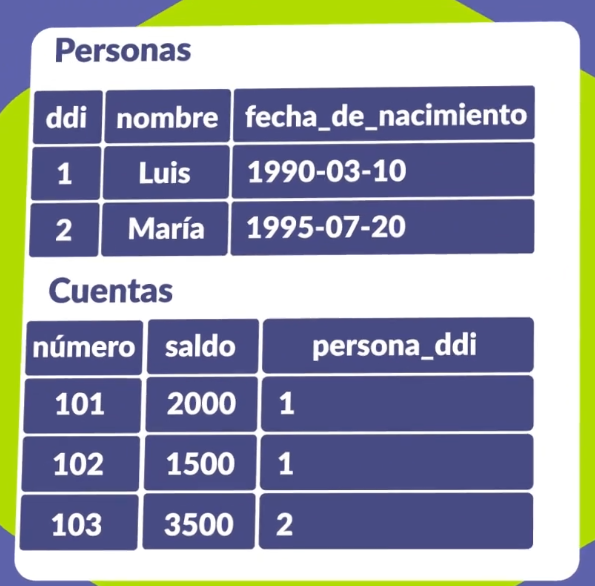
* Hay diversas alternativas para almacenar los datos, no poseen una estructura definida
* Alta velocidad de acceso y alta disponibilidad de los datos, incluso permitiendo mucha redundancia
* Las bases de datos se agrupan en nodos o clusters, así si deseamos aumentar la capacidad de la base de datos solamente se aumentan la cantidad de nodos
* Existen varios de base de datos no relacionales:
  + Base de datos orientada a documentos, como: MongoDB y CouchDB
  + Base de datos orientada a clave/valor, como MemCacheD y Redis
  + Base de datos orientada a multicolumna, como: Cassandra
  + Base de datos orientada a grafos, como: Neo4J

Ejemplo de DB para un Banco:

Tablas:



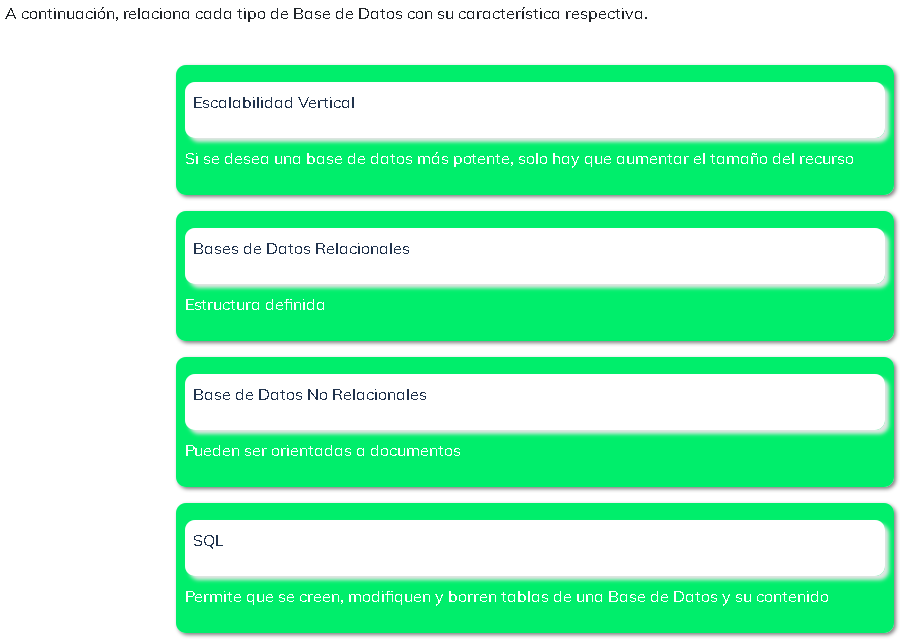
Datos: Se ha dado una relación de la tabla a través del campo persona\_ddi 🡪 ddi



2.2. Actividad Interactiva 1

# Analiza y relaciona

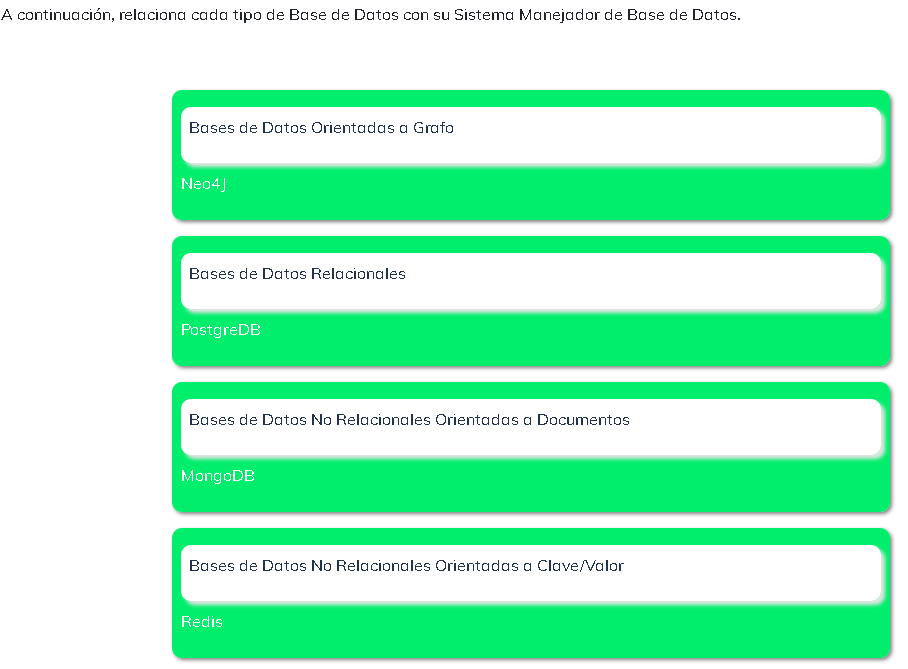
A continuación, relaciona cada tipo de Base de Datos con su característica respectiva.



2.3. Actividad Interactiva 2

# Analiza y relaciona

A continuación, relaciona cada tipo de Base de Datos con su Sistema Manejador de Base de Datos.



2.4. Ejercicio 1

Tenemos un sistema de manejo de las tareas de los estudiantes de una Universidad. Hemos identificado dos Entidades: Estudiante y Calificación. La Entidad Estudiante tiene tres Atributos: Identificador del Estudiante (ID) del tipo entero, Nombre del tipo string y Apellido del tipo string. La Entidad Calificación tiene como Atributos: Tarea del tipo string, Fecha de Entrega del tipo fecha, Calificación Obtenida del tipo entero y Estudiante ID que referencia al estudiante que realizó la tarea y es del tipo entero.

**Realiza la representación en tablas (usando estándar de nombres) de estos elementos de la base de datos usando el paradigma Relacional para los siguientes casos:**

* El estudiante de Luis (Nombre) Aguilar (Apellido) y ID (123) obtuvo la calificación de 50 puntos en la Tarea 1 entregada el 16 de marzo de 2018.
* La estudiante de Manuela (Nombre) Silva (Apellido) y ID (567) obtuvo la calificación de 90 puntos en la Tarea 1 entregada el 15 de marzo de 2018.

2.5. Más tipos NoSQL y Comparación

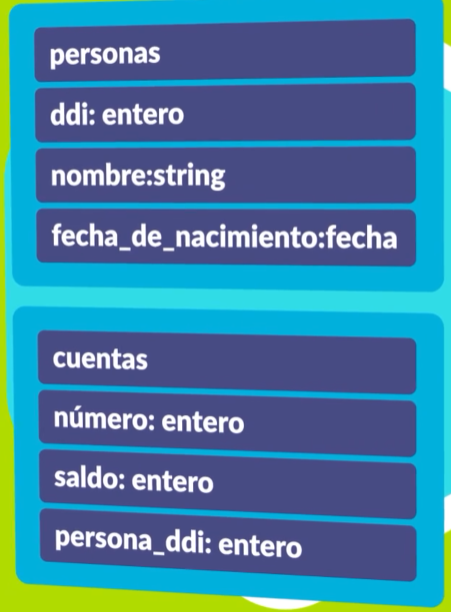
Hay mucha diversidad es la característica principal de las DB No relacionales.

Comparación:

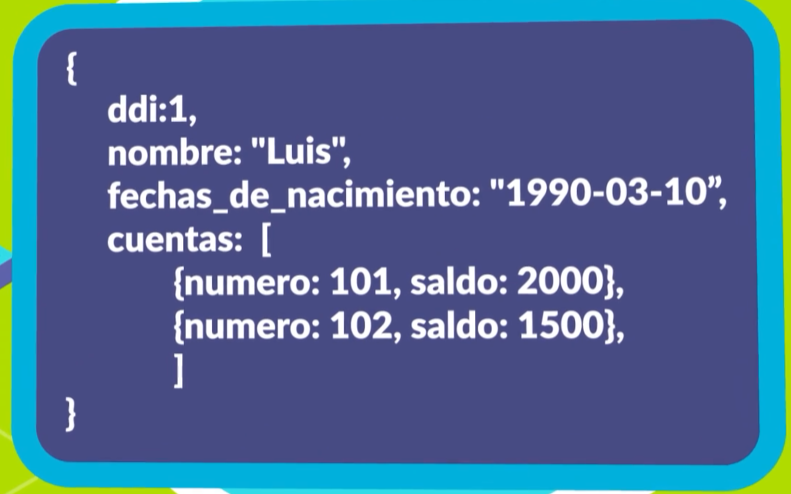
SQL

NoSQL – se representan en forma de documento, y un documento puede estar embebido en otro formando una jerarquía

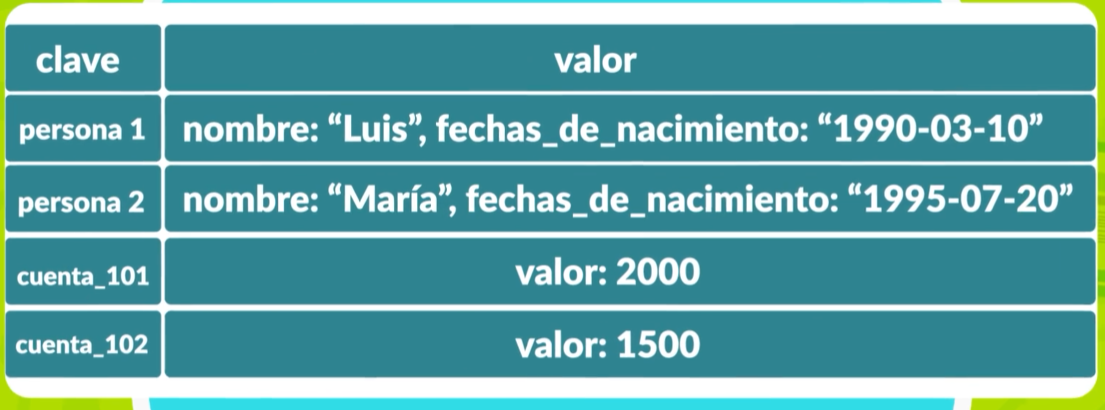
Ejemplo de cuenta bancaria



Los datos se almacenan en colecciones, las colecciones tienen documentos:



Las base de datos no relacionales orientadas a clave >> valor, es una forma más simple de guardar y recuperar los datos, esto hace que su implementación sea eficiente y la recuperación de los datos sea rápida. Usualmente estos sistemas trabajan con datos en memoria. Así que son ampliamente utilizados en sistemas que requieren almacenar datos en cache. Tales como el cache de páginas web.

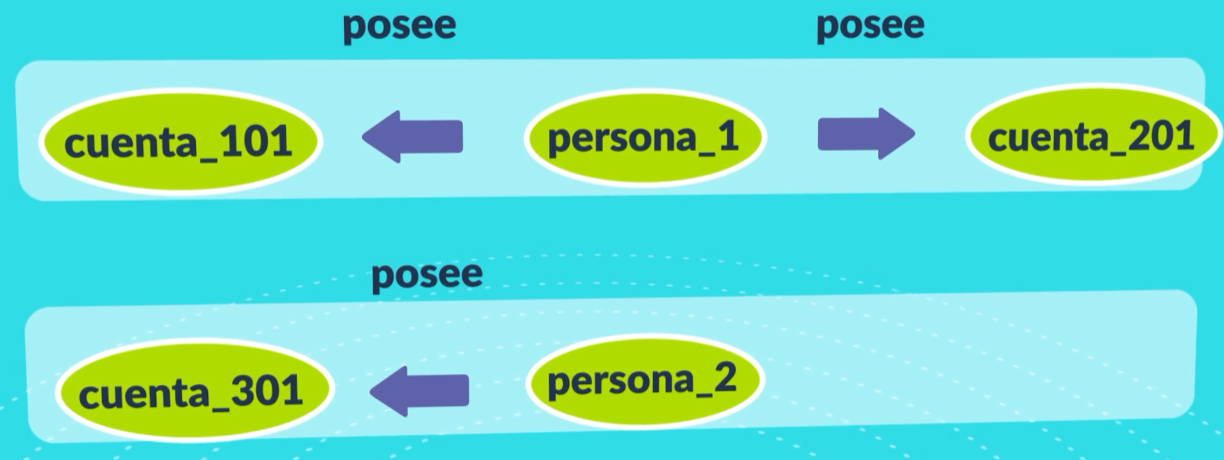


Proporciona una gran flexibilidad donde no se define una estructura de como funciona una persona o una cuenta. Dada una clave se da un valor, y para dada una clave recuperamos un valor.

El paradigma no relacional es el orientado a grafo, un grafo se define como un conjunto de nodos interconectados por arcos, que se usan para representar sistemas, tales como: una red de computadoras, donde los nodos son las computadoras y los arcos los enlaces de comunicación.

Las redes sociales también se pueden representar usando grafos, donde cada nodo es una persona, y los arcos muestran como se conectan a través del uso de alguna red social.

En la base de datos orientados a grafos vamos a encontrar nodos relacionados entre ellos. Los nodos a su vez van a tener propiedades o atributos.



Comparación:

SQL – relacionales

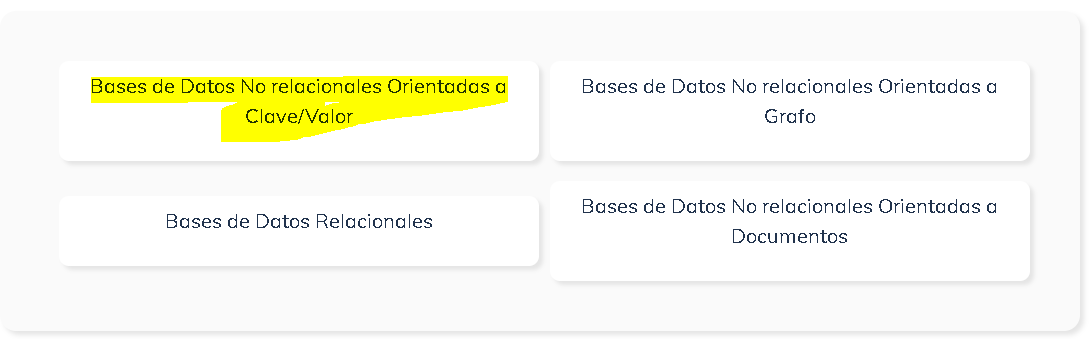
NoSQL – No relacionales

|  |  |  |
| --- | --- | --- |
| Criterio | SQL | NoSQL |
| Estructura | Definida (Definición de columnas y tablas y que tipo de datos) | Flexible, por ejemplo en orientada a documentos. Un documento puede tener 3 atributos y otro solamente 2. |
| Escalabilidad | Escalabilidad Vertical – Que se refiere a que tenemos que tener un servidor con más recursos si se requiere una base de datos con más capacidad | Escalabilidad Horizontal – que significa que está diseñadas para conectarse a otros nodos. Es decir que podemos agregar más nodos al sistema, así haciéndolo escalable |
| Orientación | Integridad / Consistencia - | Velocidad – Disponibilidad / Cluster |
| Acceso | SQL – Se hace uso del lenguaje SQL | Variable – se tiene una variedad de mecanismos para el acceso a los datos |

**2.6. Actividad Interactiva 3**

**Prueba tus conocimientos**

Usualmente las \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ trabajan con datos en memoria, así que son ampliamente usadas en aplicaciones que requieren almacenar datos en caché, tales como el caché de páginas Web.



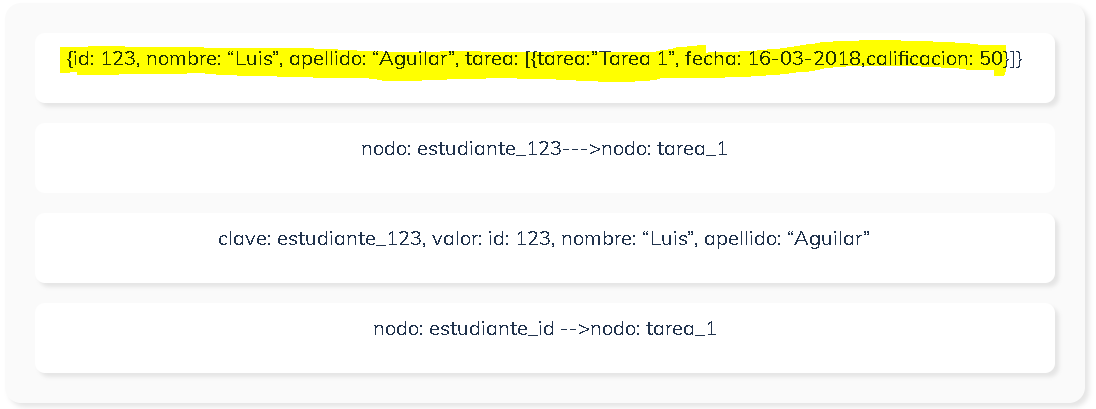
2.7. Actividad Interactiva 4

**Prueba tus conocimientos**

Tenemos un sistema de manejo de las tareas de los estudiantes de una Universidad. Hemos identificado dos Entidades: Estudiante y Calificación. La Entidad Estudiante tiene tres Atributos: Identificador del Estudiante (ID) del tipo entero, Nombre del tipo string y Apellido del tipo string. La Entidad Calificación tiene tres Atributos: Tarea del tipo string, Fecha de Entrega del tipo fecha, Calificación Obtenida del tipo entero y EstudianteID que referencia al estudiante que realizó la tarea y es del tipo entero. Considerando la representación en documento JSON de estos elementos de la base de datos usando el paradigma Relacional para el siguiente caso:

El estudiante de Luis (Nombre) Aguilar (Apellido) y ID (123) obtuvo la calificación de 50 puntos en la Tarea 1 entregada el 16 de marzo de 2018.

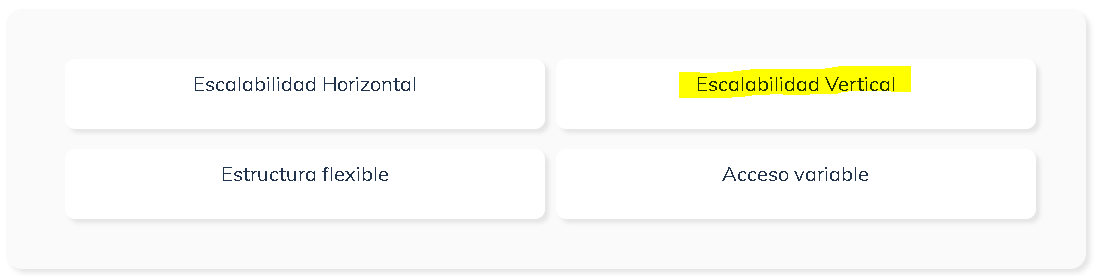
**¿Cuál de las siguientes representaciones corresponde a un sistema de base de datos No Relacional Orientado a Documentos JSON?**



2.8. Actividad Interactiva 5

**Prueba tus conocimientos**

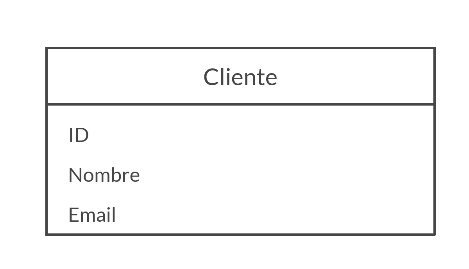
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ es una característica presente en un Sistema de Base de Datos Relacional pero no en uno No Relacional.

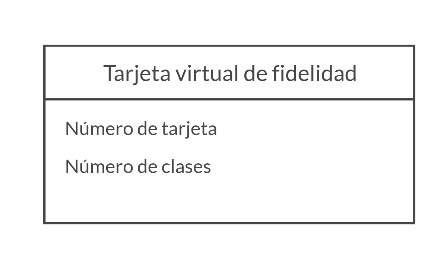


2.9. Ejercicio 2

Desafío práctico

Se está desarrollando un sistema para clientes leales de una Nueva Academia de Zumba. Cada cinco clases el cliente obtiene una gratis. Después de un análisis inicial de los requerimientos del sistema se han identificado las siguientes entidades:





Genera la representación de las entidades, sus atributos y tipos de datos usando los siguientes enfoques de bases de datos:

* Sistema de Bases de Datos Relacional
* Sistema de Bases de Datos No Relacional orientado a Documentos JSON

**Muestra la representación para el cliente cuyo ID es el 1002, cuyo nombre es Ernesto y cuyo email es ernesto@miemail.com y que tiene una tarjeta de fidelidad número 1 y número de clases acumuladas hasta la fecha 3.**

**Solucion:**

- Sistema de Bases de Datos Relacional

|  |  |  |  |
| --- | --- | --- | --- |
| Tabla - Cliente | | |  |
| id | nombre | email | tarjeta\_id |
| 1002 | Ernesto | ernesto@miemail.com | 1 |

|  |  |
| --- | --- |
| Tabla – Tarjeta virtual de Fidelidad | |
| tarjeta\_id | numero\_clases |
| 1 | 3 |

* Sistema de Bases de Datos No Relacional orientado a Documentos JSON

{

“Id”: “1002”,

“Nombre”: “Ernesto”,

Email: “ernesto@miemail.com”

Tarjeta: [{

“Id”: “1”,

“numero\_de\_clases”: 3

}]

}

1. Lección 2: Base de Datos Relacionales
   1. [Definiciones en Base de Datos Relacionales](https://learn.nextu.com/mod/lesson/view.php?id=9659&pid=P_WEB_DATABASE)

Cada entidad se debe transformar en una tabla y cada atributo se debe convertir en campos, los campos son las columnas de las tablas y cada instancia de una entidad es una fila,

Una tabla tiene una única llave primaria que representa una fila, usualmente es una clave de tipo entero.

La clave foránea es un campo que apunta a una clave primaria en otra tabla, para poder establecer relaciones entre tablas. (Asociaciones).

Varios tipos de asociaciones:

1:N – uno a muchos

N:1 – muchos a uno

N-M – de muchos a muchos – Se establece una tabla intermedia con las asociaciones

1:1 – de uno a uno

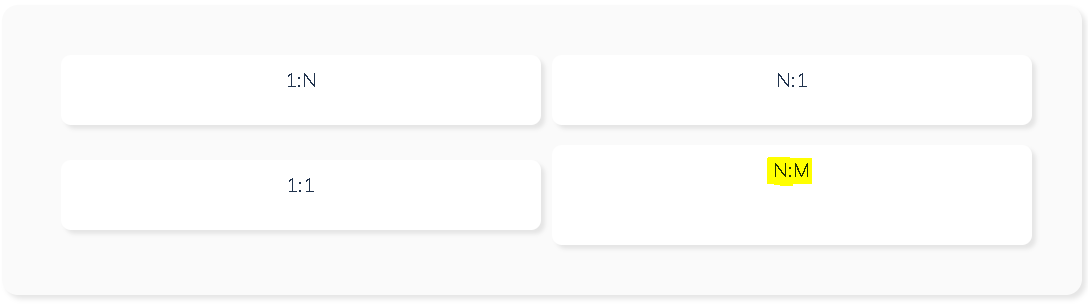
La clave primaria se identifica con una asterisco: ejemplo Id\*

* 1. Actividad Interactiva 1

**Prueba tus conocimientos**

Un nuevo restaurante ha abierto sus puertas en la ciudad. Te invitamos a ayudar a sus dueños a diseñar la base de datos para su nuevo sistema de Especialidades Culinarias. Hasta el momento se han definido dos Entidades: Cocinero y Plato. La tabla cocineros incluye información de los cocineros que laboran en el restaurante, mientras que la tabla platos incluye los platos que este ofrece a sus clientes. Los cocineros del restaurante son seleccionados de forma tal que puedan preparar todos los platos del Menú.

**¿Qué tipo de asociación es la más conveniente entre Cocinero y Plato (en ese orden)?**

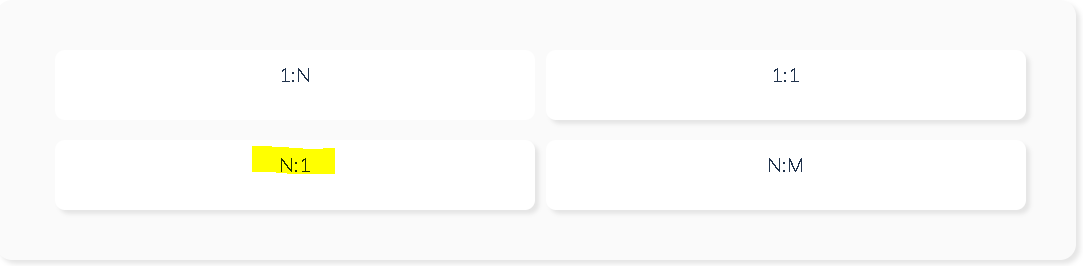


* 1. Actividad Interactiva 2

**Prueba tus conocimientos**

En el restaurante, se ha agregado información acerca de los clientes y se les permite a estos escoger un único plato favorito.

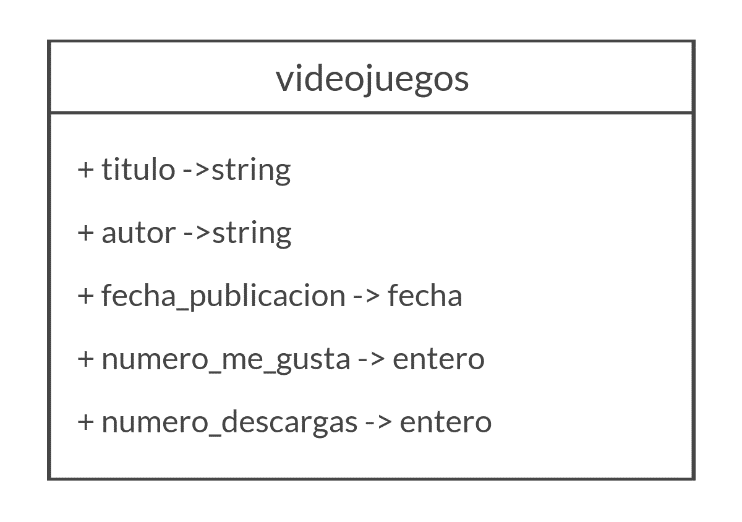
**¿Qué tipo de asociación es la más conveniente entre clientes y platos (en ese orden)?**



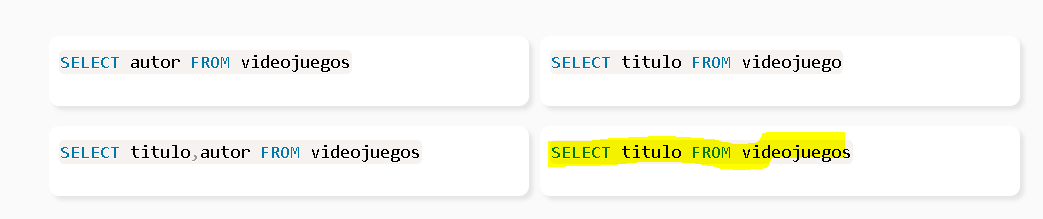
* 1. SQL Sentencia SELECT
  2. Actividad Interactiva 3

Prueba tus conocimientos

MiVideojuego es una tienda que vende juegos de computadoras en línea. Ellos tienen un sistema para el control de las ventas, el cual tiene una base de datos que incluye la tabla videojuegos con todos los videojuegos disponibles en la tienda. Los campos y tipos de la tabla se muestran a continuación:

  
El encargado de la tienda virtual le ha pedido a uno de los miembros del staff que proporcione una lista de todos los títulos de los videojuegos disponibles en la tienda.

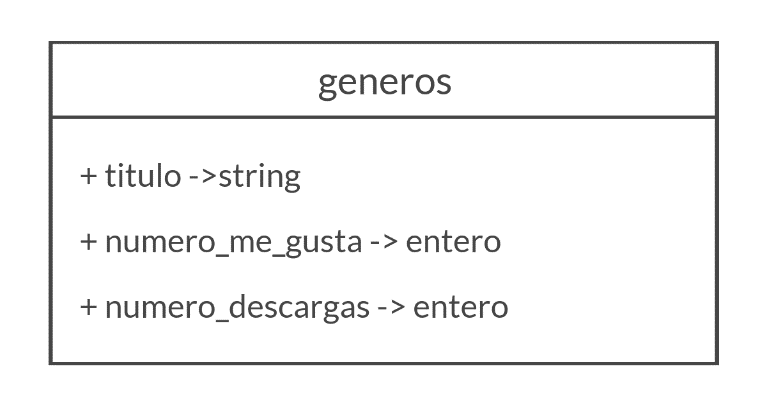
**¿Qué sentencia SQL debería usar el empleado para cumplir con este requerimiento?**



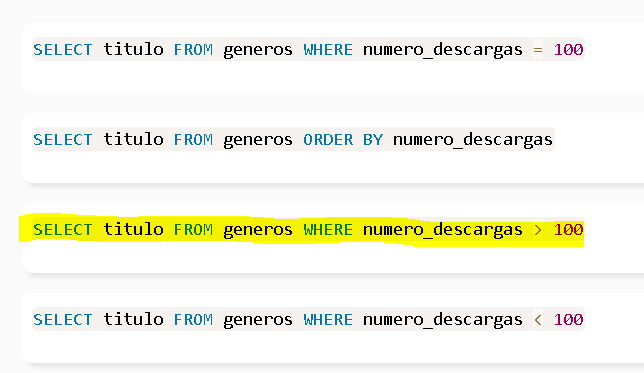
* 1. Actividad Interactiva 4

**Prueba tus conocimientos**

En la tienda MiVideojuego se tiene un sistema para el control de las ventas, el cual incluye la tabla generos la cual contiene todos los géneros de videojuegos disponibles en la tienda. Los campos y tipos de la tabla se muestran a continuación:

  
El encargado de la tienda virtual le ha pedido a uno de los miembros del staff que le proporcione una lista de todos los géneros de los videojuegos disponibles en la tienda que tienen más de 100 descargas

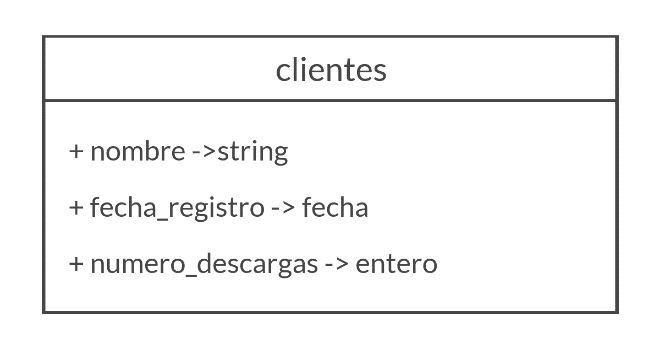
**¿Qué sentencia SQL debería usar el empleado para cumplir con este requerimiento?**



* 1. Actividad Interactiva 5

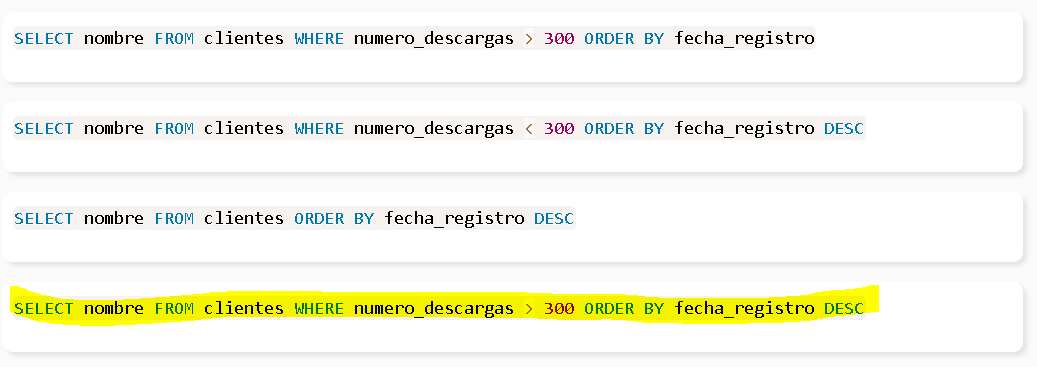
**Prueba tus conocimientos**

En el sistema para el control de ventas de la tienda MiVideojuego, se tiene una tabla con todos los clientes que compran en la tienda. Los campos y tipos de la tabla se muestran a continuación:



El encargado de la tienda virtual le ha pedido a uno de los miembros del staff que le proporcione una lista de todos los nombres de los clientes de la tienda, que tengan un número de descargas mayor a 300, ordenados descendentemente por fecha de registro.

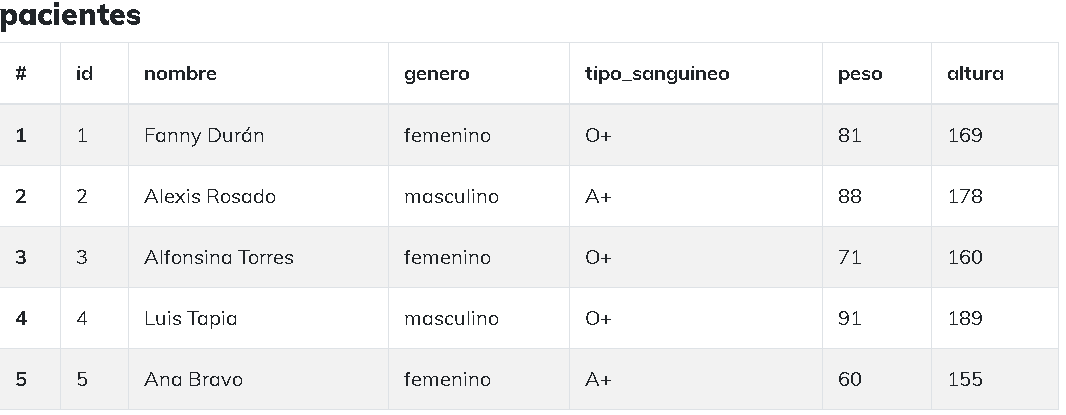
**¿Qué sentencia SQL debería usar el empleado para cumplir con este requerimiento?**



* 1. Ejercicio

**Sandbox**

Dada la tabla paciente con los valores iniciales



Realizar las consultas SELECT para cumplir los siguientes requerimientos:

1. Consultar todos los datos de las pacientes mujeres

Select \* from pacientes where genero like “femenino”

1. Consultar todos los datos de los pacientes mujeres ordenados por altura

Select \* from pacientes where genero like ‘femenino’ order by altura

1. Consultar todos los datos de las pacientes mujeres con tipo sanguíneo O+

Select \* from pacientes where genero like ‘femenino’ and tipo\_sanguineo like ‘O+’

1. Consultar todos los datos de las pacientes mujeres con tipo sanguíneo O+ ordenados por altura descendente

Select \* from pacientes where genero like ‘femenino’ and tipo\_sanguineo like ‘O+’ order by altura desc

1. Consultar el nombre de los pacientes con tipo sanguíneo O+

Select nombre from pacientes where tipo\_sanguineo like ‘O+’

1. Consultar el nombre y la altura de los pacientes hombres con tipo sanguíneo O+

Select nombre,altura from pacientes where genero like ‘masculino’ and tipo\_sanguineo like ‘O+’

1. Consultar el nombre y la altura de los pacientes hombres con tipo sanguíneo O+ y altura mayor que 180

Select nombre, altura from pacientes where genero like ‘masculino’ and tipo\_sanguineo like ‘O+’ and altura > 180

3.9. ¿Sabías qué?

**Breve historia de SQL**



El lenguaje de consulta estructurado recibe su nombre de las siglas en inglés SQL (Structured Query Language).



Las personas no suelen pronunciar SQL como las letras que lo conforman, sino que lo pronuncian como el precursor del mismo que se llama SEQUEL.



SQL fue originalmente creado por la empresa IBM como un sublenguaje de acceso a datos.



Sus primeras implementaciones comienzan a mediados de los años 70.



En 1987 se transformó en estándar ISO de la American National Standards Institute (ANSI).



Ha tenido varios problemas legales de nombre desde su implementación hasta que se definió como SQL.



Aunque en teoría es estándar, cada manejador de base de datos lo extiende con pequeñas variantes.



Ha tenido muchas versiones desde entonces, siendo las más usadas SQL 2000 y SQL 2008.

1. Lección 3: Más SQL
   1. [SQL Sentencias INSERT y DELETE](https://learn.nextu.com/mod/lesson/view.php?id=9660&pid=P_WEB_DATABASE)

INSERT INTO [tabla] VALUES (,) o

INSERT INTO [tabla] ([campo1, campo2]) VALUES (,)

DELETE FROM [tabla] WHERE [condiciones]

* 1. Ejercicio Práctico 1

**Sandbox**

Realizar las sentencias para cumplir los siguientes requerimientos:

* 1. Insertar el registro con id: 14, perteneciente a "Mario Tapia", masculino, con tipo sanguíneo O+, peso 91 y altura 189

INSERT INTO pacientes VALUES (14,"Mario Tapia","masculino","O+",91,155)

* 1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes

SELECT \* from pacientes

* 1. Insertar el registro con id: 15, perteneciente a "Ana Tapia", femenino, con tipo sanguíneo A+, peso 60 y altura 155

INSERT INTO pacientes VALUES (15,"Ana Tapia","femenino","A+",60,155)

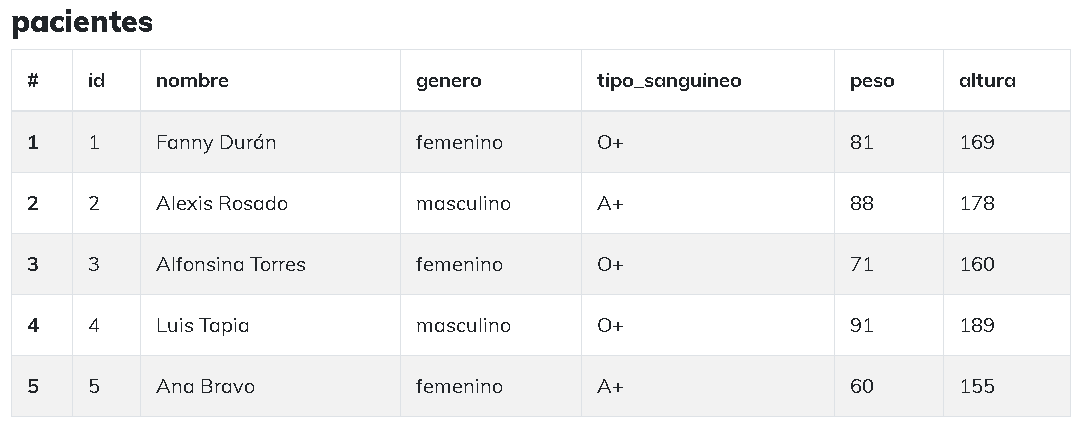
* 1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes

SELECT \* from pacientes

* 1. Ejercicio Práctico 2

**Sandbox**

Dada la tabla pacientes con los valores iniciales.



Realizar las sentencias para cumplir los siguientes requerimientos:

Tip: Para referirse a una fila en particular se suele usar el valor del campo `id` como condición

1. Eliminar al paciente Fanny Durán

DELETE FROM pacientes WHERE id=1

1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes para ver los cambios

SELECT \* FROM pacientes

1. Eliminar a todos los pacientes con tipo sanguíneo O+

DELETE FROM pacientes WHERE tipo\_sanguineo = "O+"

1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes para ver los cambios

SELECT \* FROM pacientes

1. Eliminar a todos los pacientes

DELETE FROM pacientes

1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes para ver los cambios

SELECT \* FROM pacientes

* 1. Sentencia UPDATE

2 formas de utilizar la sentencia update

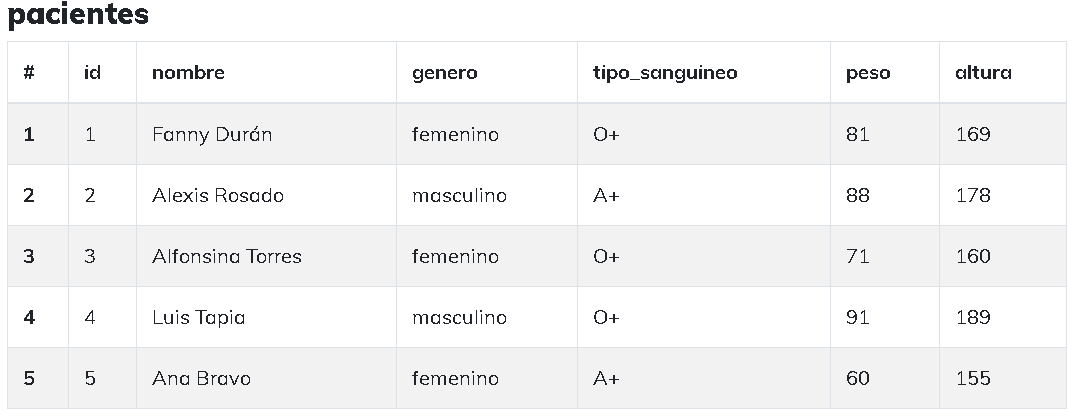
UPDATE [tabla] SET [campo=valor,]

UPDATE [tabla] SET [campo=valor,] WHERE condiciones

* 1. Ejercicio Práctico 3

**Sandbox**

Dada la tabla pacientes con los valores iniciales.



Realizar las sentencias para cumplir los siguientes requerimientos:

Tip: Para refererse a una fila en particular se suele usar el valor del campo `id` como condición

1. Actualizar el peso al Fanny Duran a 72

UPDATE pacientes SET peso=72 WHERE nombre ="Fanny Durán"

1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes para ver los cambios

SELECT \* from pacientes

1. Actualizar la altura y el peso al Luis Tapia Duran a 180, 85 respectivamente

UPDATE pacientes SET altura=180, peso=85 WHERE nombre="Luis Tapia"

1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes para ver los cambios

SELECT \* from pacientes

1. Actualizar el tipo sanguíneo a todos los pacientes a O+

UPDATE pacientes SET tipo\_sanguineo="O+"

1. Consultar la tabla actual con la sentencia SELECT \* FROM pacientes para ver los cambios

SELECT \* from pacientes

* 1. ¿Sabías qué?

**Resumen de Comandos SQL**

En esta lectura conseguirás rápidamente los comandos iniciales para trabajar con SQL en sus operaciones básicas (SELECT, INSERT, DELETE, UPDATE).

Recuerda que las palabras encerradas en corchetes [ ] son nombres de tus elementos: tablas, columnas, etc.

**SELECT (Para consultar tablas):**

SELECT [campos] FROM [tabla]

SELECT [campos] FROM [tabla] WHERE [condiciones]

SELECT [campos] FROM [tabla] WHERE [condiciones] ORDER BY [campos y orden(DESC / ASC)]

**INSERT (Para insertar en una tabla):**

INSERT INTO [tabla] VALUES (,) INSERT INTO [tabla] ([campo1], [campo2]) VALUES (,)

**DELETE (Para borrar filas):**

DELETE FROM [tabla] DELETE FROM [tabla] WHERE [condiciones]

**UPDATE (Para actualizar):**

UPDATE [table] SET [campo] = [valor],[campo] = [valor]...

UPDATE [table] SET [campo = valor] WHERE [condiciones]

Esperamos que esta ayuda sea de utilidad para ti en el futuro. Recuerda que estás no son las únicas formas de utilizar esos comandos, sin embargo, son las formas más utilizadas. Conoceremos más variantes de estos comandos próximamente.

1. Lección 4: Bases de Datos No relacionales
   1. [Bases de Datos No Relacionales JSON](https://learn.nextu.com/mod/lesson/view.php?id=9661&pid=P_WEB_DATABASE)

{JSON} 🡪 JavaScript Object Notation

Es un estándar de formato de texto, se usa para describir objetos basados en JS y es soportado por la mayoría de los lenguajes de programación. Pueden almacenar o recuperar un documento en formato JSON, incluso extraer los valores que tienen dicho documento. Está compuesto por 3 elementos:

* Valores – Valores simples como números, strings, null
* Objetos – Hash, diccionario, arreglo asociativo
* Arreglos – Listas o arreglos

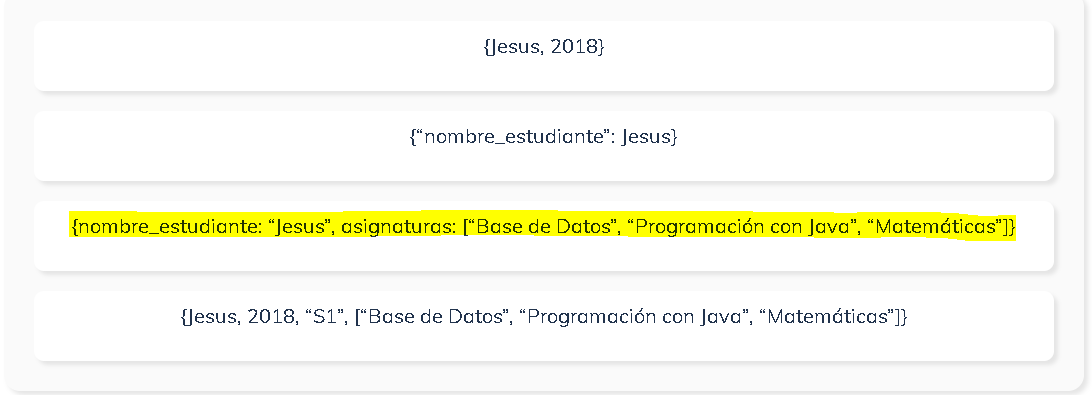
Ejemplo:



* 1. Actividad Interactiva 1

Prueba tus conocimientos

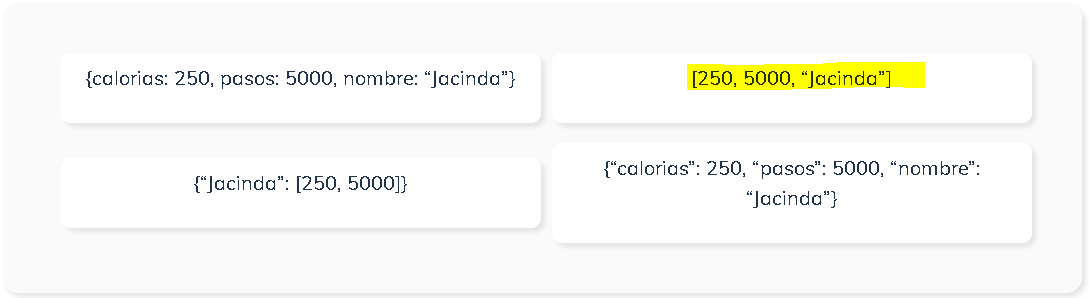
¿Cuál de las siguientes estructuras de un valor de JSON sería la correcta para representar las asignaturas que cursa un estudiante en



* 1. Actividad Interactiva 2

Prueba tus conocimientos

¿Cuál de las siguientes estructuras es un arreglo JSON?



* 1. ¿Sabías qué?

**Cápsula de conocimiento**

Documentos JSON: Embebido versus Por Referencias

En las Bases de Datos No Relacionales Orientadas a Documentos podemos tener documentos dentro de otros documentos. Esto hace a este tipo de base de datos particularmente interesante a la hora de modelar situaciones donde los datos se pueden organizar de forma jerárquica. Por ejemplo, los Blogs usualmente se estructuran de esta manera, ya que un blog contiene publicaciones y las publicaciones tienen comentarios que la audiencia realiza. Este caso se ajusta a nuestra visión de estructura jerárquica.  
Los documentos JSON tienen dos formas de permitir que los documentos estén definidos dentro de otros documentos: el enfoque embebido y el enfoque por referencias. En el enfoque embebido un documento se encuentra dentro de otro; mientras que en el enfoque por referencias los documentos se encuentran separados y se usan referencias para establecer relaciones entre ellos.  
Aunque no existe un consenso en cuanto a cuándo usar un enfoque o el otro, sí hay un conjunto de buenas prácticas, las cuales pueden ayudar a determinar el método a usar en un contexto y situación particular.

* 1. **Enfoque Embebido:**  
       
     + **Relaciones 1:N :** El enfoque embebido se ajusta adecuadamente a escenarios donde la relación entre entidades es de 1:N, es decir, una entidad puede tener asociadas muchas otras entidades, pero no viceversa. Este es el caso de las publicaciones de un Blog, donde una publicación puede contener muchos comentarios, pero no al contrario. El ejemplo a continuación muestra la aplicación del enfoque embebido para representar las publicaciones y comentarios.

publicaciones: [{ id: 101, titulo: "Título de publicación 1", contenido: "Este es el contenido ...", comentarios: [{ fecha\_hora: "2018-05-10 05:00:00", comentario: "Muy bien", }, { fecha\_hora: "2018-05-11 05:00:00", comentario: "Perfecto", }, ] }, ... ]

JSON

* + - **Documento Interno Asociado a un Solo Documento:**otra situación donde el enfoque embebido es útil, es cuando un documento interno es solamente usado por el documento que lo contiene. En el ejemplo del Blog, el documento comentario nada más se usa en publicaciones siendo conveniente representarlo usando este enfoque, ya que los cambios al documento comentario no deben realizarse en ningún otro lado.
    - **Recuperando el Documento como un Todo:**Conviene también usar el enfoque embebido cuando se debe recuperar el documento interno junto al que lo contiene. Por ejemplo, cuando consultamos las publicaciones es necesario recuperar también los comentarios porque, por ejemplo, así lo requiere la interfaz con el usuario.
    - **Pocos cambios en el Documento Embebido:** Si el documento embebido en otro no se actualiza frecuentemente, se puede usar el enfoque embebido; es decir cuando hay pocos comentarios para una publicación. De lo contrario sería ineficiente recuperar el documento completo debido a su tamaño.
    - **Documento Embebido es Pequeño:** El enfoque embebido se debería usar si el documento embebido es pequeño. Para objetos de gran tamaño, este enfoque puede resultar en un alto consumo de espacio de memoria.

1. **Enfoque por referencias:**  
     
   * **Relaciones N:M:** El enfoque por referencias se ajusta adecuadamente a situaciones donde la relación entre entidades es de N:M, es decir, una entidad puede tener asociadas muchas otras entidades y esas otra entidades pueden estar asociadas a una entidad. Si en nuestro ejemplo del Blog consideramos las relaciones entre publicaciones y los autores de las publicaciones, veremos que una publicación puede tener muchos autores y un autor puede escribir muchas publicaciones.
   * **Múltiples Documentos usan un Documento Interno:**En el caso que explicamos anteriormente, donde un comentario es usado por publicaciones y usuarios, es conveniente emplear el enfoque por referencia para evitar el problema de tener datos consistentes, que mencionamos en el caso de aplicar el enfoque embebido. El siguiente es un ejemplo de las Publicaciones en un Blog usando el Enfoque por Referencias, en el cual cada documento es un objeto separado y las relaciones se establecen a través del uso de referencias, solo necesitamos actualizar el documento usado por otros una sola vez.

publicaciones: [{ id: 101, ... autores: [{ id: 1001, nombre: "Luis", ... },{ id: 1002, nombre: "Maria", }, ... ] }, ... ] autores: [{ id: 1001, nombre: "Luis" ... publicaciones: [{ id: 101, titulo: "...", ... },{ id: 104, titulo: "...", ... }, ... ] }, ... ]

**JSON**

* + **Documento Interno es Opcional:**El enfoque por referencia permite solo consultar el documento requerido sin necesidad de tener que recuperar los documentos internos. Por ejemplo, se puede querer realizar una consulta a los datos de un usuario, por lo tanto no es requerido traer la información de los comentarios asociados a él.
  + **Actualizaciones Frecuentes en el Documento Interno:**Cuando el documento interno se actualiza con frecuencia se puede usar el enfoque por referencias ya que, a diferencia del enfoque embebido, solo hay que modificar al documento interno.
  + **Documento Interno tiene un Gran Tamaño:**Cuando el documento interno es de gran tamaño conviene usar el enfoque por referencias, para evitar tener que recuperar el documento interno y todas sus partes. En este caso, solo hay que traer el documento interno que se desea actualizar.

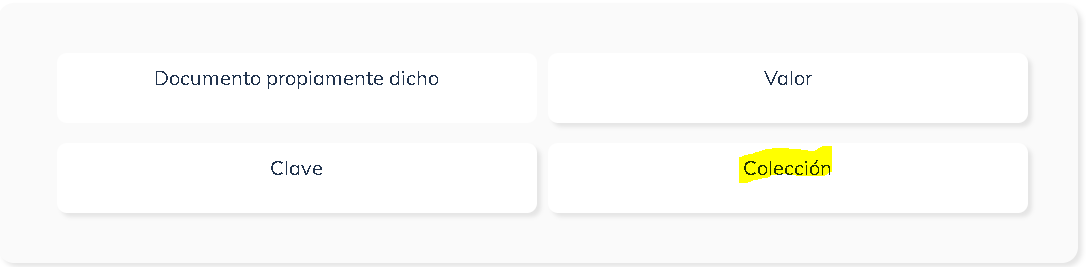
En la siguiente tabla se resumen las ventajas y desventajas de ambos los enfoques embebido y por referencia:

|  |  |  |
| --- | --- | --- |
| **Enfoque** | **Ventajas** | **Desventajas** |
| Embebido | * No es necesario ir a otra colección a buscar un documento. * Es útil cuando los datos se pueden organizar jerárquicamente. | * Usa más memoria al recuperar el documento “padre” completo incluyendo los documentos embebidos. * El documento padre puede tener muchos documentos embebidos haciendo la consulta ineficiente dado el alto volumen de datos. * El documento es usado en muchos otros documentos lo que conlleva tener información duplicada, y se dificulta mantener la consistencia de los datos. |
| Por referencias | * No es necesario actualizar los documentos múltiples veces. * No hay que recuperar los elementos referenciados siempre, por lo que se requiere menos espacio de memoria para almacenar los documentos. | * Las consultas requieren mayor tiempo para ser procesadas. * Es necesario establecer las jerarquías a través del uso de referencias. * Las consultas y actualizaciones suelen ser más lentas debido al número de referencias a otros elementos |

* 1. Actividad Interactiva 3

**Prueba tus conocimientos**

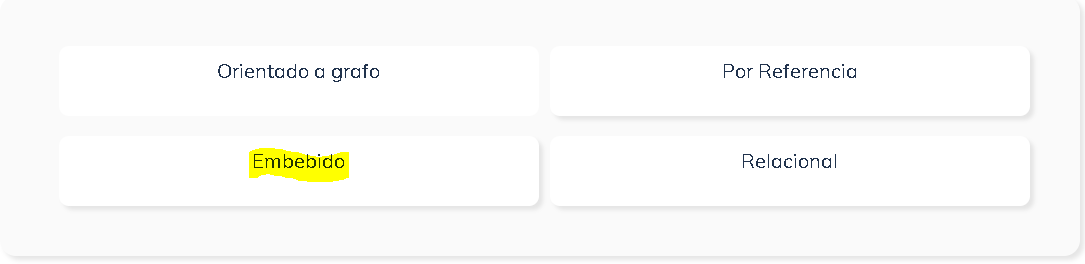
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ es una agrupación de varios documentos JSON en MongoDB.



* 1. Actividad Interactiva 4

**Prueba tus conocimientos**

En el enfoque \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ un documento JSON completo puede estar dentro de otro.



* 1. Ejercicio

**Desafío práctico**

Se ha creado una nueva red social que se llama `Cara de Libro`, esta red permite a usuarios crear portadas alternativas a libros ya conocidos. Tenemos los usuarios que se registran colocando su email, password y nombre.

El sistema tiene una lista de libros a los cuales se les desea diseñar una nueva portada. De cada libro se muestra, el nombre, autores y una fecha límite para la decisión de la nueva portada.

Cada usuario puede proponer una portada alternativa a esos libros indicando el "url de la imagen de la portada alternativa" como un string  y la fecha de publicación. Los usuarios pueden ver las alternativas de portadas, y darle un puntaje a cada una con un valor entre 0 y 10, donde 0 significa desagradable, y 10 significa me encanta.

**Define documentos JSON que se deben almacenar en esta situación para los siguientes valores:**

* Se tienen los siguientes usuarios:
  + Luis Granados (lg@gmail.com), password: 12345678
  + María Granados (mg@gmail.com), password: 12345678

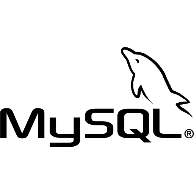
* Se tienen los siguientes libros:
  + El señor de los diamantes, Bob Fisher, fecha límite: 2019-01-01
  + Carpintería Fácil, Mónica Santos, fecha límite: 2019-02-01

* Luis agregó dos portadas con los urls ("*http://save.com/portada\_luis1.jpeg*", "*http://save.com/portada\_luis2.jpeg*") al libro de Carpintería el día (2018-05-01).
* María agregó una portada el día (2018-06-10) al libro de Carpintería con e l url ("*http://save.com/portada\_maria1.jpeg*")
* Hasta el momento se tienen 10 puntos para el url "*http://save.com/portada\_luis1.jpeg*", 8 para el url "*http://save.com/portada\_maria1.jpeg*", y 0 para el url "*http://save.com/portada\_luis2.jpeg*".
  1. ¿Sabías qué?

**Cápsula de conocimiento**

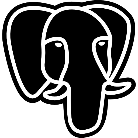
# **Base de Datos Relacionales y JSON**

Hemos visto distintos tipos de paradigmas relacionados con las Bases de Datos. Las Bases de Datos Relacionales guardan los datos en tablas. Las Bases de Datos NoSQL, en específico las orientadas a documentos JSON, guardan sus datos en Documentos JSON. Es interesante saber que algunas Bases de Datos Relacionales pueden almacenar en sus columnas datos JSON, ya que existe en ellas, el tipo JSON. El soporte para documentos JSON, son funcionalidades extendidas y relativamente nuevas para permitir guardar documentos en estas Bases de Datos Relacionales, cuando se tiene un JSON que ya forma parte del sistema. Además del soporte para documentos JSON, se pueden guardar otros tipos por ejemplo XML.  
Veamos un resumen de las características de algunos manejadores de Bases de Datos conocidos:



**MySQL**

MySQL soporta el tipo JSON actualmente para sus columnas. Al definir una columna como JSON, se obtienen ciertos beneficios como: Validación del documento (si no está bien formado se devuelve un error), Eficiencia de almacenamiento, entre otros.



**PostgreSQL**

PostgreSQL por su lado también ofrece soporte para el tipo JSON como columna. Esto permite recuperar y acceder a los elementos de un JSON previamente almacenado.

Además de MySQL y PostgreSQL, otros manejadores de base de datos relacionales soportan documentos JSON como tipo y tienen funciones que permiten recuperar valores y agregar nuevos datos, la forma de acceder a estos no es igual que las bases de datos NoSQL orientada a documentos JSON y su eficiencia tampoco es igual.

6. UNIDAD 2: Prueba

|  |  |
| --- | --- |
| **Comenzado en** | Tuesday, 12 de January de 2021, 11:38 |
| **Estado** | Terminados |
| **Finalizado en** | Tuesday, 12 de January de 2021, 11:44 |
| **Tiempo empleado** | 6 mins 24 segundos |
| **Puntos** | 10/10 |
| **Calificación** | **100** de un total de 100 |

Top of Form

### Pregunta 1

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

Deseamos insertar una nueva mascota a  la Tabla mascotas que tiene los campos: mascota\_id (clave), nombre, tipo, raza y fecha\_de\_nacimiento; la mascota es un gato cuyo nombre es kitty, su raza es persian y sus fecha de nacimiento es el 30 de Marzo de 2018. ¿Cuál de la siguientes sentencias debes utilizar?.

Seleccione una:

a. SELECT nombre FROM mascotas

b. UPDATE mascotas SET nombre = “kitty”

c. DELETE FROM mascotas WHERE nombre = “kitty”

d. INSERT INTO mascotas (nombre,tipo,raza,fecha\_de\_nactimiento) VALUES ( “kitty”, “gato”, “persian”, “2018-03-30”)

#### **Retroalimentación**

Esa es la sentencia qué inserta con los datos requeridos.

La respuesta correcta es: INSERT INTO mascotas (nombre,tipo,raza,fecha\_de\_nactimiento) VALUES ( “kitty”, “gato”, “persian”, “2018-03-30”)

### Pregunta 2

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

Victoria abrió una academia de baile en la ciudad; para controlar sus coreografías y coreógrafos le han recomendado usar una base de datos no relacional orientada a documentos JSON. Se sabe que una coreografía puede ser creada por varios coreógrafos y un coreógrafo puede crear más de un coreografía. Basado en las mejores prácticas,  ¿qué enfoque para el manejo de los documentos JSON deberían usar los diseñadores de la base de datos?

Seleccione una:

a. Embebido.

b. Por Referencia.

c. Integrado.

d. Consistente.

#### **Retroalimentación**

Por referencia debido al tipo de asociaciones qué se tienen.

La respuesta correcta es: Por Referencia.

### Pregunta 3

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

Una profesora desea saber el titulo de los reportes  entregados antes de la fecha tope de entrega, la cual era el 30 de Marzo de 2018. La tabla donde se almacenan los reportes se llama asignaciones y tiene lo siguientes campos: id\_asignacion (clave), titulo, fecha\_de\_entrega, calificacion, id\_estudiante (clave foránea). ¿Cuál de la siguientes sentencias debes utilizar?

Seleccione una:

a. SELECT titulo FROM asignaciones WHERE fecha\_de\_entrega <= “2018-03-30”

b. UPDATE asignaciones SET fecha\_de\_entrega = “30-03-2018”

c. DELETE FROM asignaciones WHERE fecha\_de\_entrega = “30-03-2018”

d. SELECT titulo FROM asignaciones

#### **Retroalimentación**

Esa es la sentencia qué se necesita con condición en la fecha

La respuesta correcta es: SELECT titulo FROM asignaciones WHERE fecha\_de\_entrega <= “2018-03-30”

### Pregunta 4

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

En el siguiente ejemplo, ¿qué tipo de base de datos estamos usando? Los datos de Estudiante se representan con un clave, por ejemplo, estudiante1, y un valor, por ejemplo: (nombre: “Geraldine”, asignatura: “Bases de Datos I”, semestre: “S1”).

Seleccione una:

a. Relacional.

b. No Relacional Orientado a Documentos.

c. No Relacional Orientado a Grafo.

d. No Relacional Orientado a Clave/Valor.

#### **Retroalimentación**

Este es el paradigma de esa representación.

La respuesta correcta es: No Relacional Orientado a Clave/Valor.

### Pregunta 5

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_es un tipo de datos en un documento JSON que se forma a partir de la estructura varios nombre (del atributo): valor.

Seleccione una:

a. Un objeto.

b. Un arreglo.

c. Un string.

d. Un número.

#### **Retroalimentación**

Un objeto permite guardar esa estructura.

La respuesta correcta es: Un objeto.

### Pregunta 6

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ es la sentencia SQL qué debes utilizar para consultar todos los títulos de los libros y correspondientes autores  de la Tabla Libros.

Seleccione una:

a. SELECT titulo,autor FROM libros WHERE autor = “Maria Elena”

b. SELECT titulo,autor FROM libros

c. SELECT titulo FROM libros WHERE autor = “Maria Elena”

d. SELECT autor FROM libros

#### **Retroalimentación**

Esa es la sentencia correcta sin condiciones y con los campos correctos.

La respuesta correcta es: SELECT titulo,autor FROM libros

### Pregunta 7

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

Deseamos actualizar la Tabla mascotas que tiene los campos: mascota\_id (clave), nombre, tipo, raza y fecha\_de\_nacimiento,de forma tal de que el nombre del perro “dogi” se cambie a “doggi” ¿Cuál de la siguientes sentencias debes utilizar?. Asuma qué existe un único nombre por perro.

Seleccione una:

a. SELECT nombre FROM mascotas

b. UPDATE mascotas SET nombre = “doggi” WHERE nombre = “dogi”

c. DELETE FROM mascotas WHERE nombre = “doggi”

d. INSERT INTO mascotas (nombre = “doggi”, tipo = perro, raza= “labrador”, fecha\_de\_nacimiento= “30-03-2018”)

#### **Retroalimentación**

Esa es la sentencia qué actualiza de la manera adecuada.

La respuesta correcta es: UPDATE mascotas SET nombre = “doggi” WHERE nombre = “dogi”

### Pregunta 8

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿De qué tipo es la siguiente relación entre: “una tabla clientes de una escuela de baile” y “clases\_de\_baile”?, donde un cliente puede asistir a varias clases de baile y a una clase de baile pueden ir muchos clientes.

Seleccione una:

a. 1:N.

b. N:M.

c. N:1.

d. M:M.

#### **Retroalimentación**

La relación es del tipo todos con todos.

La respuesta correcta es: N:M.

### Pregunta 9

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

La siguiente relación entre: “una tabla estudiantes de una universidad” y “datos de la asignación (individual) del estudiante”, donde un estudiante puede realizar varias asignaciones pero una asignación sólo puede ser realizada por un estudiante, es del tipo:

Seleccione una:

a. 1:N.

b. N:M.

c. N:1.

d. M:M.

#### **Retroalimentación**

Esa es la asociación correcta, un estudiantes muchas asignaciones, una asignación pertenece a un estudiante.

La respuesta correcta es: 1:N.

### Pregunta 10

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

En el paradigma \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ las estructuras están bien definidas y los datos se almacenan en tablas que contienen atributos, usualmente llamados campos.

Seleccione una:

a. Relacional.

b. No Relacional Orientado a Documentos.

c. No Relacional Orientado a Grafo.

d. No Relacional Orientado a Clave/Valor.

#### **Retroalimentación**

Relacional ya que se definen las tablas.

La respuesta correcta es: Relacional.

Bottom of Form