**UNIDAD 4: Otros Manejadores NoSQL**

1. [UNIDAD 4: Introducción](https://learn.nextu.com/mod/page/view.php?id=10301&pid=P_WEB_DATABASE)

**Otros Manejadores NoSQL**

Las bases de datos NoSQL integran todas aquellas que no son relacionales, sin embargo, con el tiempo se han definido un conjunto de aspectos que las caracterizan. En esta ocasión comenzamos estudiando un manejador de base de datos NoSQL orientado a grafos llamado Neo4j, el cual se basa en la teoría de grafos, donde se definen nodos relacionados entre sí a través de arcos dirigidos que representan dichas relaciones. Luego de estudiar dicho manejador, exploramos una nueva tendencia en el mundo de las bases de datos denominada NewSQL, la cual toma lo mejor de los paradigmas relacionales y no relacionales. Después de haber estudiado una serie de manejadores no relacionales, es importante conocer cómo los mismos son usados durante el desarrollo de aplicaciones; así las dos últimas partes están dirigidas a mostrar cómo podemos interactuar con las bases de datos a través del uso de los lenguajes de programación, tales como: JavaScript/Node, Java, Python, PHP, entre otros. De las formas de interactuar, examinamos una en particular que se basa en el paradigma orientado a objetos, conocida como Object-Relational Mapping (ORM).

**Objetivos de aprendizaje**

1. Identificar las bases de datos NoSQL orientadas a grafos, Neo4j, con la finalidad de utilizarlas en la solución de problemas en escenarios que se acoplen con este paradigma.
2. Identificar las características de la nueva tendencia en base de datos, conocida como NewSQL, con la finalidad de explorar sus usos potenciales en un futuro cercano.
3. Interpretar las formas en que podemos interactuar con una base de datos  a través de diversos lenguajes de programación.
4. Interpretar las formas en que podemos interactuar con una base de datos  a través de diversos lenguajes de programación usando Object-Relational Mapping (ORM).
5. Lección 1: Neo4J
   1. [Neo4J](https://learn.nextu.com/mod/lesson/view.php?id=10302&pid=P_WEB_DATABASE)

¿Qué es?

Características:

* Es un proyecto que tiene una versión comercial, así como una versión de código abierto.
* Desarrollado en Java
* Contiene un API REST
* Escalabilidad horizontal y vertical, con una orientación a grandes volúmenes de datos en ciertas situaciones.

¿Qué es un grafo?

Los grafos constan de nodos y relaciones

Los nodos tienen propiedades (atributos de los datos que queremos guardar)

Las relaciones son conexiones entre nodos.

Grafo dirigido

Las relaciones también pueden tener propiedades, lo que significa que también podemos colocar datos en la relación.

También se tienen etiquetas que identifican un tipo de nodos 🡪 de esta forma se pueden agrupar los nodos

Características resaltantes:

* Posee representación nativa de nodos 🡪 un nodo puede consultar info de los nodos vecinos. (a través del uso de apuntadores en el Driver)
* Tiene una herramienta que nos permite ver gráficamente como están los nodos.
* Adicionalmente es posible exportar los nodos de forma tabular.
* Tiene un lenguaje propio que se llama Cypher

Con Cypher podemos interactuar con neo4j para:

* Crear
* Actualizar
* Eliminar
* Consultar
* Soporta operaciones más complejas como: ordenamiento, también se pueden agrupar los datos y realizar operaciones como la suma y el conteo.
* Se pueden realizar operaciones propias de los grafos, tales como: alcanzabilidad; camino más corto entre un nodo y otro; Relaciones que conectan un nodo con otro.

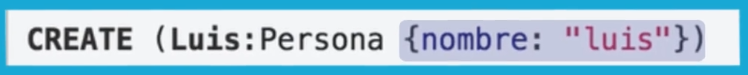
Ejemplo:



Se pueden colocar las propiedades que se quieran sin seguir una estructura predeterminada.

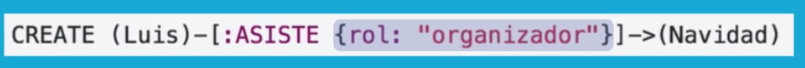
En el ejemplo anterior se creo un nodo de nombre “fiesta”.

Se puede crear otro nodo:

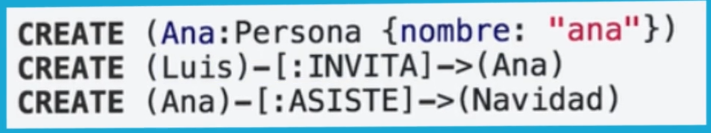


Neo4j crea nodos de diferentes colones ya que son diferente tipo.

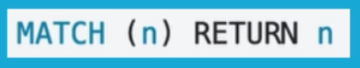
Se puede crear relaciones:



Otro nodo-relación podría ser:



Para retornar todos los nodos:

 🡪 se pueden poner condiciones en el match para filtrar las condiciones que queremos consultar en nuestra red de nodos.

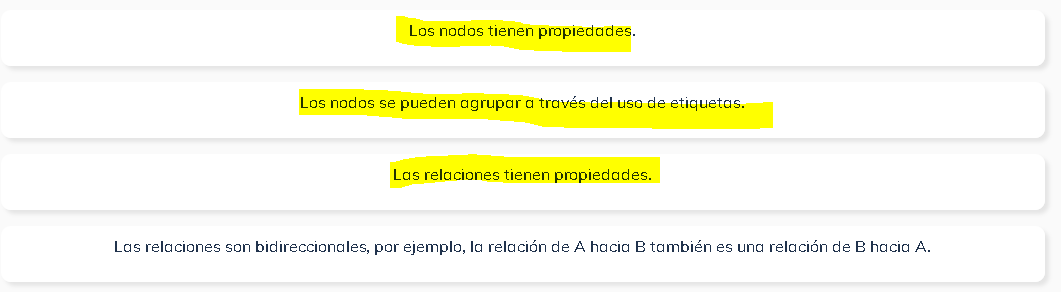
Por ejemplo, esto puede ser útil en las redes sociales.

No es eficiente para consultas globales, por el contrario es útil en situaciones donde tenemos que consultar datos locales

* 1. Actividad interactiva 1

**Prueba tus conocimientos**

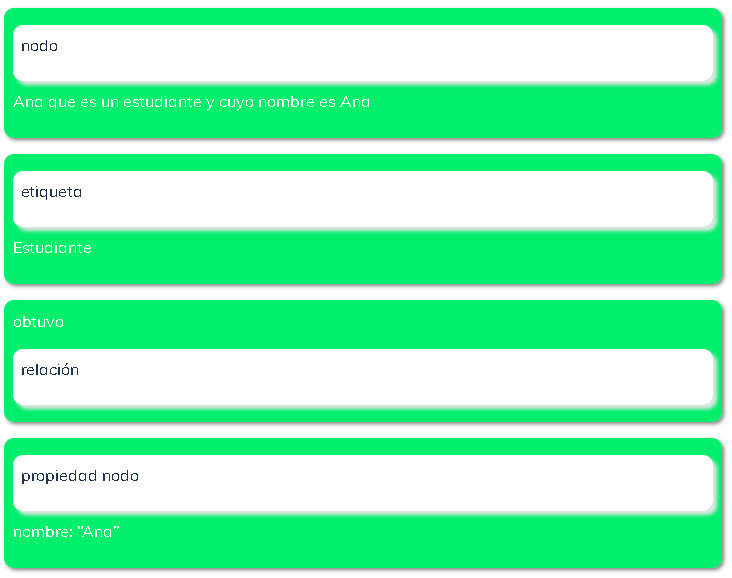
Neo4j es orientado a grafos y por lo tanto usa la teoría de grafos como su base fundamental. Sin embargo, hay ciertas peculiaridades propias del uso de grafos en Neo4j que deseamos que senales. Señala todas las que apliquen.



* 1. Actividad interactiva 2

**Analiza y relaciona**

Supongamos que queremos diseñar una base de datos de estudiantes y sus calificaciones obtenidas en las evaluaciones de este semestre. Ayúdanos a diseñar el grafo de Neo4j realizando la correspondencia entre los elementos de un grafo Neo4j y los valores para este ejemplo.



* 1. ¿Sabías qué?

**Cápsula de conocimiento**

**Lectura: Usos de Base de Datos Orientadas a Grafos**

Las bases de datos orientadas a grafos tienen unas características que las diferencian de las demás. Están orientadas a ser usadas para resolver problemas específicos. A continuación, deseamos proporcionarte un panorama más claro de cómo y cuándo podemos usar estas bases de datos.

*¿Cuándo usar una base de datos orientada a grafos?*

* Cuando los datos que se desean almacenar están relacionados de manera natural en grupos
* Los datos tienen sentido como nodos y relaciones
* No se hacen consultas globales
* Las consultas más comunes están centradas alrededor de una entidad.

*Ejemplos:*

1. Redes sociales
2. Juegos colaborativos
3. Sistemas de recomendaciones
4. Detección de fraude
5. Recomendaciones de producto

*¿Cuándo no usar una base de datos orientada a grafos?*

* En contraparte no se deben usar estas bases de datos cuando los datos no se pueden organizar naturalmente en forma de grafos, por lo que representarlos de esa manera sería muy complejo y engorroso.
* Cuando se hacen consultas globales.

*Ejemplos*

En estos ejemplos asumimos que los datos no se conectan de forma natural en estructura de grafos, y las consultas que se hacen son más orientadas a valores globales.

* Base de datos de usuarios
* Inventario
* Ventas

Recordemos que en las organizaciones se pueden usar sistemas que tengan varios tipos de bases de datos de acuerdo a su propósito. Esperamos que esta información sea de utilidad para ti.

1. Lección 2: NewSQL
   1. [New SQL](https://learn.nextu.com/mod/lesson/view.php?id=10303&pid=P_WEB_DATABASE)

Bases de Datos SQL 🡪 bien definidas

Bases de Datos NoSQL 🡪 no siguen el paradigma estructurado

Ventajas DB SQL:

* Son consistentes, ya que tienen restricciones, utilizan transacciones y tiene un lenguaje SQL.
* Utiliza un lenguaje estándar altamente usado SQL.

Desventajas DB SQL:

* Tiene una estructura predefinida, por lo que si la estructura de la DB está cambiando constantemente no es buena idea usar SQL.
* Otro problema es la escalabilidad, estas tecnologías fueron diseñadas en los años 80. Antes la escalabilidad se obtenía adquiriendo equipo más potente para procesar más transacciones por unidad de tiempo.
* De lo anterior, se puede decir que estas bases de datos soportan un acceso moderado, dependiendo del equipo que se tenga y a la cantidad de registro y operaciones que se realizan por minuto.

Ventajas DB NoSQL:

* Son flexibles, es decir podemos cambiar las estructuras de los datos almacenada a través del tiempo
* Son escalables, por lo que se puede tener alta disponibilidad, estas bases de datos funcionan con nodos, por lo que se si requiere aumentar la capacidad, lo que se requiere es agregar más nodos. Ya que ellos trabajan en conjunto y de manera distribuida, con el fin de proporcionar alta disponibilidad a los usuarios.

Desventajas DB NoSQL:

* No tienen tantas validaciones de consistencia y transacciones, aunque depende del lenguaje utilizado
* No tienen un lenguaje tan estandarizado como si lo tiene SQL.
* Cada una de las DB NoSQL tienen un paradigma distinto, acoplándose de una mejor manera a ciertas situaciones (propósito específico).

NewSQL 🡪 Trata de rescatar lo mejor de ambos paradigmas.

* Mantener transacciones y sea consistentes.
* Que soporte el lenguaje SQL y sea escalable

NewSQL se acopló en el 2011, sin embargo, existen nuevas generaciones de Db que cumplen con estas características.

Se han pensado con características:

* Distribuidas desde un comienzo. Desde el inicio se asume que tiene un clusters.
* Con tolerancia a fallas. Si un nodo se cae los demás pueden mantener el funcionamiento del sistema.
* Se pueden crear nodos bajo demanda mientas se mantiene el control de concurrencias.
* Los datos pueden estar distribuidos y las ejecuciones también.
* Utilizan técnicas para el manejo eficiente en memoria. Donde algunas de estas bases de datos mantienen los datos en memoria.

Ejemplos de manejadores NewSQL:

* CockroachDB
* GoogleSpanner
* TiDB 🡪 Capa que se le agrega a MySQL
* VoltDB
* Altibase
* nuoDB

Estudiaremos a VoltDB:

* El cual es un manejador que está en memoria para proporcionar alta disponibilidad y acceso de los datos de forma rápida.

Tiene una versión de código abierto, así como una pagada.

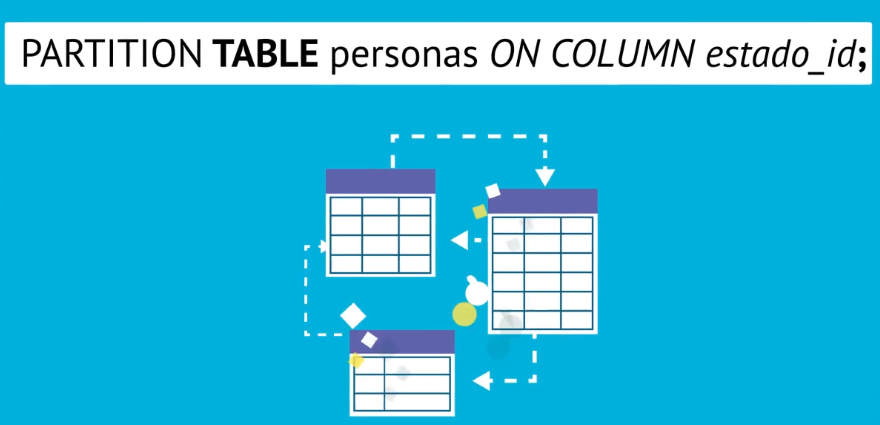
* Está escrito en Java pero el núcleo está escrito en C++
* Está basado en nodos.

Diferencias con manejadores SQL:

* No se habla de Base de datos sino de catálogos, donde los catálogos van a estar distribuidos en varios nodos.
* Tienen dos esquemas principales de distribución de los nodos.
* Los nodos son creados de forma tal que un nodo es independiente de otro. El sistema en sí esta diseñado de esta manera.
* Los nodos trabajan de forma tal que se pueda distribuir la carga entre todos ellos.
* Conceptos importantes: Particiones y Replicaciones:
  + Las particiones permiten dividir los datos bajo un criterio. Por ejemplo, columnas



Otro ejemplo puede ser las particiones por personas



También se puede crear replicaciones, por ejemplo:



* Así como realiza las particiones también sincroniza varias tablas en varios nodos. Es con el fin de que si un nodo se cae, existe otra copia en otro nodo que mantiene los datos en la tabla.
* Si queremos aumentar la capacidad podemos aumentar más nodos

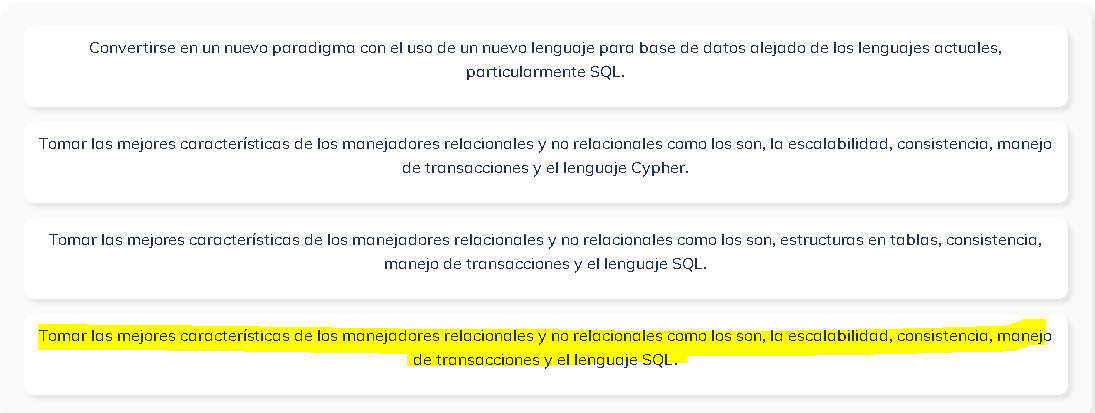


Por lo que esta Db puede ejecutar millones de operaciones por segundo o minuto, de acuerdo con la cantidad de nodos que se agreguen. Pero se traduce en una alta disponibilidad

* 1. [Actividad interactiva](https://learn.nextu.com/mod/lesson/view.php?id=10303&pageid=46921&pid=P_WEB_DATABASE)

**Prueba tus conocimientos**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ es el propósito de los manejadores NewSQL.



* 1. ¿Sabías qué?

**Cápsula de conocimiento**

**Lectura: Usos de Base de Datos NewSQL**

Las Bases de Datos NewSQL combinan las características de las Bases de Datos Relacionales y las Bases de Datos No Relacionales, lo que permite que se usen en un amplio rango de situaciones. A continuación, estudiaremos algunos casos donde dichas bases de datos pueden ser beneficiosas.

**¿Cuándo usar una Base de Datos NewSQL?**

Se pueden usar en cualquier sistema en general, pero están diseñadas para tener un alto rendimiento, por lo tanto, pueden ser de más utilidad en sistemas de alto tráfico.

Ejemplos

1. Sistemas bancarios
2. Redes sociales
3. Juegos online
4. Detección de fraude
5. Sistemas con datos en vivo

**¿Cuándo No usar una Base de Datos NewSQL?**

Estas bases de datos no deben usarse cuando existe una característica particular de los datos para lo cual otro enfoque de bases de datos sea más adecuado, por ejemplo, grafos, clave/valor, documentos.

Estas bases de datos tampoco deberían usarse cuando no se necesita un ambiente distribuido o los datos son tan simples que no se requiere un alto rendimiento.

Recordemos que en las organizaciones se pueden usar sistemas que tengan varias bases de datos de acuerdo con su propósito. Las Bases de Datos NewSQL están comenzando a ser adoptadas por la comunidad de programadores.

1. Lección 3: Interanctuando con los lenguajes de Programación
   1. [Lenguajes de programación y NoSQL](https://learn.nextu.com/mod/lesson/view.php?id=10304&pid=P_WEB_DATABASE)

Utilizan conectores. 🡪 Driver.

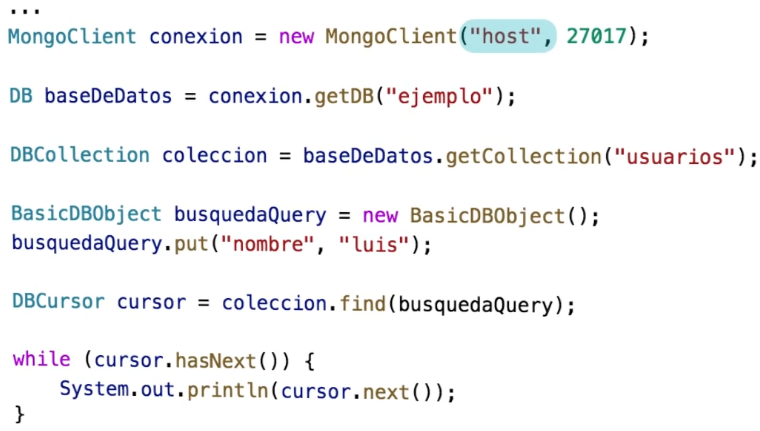
El driver actuará como un cliente.

Ejemplo: Mongo-Java

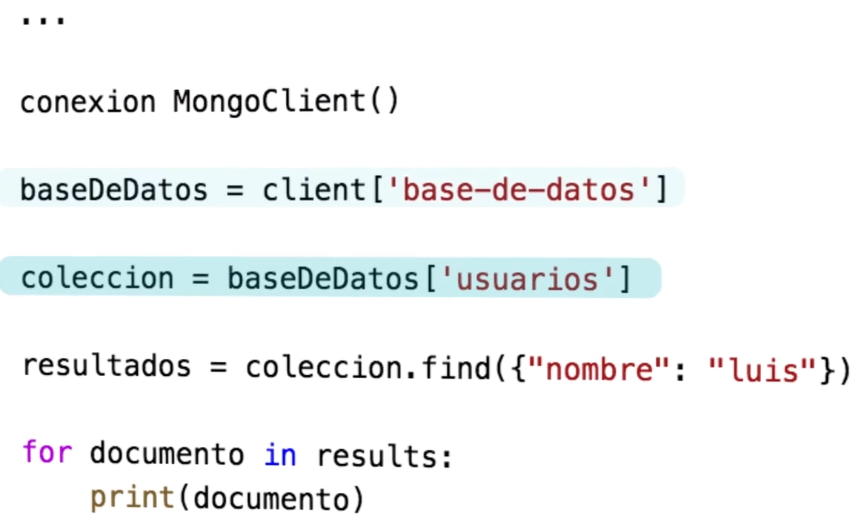
Se debe definir el puerto (puerto pretedeterminado de Mongo 27017) y el host (localhost o 127.0.0.1)

Luego se selecciona la DB y la colección.

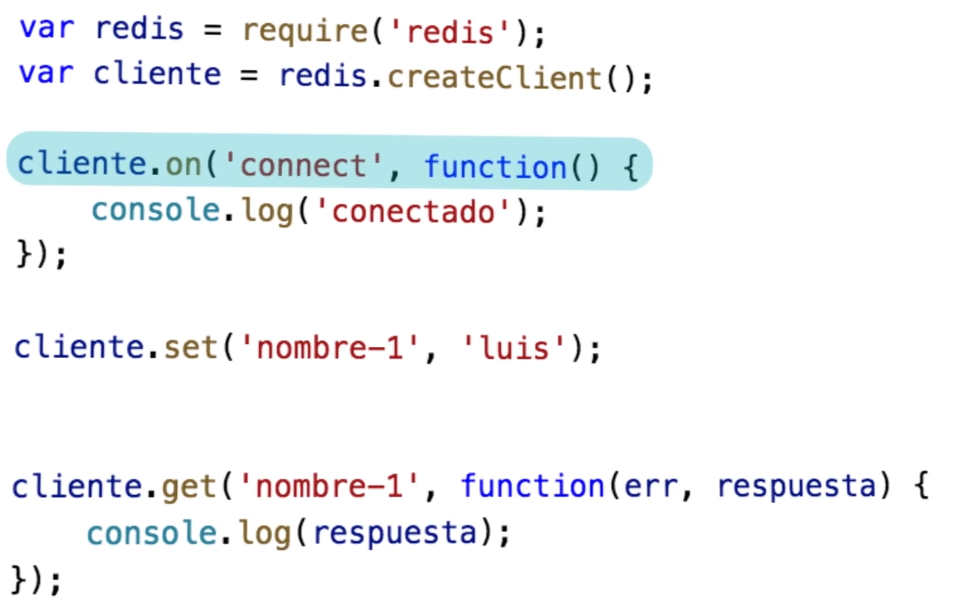
Ejemplo de Mongo-Java:



Ejemplo Mongo-Python:



Ejemplo Redis-JavascriptNode:



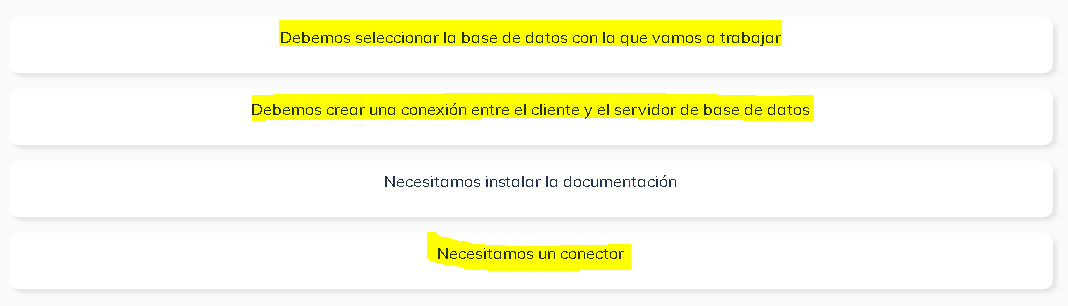
Ejemplo Redis-PHP:



* 1. Actividad interactiva 1

**Prueba tus conocimientos**

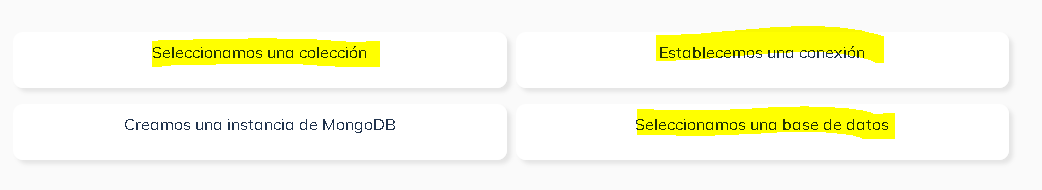
¿Cuáles son los aspectos generales a tomar en cuenta a la hora de trabajar con un manejador de base de datos desde un lenguaje de programación? Señala todas las que apliquen.



* 1. Actividad interactiva 2

**Prueba tus conocimientos**

¿Cuáles son algunos de los pasos generales para conectarse a MongoDB desde una aplicación que esté usando Java como lenguaje de programación? Señala todas las que apliquen.



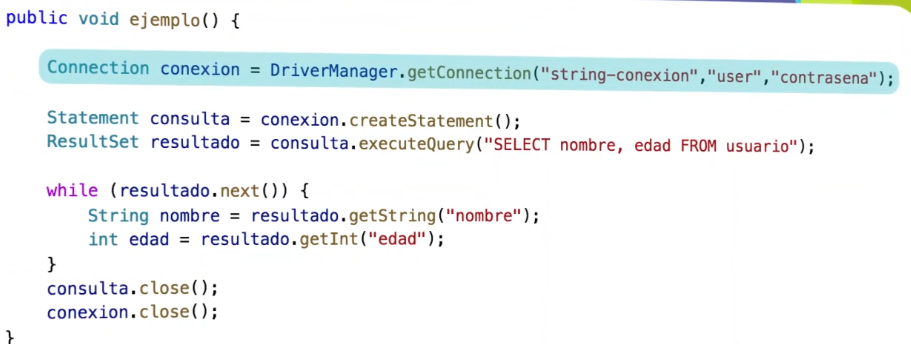
* 1. Lenguajes de programación y SQL

Interacción de SQL y Lenguaje de programación.

* Se crea una conexión especificando a quien se quiere conectar
* Se define una consulta SQL
* Se ejecuta la consulta
* Se cierra la conexión.

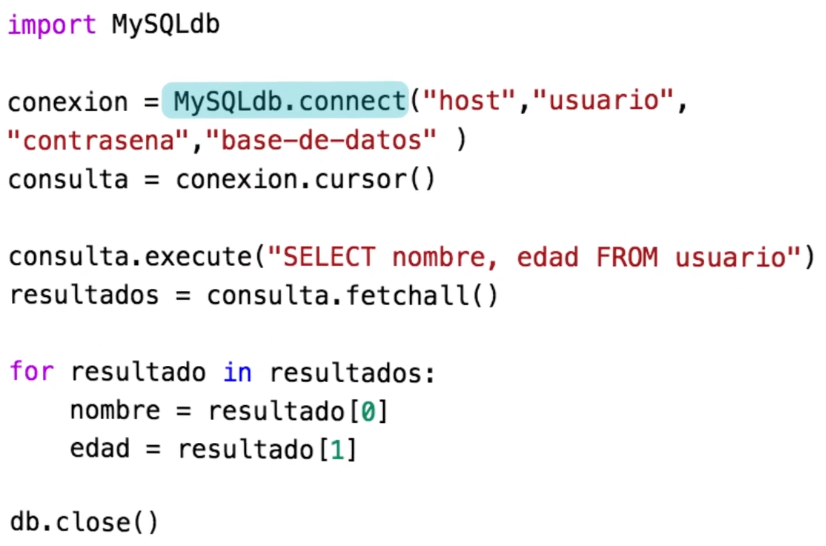
Sql-Java:

Se utiliza una librería llamada JDBC

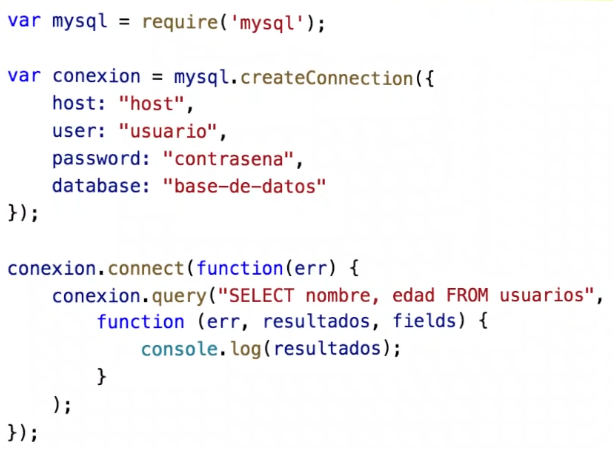


El string-conexión contiene el host o IP, y el puerto y la base de datos

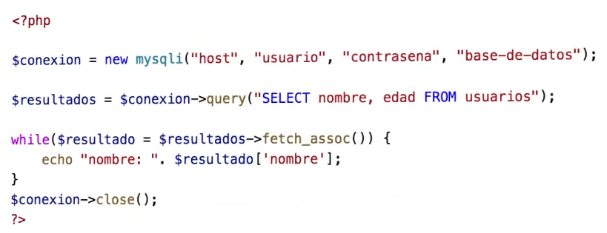
MySQL-Python:



SQL – JavaScript Node:



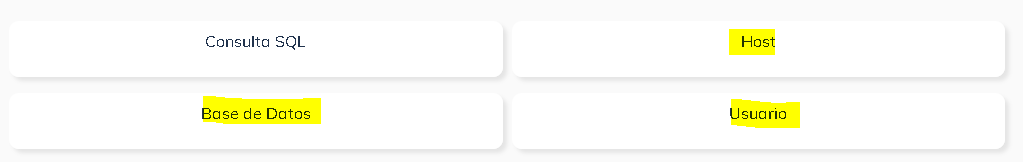
SQL-PHP:



* 1. Actividad interactiva 3

**Prueba tus conocimientos**

¿Cuáles son algunos de los parámetros que se utilizan para especificar dónde está la base de datos SQL a la cual queremos conectarnos independientemente del lenguaje de programación que estemos utilizando? Señala todas las que apliquen.



* 1. Actividad interactiva 4

**Prueba tus conocimientos**

¿Cuál es el problema del siguiente código Java usado para interactuar con el manejador SQL?

public void ejemplo() {

Connection conexion = DriverManager.getConnection("string-conexion","user","contrasena");

Statement consulta = conexion.createStatement();

while (resultado.next()) {

String nombre = resultado.getString("nombre");

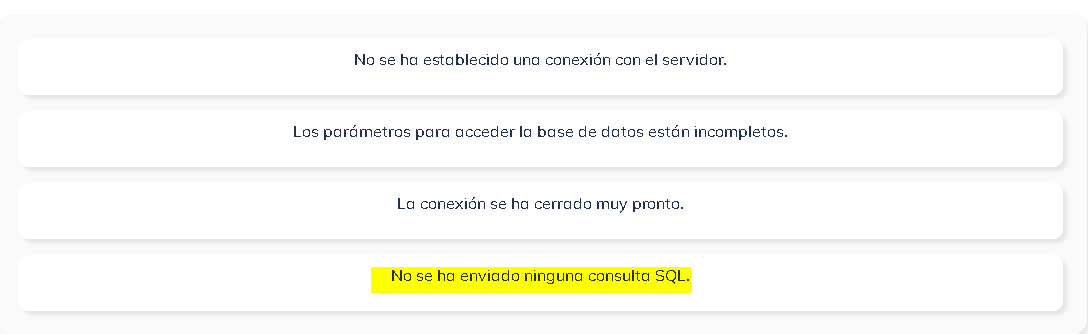
int edad = resultado.getInt("edad");

}

consulta.close();

conexion.close();

}



1. Lección 4: ORM
   1. [ORM operaciones básicas](https://learn.nextu.com/mod/lesson/view.php?id=10305&pid=P_WEB_DATABASE)

ORM – Object-Relational Mapping

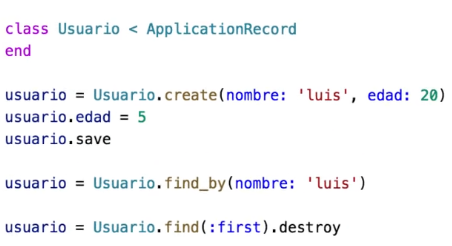
Correspondencia a lenguajes orientados a objetos como los son Java y C++

Se inicio para trabajarse con SQL, pero hoy en día también se usa con NoSQL

Funcionamiento:

* Se agrega una capa adicional implementada como una librería ORM.
* A la librería del conector de la Base de datos que estamos usando.
* Se debe configurar el ORM donde indiquemos donde está la base de datos y su nombre.
* La forma en como esta diseñado el ORM proporciona independencia del manejador.

Ejemplo ORM-Ruby:



* 1. Actividad interactiva 1
  2. Actividad interactiva 2
  3. ORM: Manipulando la base de datos
  4. Actividad interactiva 3
  5. Actividad interactiva 4
  6. Actividad interactiva 5

1. UNIDAD 4: Prueba