UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

Diplomatura en Bases de Datos

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 3: Herramientas de cara al futuro

Unidad 1: Bases de datos documentales

Presentación:

En esta Unidad descubrimos qué es una base de datos documental, para lo cual necesitamos entender los modelos JSON y del XML. Finalmente, recorremos los diversos tipos de bases documentales presentes en el mercado discutiendo sus características.

Objetivos:

Que los participantes:

- Conozcan entiendan qué es una base de datos documental.
- Identifiquen en qué problemas conviene aplicar esta tecnología.
- Aprendan a seleccionar entre las distintas herramientas presentes en el mercado.
- Se familiaricen con los usos básicos de una base de datos documental.

Bloques temáticos:

1. Descripción del concepto de base documental.

2. Algunas opciones de bases de datos documentales.

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

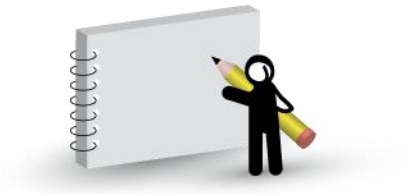
- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

**** El MEC es el modelo de E-learning colaborativo de nuestro Centro.***



Tomen nota:

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares.

Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación.

Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga.

Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas.

Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia.

Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Descripción del concepto de base documental

Una base de datos documental u orientada a documentos es un tipo de modelado de datos en las bases de datos NoSQL (no sólo SQL). Este modelo se basa en el paradigma "key-value" (valor-clave).

Nota conceptual:

Ya veremos en detalle más adelante las bases No SQL. Por ahora coniene terer presente que NoSQL implica que cumplen con la funcionalidad SQL y la superan.

En este caso, el valor es un documento de tipo JSON o XML. Dicho de otra manera una base de datos de documentos es una base de datos no relacional que almacena los datos como documentos estructurados, generalmente en XML o formatos JSON.
La definición de "base de datos de documentos" no implica nada más específico que modelo de almacenamiento de documentos: las bases de datos de documentos son libres de implementar transacciones ACID u otras características de una RDBMS tradicional, aunque las bases de datos de documentos dominantes proporcionan un apoyo modesto a las transacciones.

Las bases de datos de documentos basadas en JSON florecieron después de la ruptura no relacional de 2008, debido a tres razones principales:

- En primer lugar, abordan el conflicto entre la programación orientada a objetos y el modelo de base de datos relacional que había frustrado a los desarrolladores de software, y que motivó gran parte del movimiento a la orientación a objetos de base de datos de mediados de la década de 1990.

- Segundo, porque los formatos de documentos autodescriptivos podrían ser interrogados independientemente del programa que los creó, apoyando el acceso de consulta "ad-hoc"" a la base de datos que estuvo ausente en los almacenes de valores-clave puros.

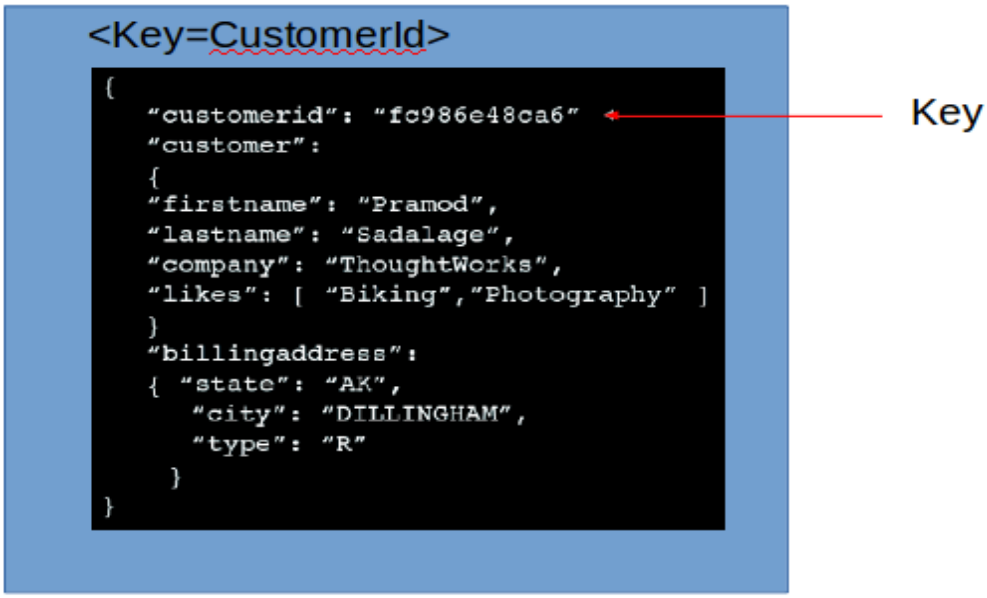
- En tercer lugar, se alinearon bien con los paradigmas de programación dominantes basados en la web, particularmente el modelo de programación AJAX. Una base de datos de documentos, al permitir alguna forma de descripción de datos sin

imponer un esquema, tal vez proporciona un medio feliz entre el esquema rígido de la base de datos relacional y el de almacenes de valores-clave sin esquema.

Los programadores son libres de cambiar el modelo de datos a medida que cambian los requisitos dentro de una aplicación, pero los consumidores de datos aún pueden interrogar los datos para determinar su significado.

La alineación con las prácticas de programación de desarrollo web ha resultado en bases de datos de documentos JSON, y la base de datos MongoDB en particular, se convierte en la opción predeterminada para muchos sitios web.

Entonces, con una clave podemos obtener un conjunto estructurado de información de la siguiente forma:



Para hacer la misma operación en un modelo relacional se deberían hacer muchos "JOIN" lo cual reduce notablemente el rendimiento. En otras palabras, una base de datos documental toma los datos que se desean almacenar y los agrega a los documentos utilizando, por ejemplo, el formato JSON (objeto JavaScript Notación).

Entonces en un solo documento podemos tener la misma información que generalmente se almacena en muchas tablas en un modelo relacional.

Así todo lo asociado con un registro individual está encapsulado en lo que se conoce como un documento.
Un documento generalmente está codificado en alguno de los formatos estándar mencionados como XML o JSON.

Los documentos son similares a las filas en una base de datos relacional, pero ellos proporcionan más flexibilidad. Cada documento puede compartir una estructura común, pero también se puede variar en los tipos de campos que contienen.

Este enfoque permite agregar nueva información a algunos documentos sin requerir que todos los documentos tengan la misma estructura.

Bases de datos NoSQL

Este concepto se creó en 1998 y la idea principal detrás de NoSQL es intentar hacer que las bases de datos sirvan de modelo sin usar SQL y relaciones tabulares como es el caso en el modelo relacional.

Estos dos tipos de modelos son muy diferentes no sólo en su estructura sino también en su forma de acceder a los datos.

Estos dos tipos de modelos son muy diferentes no sólo en su estructura sino también en su forma de acceder a los datos. De hecho, el modelo relacional toma los datos y los separa en muchas tablas interrelacionadas que contienen filas y columnas.

Las tablas se referencian entre sí a través de claves externas que están almacenadas en columnas. Y con esta estructura cuando se buscan datos, la información deseada debe recopilarse de muchas tablas y ser combinadas antes de que se pueda proporcionar a la aplicación.

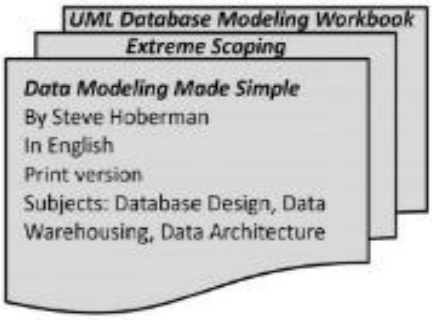
Por otro lado cuando tiene que escribir datos, la acción de escritura debe realizarse en muchas tablas. Hay muchas bases de datos NoSQL diferentes (distintas de las orientadas a documentos) y cada una de ellos tienen una estructura diferente, por ejemplo:

- Almacenes de gráficos que están diseñados para datos cuyas relaciones se pueden representar como un gráfico (elementos interconectado con un número indeterminado de relaciones entre ellos). Un ejemplo de tipo de datos que pueden ser representados por este tipo de modelo podrían ser las relaciones sociales, enlaces de transporte público, mapas de carreteras o topologías de red.

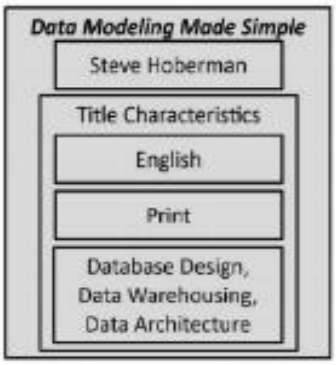
- Los almacenes de tipo key-value son muy simples porque básicamente se basan en un diccionario (mapa) donde los datos están representados por un par clave/valor (key/value).

- Los almacenes de columna ancha son similares a la tabla de relaciones, excepto que en una base de datos NoSQL el número de columna es dinámica en contraste con la tabla relacional donde las columnas se fijan en la creación de el esquema. Entonces, en una base de datos de almacenes de columna ancha cuando se quiere agregar un dato, la cantidad de la columna puede variar, evitando devolver la columna con valor NULL.

- El almacén de documentos puede verse similar al de columna ancha en el sentido de que comparten la forma sin esquema, pero la implementación es diferente. En una base de datos de documentos, cada registro se llama documento y se almacena por separado.



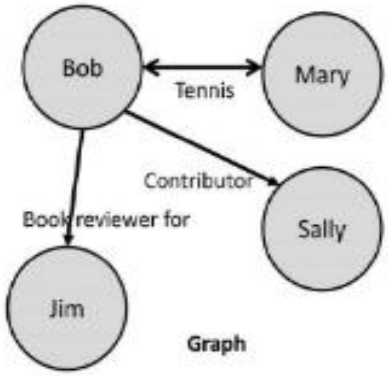
Document-oriented
(Library Card Catalog)



Column-oriented

Key	Value
Title Name	*Data Modeling Made Simple*
Author Name	Steve Hoberman
Language	English
Format	Print version
Subjects	Database Design, Data Warehousing, Data Architecture

Key-value



Graph

Steve Hoberman, 2014, "Data modellig for MondoDB"

¿Por qué elegir NoSQL?

Hay muchos argumentos que podemos proporcionar para respaldar la elección de elegir NoSQL. Como por ejemplo:

- **Performance:**
  Para escribir y leer en la base de datos, un almacén de documentos es más rápido que una base de datos SQL tradicional. Por ejemplo, MongoDB puede ser 20 veces más rápido que Postgres.

  Esto se debe a la naturaleza más simple del documento, la base de datos solo

necesita almacenar directamente el documento y para una lectura solo tiene que hacer una búsqueda de los datos mientras que para una consulta relacional, necesita decodificar la consulta y buscar en toda la tabla requerida y agregar el resultado.

Así, la simplicidad del documento es una ventaja real para el almacenamiento de datos orientado al rendimiento.

- **Modelo de datos más flexible:**
  Se debe a que nuestro modelo de datos carece de un esquema fijo.
  Una base de datos de documentos no usa un esquema fijo, sino que puede almacenar cualquier documento que necesite creado en tiempo de ejecución.

  Esto permite que la base de datos sea modular y flexible, así la aplicación puede almacenar todo lo que necesita y, en caso de que cambie la lógica comercial de un programa, la base de datos no necesita ningún cambio adicional.

  Cada documento contiene su propio esquema y una colección del documento puede tener un esquema diferente. Una única base de datos también puede funcionar para múltiples tipos de aplicaciones porque toda la lógica de datos está en el lado del cliente.

- **Mejor escalabilidad que un modelo relacional:**
  Muchas bases de datos NoSQL están diseñadas con escalabilidad y gran cantidad de accesos y el almacén de documentos no es una excepción.

  Debido a su naturaleza simple, permiten aumentar fácilmente la base de datos y dividirla entre diferentes servidores o nodos sin tener que aumentar el poder del servidor, esto permite que la base de datos se ejecute en hardware más simple para el mismo rendimiento y para mejorar la confiabilidad

Por qué elegir un almacén de documentos

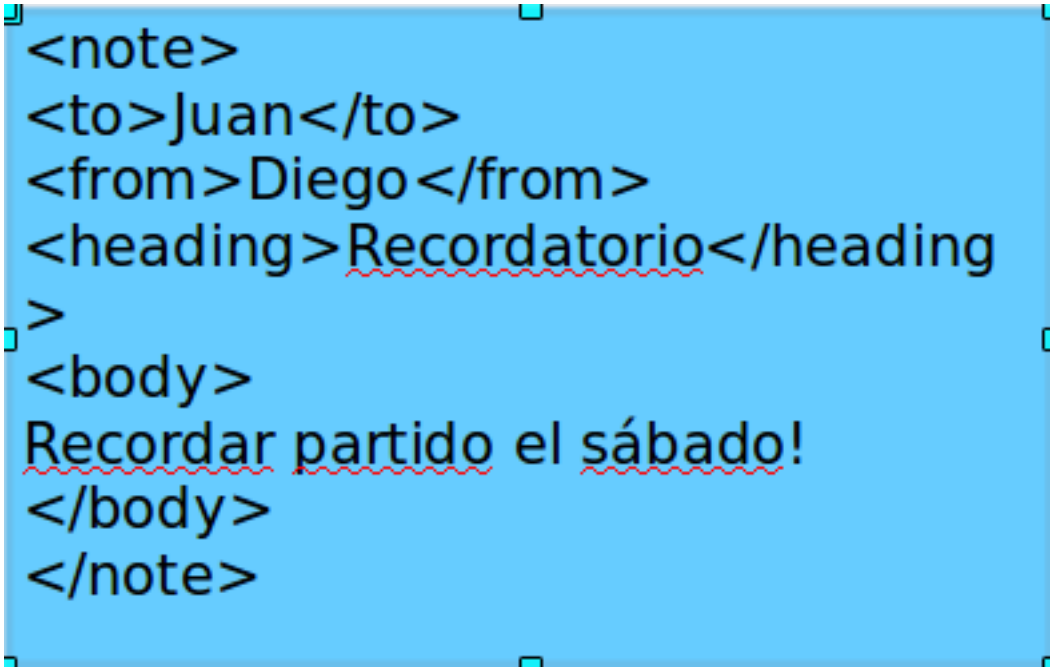
Se recomienda un almacén de documentos cuando la aplicación no necesita la complejidad de SQL y sólo almacene datos simples sin mucha relación.

También es un buen sistema cuando la aplicación necesita velocidad sin comprometer la confiabilidad y la escalabilidad.

Cómo funciona su almacenamiento de datos

Basado en XML

En un almacén de documentos, los datos pueden estar basados en XML, o sea usar archivos XML para almacenar los datos en disco. Un ejemplo de documento XML es el siguiente:



Las primeras bases de datos de documentos se construyeron alrededor del estándar de documento XML.

Las bases de datos XML son interesantes principalmente como los precursores arquitectónicos de la base de datos de documentos JSON modernas.

Las bases de datos XML de hoy representan un nicho significativo pero pequeño en el mercado global de bases de datos.

XML (eXtensible Markup Language) surgió como resultado de la convergencia de esfuerzos para desarrollar un lenguaje de marcado generalizado como el sucesor de varios formatos especializados como SGML y una realización que HTML, la base de la Web 1.0, combinaba el diseño y los datos de manera incómoda.

XML era capaz de representar casi cualquier forma de información y, junto con las hojas de estilo en cascada (CSS) que controla la "renderización", permitió a los sitios web de segunda generación separar los datos y el formato.

XML fue ampliamente utilizado más allá de estos casos de uso de Web 2.0 y se convirtió en un formato estándar para muchos tipos de documentos, que eventualmente incluyen documentos de procesamiento de texto y hojas de cálculo. XML también es la base para muchos protocolos de intercambio de datos y, en particular, fue una base para especificaciones de servicios web tales como SOAP (Protocolo simple de acceso a objetos).

Durante la década de 2000, se esperaba ampliamente que la mayoría de los documentos en una organización fuera de la base de datos relacional terminaría representada como XML. Y si bien el impulso detrás de XML se ha ralentizado, hoy una gran variedad de tipos de documentos usan XML.

Herramientas y estándares XML

XML es respaldado por un rico ecosistema que incluye una variedad de estándares y herramientas para ayudar con la creación, validación, búsqueda y transformación de documentos XML. Éstas incluyen:

- XPath: una sintaxis para recuperar elementos específicos de un documento XML. XPath proporciona una manera simple y conveniente de filtrar documentos usando comodines y etiquetas de referencias.

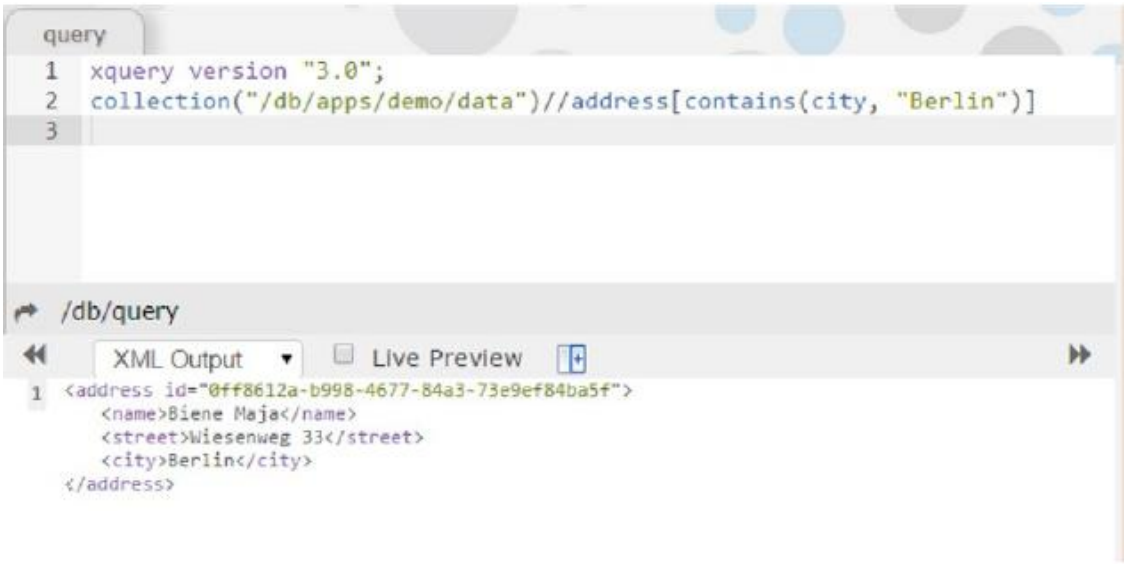
- XQuery: un lenguaje de consulta para interrogar documentos XML. El XQuery Update relacionado proporciona mecanismos para modificar un documento. XQuery es a veces denominado "el SQL de XML".

- Esquema XML: un tipo especial de documento XML que describe los elementos que puede estar presente en una clase especificada de documentos XML. El esquema XML se puede usar para validar que un documento está en el formato correcto o para ayudar a los programas que desean interrogar documentos XML que se ajusten a ese formato.

- XSLT (Transformaciones de lenguaje de hojas de estilo extensibles): un lenguaje para transformar documentos XML en formatos alternativos, incluidos formatos no XML como HTML.

- DOM (Document Object Model): una API orientada a objetos que los programas pueden usar para interactuar con XML, XHTML y documentos estructurados de manera similar.

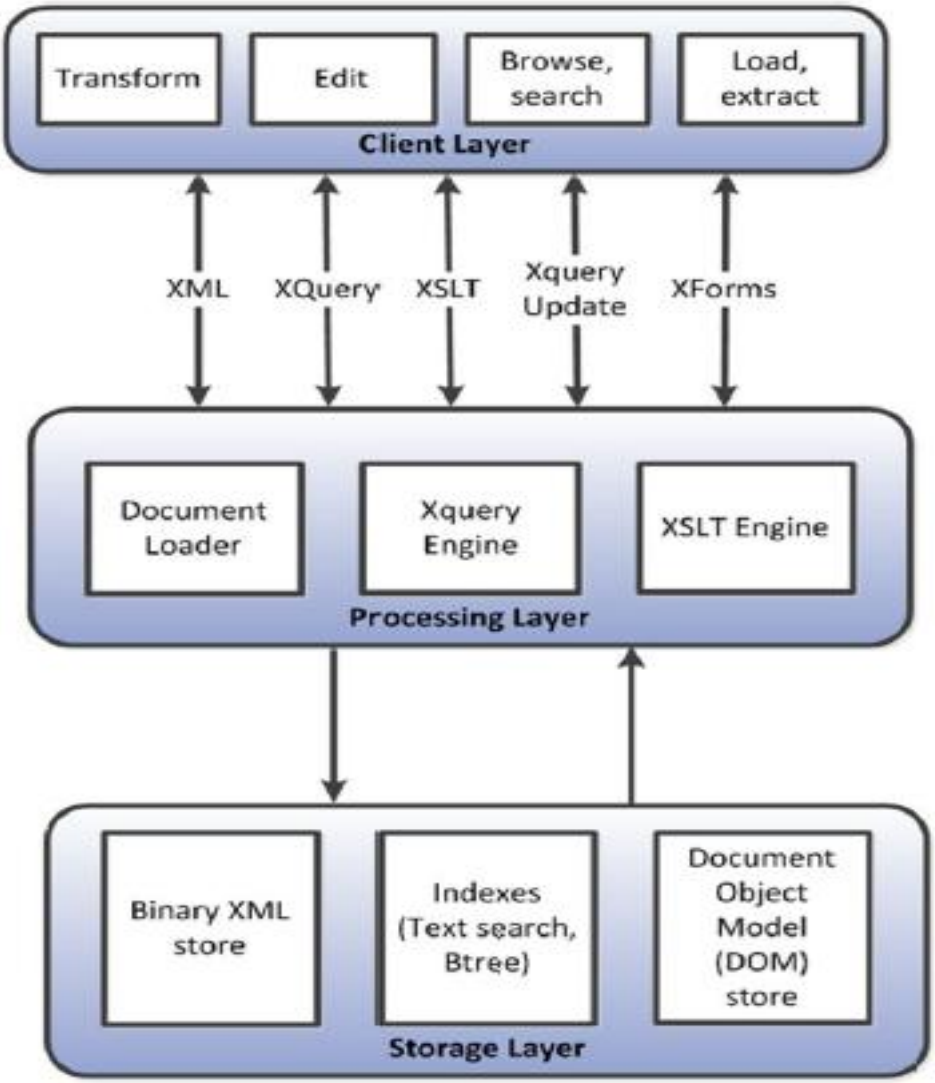
Como el "SQL para XML", XQuery ocupa un lugar destacado en la mayoría de las arquitecturas de bases de datos XML. La siguiente figura muestra una declaración simple de XQuery que busca documentos con un elemento ADDRESS que incluye la cadena "Berlin" dentro de un elemento CITY incrustado.



Bases de datos XML

El creciente volumen de documentos XML dentro de las organizaciones proporcionó una motivación para alguna forma de sistema de gestión de documentos XML o base de datos XML nativa.

Las bases de datos XML generalmente consisten en una plataforma que implementa los diversos estándares XML, como XQuery y XSLT, y que proporciona servicios para el almacenamiento, la indexación, la seguridad y el acceso simultáneo de archivos XML La siguiente figura ilustra una arquitectura de base de datos XML genérica simplificada.



Una variedad de bases de datos XML surgió durante la primera mitad de la década de 2000, y experimentaron un saludable consumo. Sin embargo, no se posicionaron como alternativas al RDBMS sino, más bien, como un medio de gestión a la expansión de documentos XML que enfrentaron muchas organizaciones.

Dos de las bases de datos XML más importantes incluyen la base de datos XML de código abierto eXist y la base de datos XML comercial MarkLogic.

Soporte de XML en sistemas relacionales

Los proveedores de bases de datos relacionales presentaron compatibilidad con XML dentro de sus ofertas principales, lo que generalmente permite que los documentos XML que deben almacenarse dentro de columnas largas (long/BLOB) en las tablas de la base de datos y proporcionar soporte para varios estándares XML como DOM, XSLT y XQuery.

Además, se agregó soporte XML a la definición ANSI SQL (SQL / XML), que permite la manipulación de XML dentro de sentencias de lenguaje SQL estándar. Soporte para estas extensiones SQL aparece en Oracle, Postgres, DB2, MySQL y SQL Server.

Soporte de JSON en sistemas relacionales

En este caso, la base de datos usa archivos JSON en lugar de XML. Este tipo de almacenamiento es mejor cuando la salida debe ser utilizable para una aplicación web o para programar el almacenamiento de objetos porque las herramientas JSON están incluidas en todos los navegadores web modernos y también se pueden usar para representar un objeto de cada lenguaje de programación orientado a objetos.

Ejemplo de archivo JSON:

```
{
" docs " :
[
{
" i d " : "562 e35cf7f2d7ead13ac9188 " ,
" index " : 0 ,
" age " : 40 ,
" eyeColor " : " blue " ,
"name " : "Reeves Pruit t " ,
" gender " : "male " ,
"whishes " : {
" f r u i t s " : "Avocado " ,
"number " : 195
}
} ,
{
" i d " : "562 e35c f51a04e5226fa09e7
" ,
" index " : 1 ,
" age " : 20 ,
" eyeColor " : "brown " ,
"name " : "Dean Blankenship " ,
" gender " : "male " ,
"whishes " : {
" f r u i t s " : "Melon " ,
"number " : 337
}
}
}
```

Bases de datos de documentos JSON

XML tiene muchas ventajas como formato estándar para documentos basados en archivos y para el intercambio de datos. Pero tiene una serie de inconvenientes como formato de almacenamiento para aplicaciones de bases de datos serias.
XML es criticado justificadamente como un desperdicio de espacio y computacionalmente costoso de procesar. Las etiquetas XML son detalladas y repetitivas típicamente aumentando la cantidad de almacenamiento requerido por varios factores.

En parte como resultado, en el lenguaje XML el formato es relativamente costoso de analizar. XML ha sido durante mucho tiempo el formato predominante para los archivos estructurados, pero para el intercambio de datos y para bases de datos de documentos en sí, un formato más reciente -Notación de objetos JavaScript (JSON) -prometía mayores beneficios y ha alcanzado una popularidad mucho mayor que sus predecesores XML.

Las bases de datos basadas en documentos JSON y las bases de datos basadas en documentos XML comparten muchas similitudes, no la menor de ellas es la similitud superficial entre los formatos XML y JSON.

Sin embargo, cada uno está diseñado para admitir casos de uso bastante diferentes y aplicaciones significativamente diferentes.

Las bases de datos XML típicamente se usan como sistemas de gestión de contenido; es decir, organizar y mantener colecciones de archivos de texto en Formato XML: documentos académicos, documentos comerciales, etc.

Las bases de datos de documentos JSON, por otro lado, la mayoría soporta cargas de trabajo operacionales basadas en la web, almacenando y modificando el contenido dinámico y datos transaccionales en el núcleo de las aplicaciones modernas basadas en la web.

JSON y AJAX

JSON fue creado por el pionero de JavaScript Douglas Crockford como parte de un intento de construir un marco para aplicaciones web más dinámicas e interactivas.

JSON se pensó deliberadamente como un peso más ligero sustituto de XML, y se convirtió en una alternativa significativa a XML en el modelo de programación AJAX que impulsó una revolución en aplicaciones web a mediados de la década de 2000.

AJAX (Asynchronous JavaScript And XML) es un patrón de programación flexible en el que JavaScript en un navegador web se comunica con un servidor web de fondo a través del intercambio asincrónico de XML o JSON de documentos, en lugar de intercambiar páginas HTML completas. Fue AJAX lo que permitió una rica experiencia interactiva proporcionada por aplicaciones web innovadoras como Google Maps y Gmail. Aunque la "X" en AJAX se refiere a XML, los documentos JSON también se pueden usar como medio de interacción con el servidor web.

De hecho, debido a su estrecha integración con JavaScript, JSON se hizo cada vez más más popular que XML para el intercambio de datos. Luego de unos años de la introducción del enfoque AJAX, todos los desarrolladores web serios se familiarizaron con la programación de JavaScript y con el modelo JSON.

Detalles de las bases de datos JSON

El surgimiento de bases de datos de almacenes clave-valor como Dynamo de Amazon, junto con el crecimiento de la popularidad de JSON como formato de intercambio de datos, preparó el escenario para la aparición de la base de datos de documentos JSON.

No hay una sola especificación o manifiesto que delinee las propiedades de un documento basado en JSON en una base de datos de este tipo.

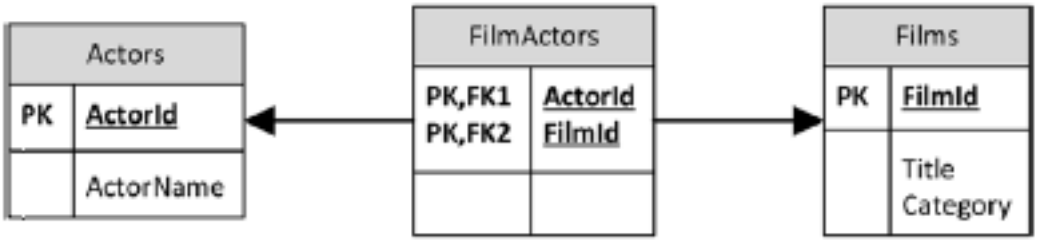
Para ser una base de datos de documentos JSON, todo lo que necesita hacer es almacenar datos en el formato JSON. En una base de datos de documentos JSON, la jerarquía de almacenamiento suele ser la siguiente:

- Un documento es la unidad básica de almacenamiento, que corresponde aproximadamente a una fila en un RDBMS. Un documento comprende uno o más pares clave-valor, y también puede contener documentos y matrices anidados. Las matrices también pueden contener documentos que permitan una estructura jerárquica compleja.

- Un grupo de recopilación o datos es un conjunto de documentos que comparten un propósito común; esta es más o menos equivalente a una tabla relacional. Los documentos en una colección no tienen que ser del mismo tipo, aunque es típico que los documentos en una colección representen una categoría común de información.

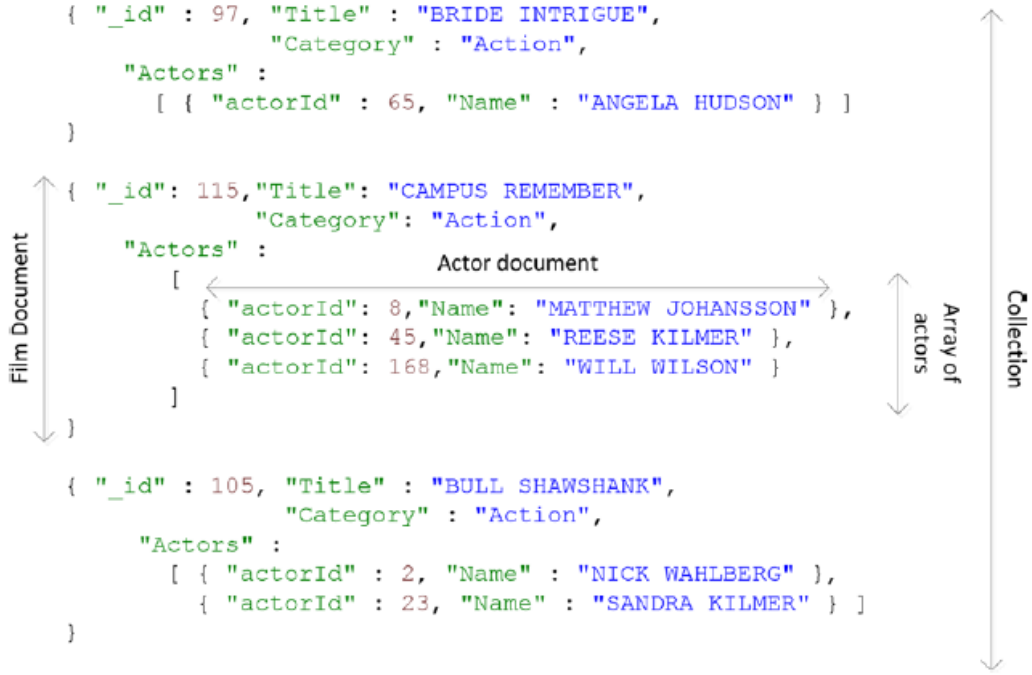
Mientras que una base de datos de documentos podría implementar teóricamente un tercer esquema de formulario normal exactamente como se usaría dentro de un sistema relacional, las bases de datos de documentos más típicamente modelarían los datos en un formato más pequeño como número de colecciones, con documentos anidados que representan las relaciones maestro-detalle.

Por ejemplo, si se considera una base de datos de películas y actores. En un modelo de datos relacionales, se representarían actores y películas en tablas separadas, y se crearía una tabla de unión que indique qué actores aparecieron en que film, así como se muestra en la figura.



En una base de datos de documentos JSON, se podrían crear tres colecciones con pares clave-valor correspondientes a las columnas en un esquema relacional, pero hacerlo sería antinatural.

Las bases de datos de documentos generalmente no proporcionan operaciones de unión, y los programadores generalmente prefieren que la estructura JSON se asocie estrechamente con la estructura de objeto de su código. Entonces, sería más natural representar estos datos como se muestra en la figura.



Modelos de datos en bases de datos documentales

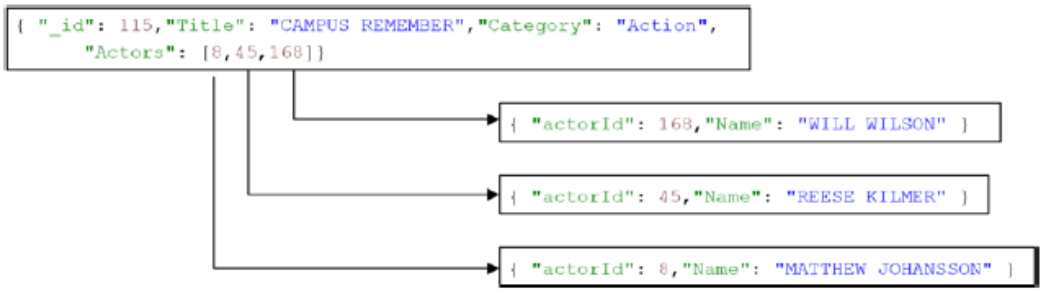
En la figura anterior, los "actores" están anidados como una matriz dentro de los documentos de "películas".

Este patrón a menudo se refiere como 'incrustación de documentos'. El patrón de diseño tiene la ventaja de permitir que una película y todos sus actores sean recuperados en una sola operación, y evita la necesidad de realizar uniones dentro del código de la aplicación.

Por otro lado, el enfoque da como resultado la duplicación de "actores" en múltiples documentos, y en un diseño complejo, esto podría llevar a problemas y posiblemente inconsistencias si alguno de los atributos del "actor" necesita ser cambiado.

La cantidad de actores en una película es relativamente pequeña, pero en otros escenarios de aplicación, puede haber problemas si la cantidad de miembros en un documento incrustado aumenta sin límite (porque el el tamaño de un solo documento JSON suele estar limitado a 64 MB en MongoDB, por ejemplo). Por estas razones, un diseñador de bases de datos puede elegir en su lugar vincular múltiples documentos utilizando identificadores de documentos, de forma muy parecida a como una base de datos relacional relaciona las filas a través de claves externas.

Por ejemplo, en la siguiente figura, se inserta una matriz de identificaciones de actores en el documento "películas", que se puede usar para localizar a los actores. que aparece en una película.



El modelado de datos en las bases de datos documentales es menos determinista que para las bases de datos relacionales: no hay equivalente a la tercera forma normal que define un modelo "correcto".

Además, al modelar en un entorno relacional la base de datos se basa principalmente en la naturaleza de los datos que se almacenarán, en una base de datos documental, en cambio, resulta mucho más significativo tener en cuenta la naturaleza de las consultas que se ejecutarán.

2. Algunas opciones de bases de datos de documentos

CouchDB

Entre las opciones de bases de datos documentales se encuentra CouchDB que es una base de datos JSON que usa un árbol B para el almacenamiento. CouchDB, creado por Damien Katz, fue el primer sistema de base de datos basado en JSON.

Damien Katz tenía trabajó en Lotus Notes, un sistema de colaboración con sólidas capacidades de manejo de documentos. En 2005, él decidió crear un sistema de base de datos más alineado con el desarrollo web y modelos de programación orientado a objetos.

El resultado fue CouchDB. Inicialmente, CouchDB se escribió en C ++ y almacenó documentos XML.

Alrededor de 2007, una nueva arquitectura surgió e incorporaba JSON como el formato de almacenamiento primario, un comando de JavaScript, una interfaz de consulta y un motor central reescrito en el lenguaje Erlang.

A medida que maduró, CouchDB adoptó otros paradigmas que habían inspirado almacenes de valores-clave y Hadoop, incluyendo una implementación de JavaScript de MapReduce, una consistencia eventual y modelos para versiones múltiples, y un modelo de clustering / sharding basado en hash para permitir que CouchDB escale entre nodos.

CouchDB se convirtió en un proyecto Apache en 2008 y fue respaldado por IBM. En 2009, se formó la compañia comercial "couch.io"" (más tarde CouchOne) para mantener y promover la tecnología.

MemBase y CouchBase

Cuando el interés en las bases de datos no relacionales estalló en 2009, CouchDB ya tenía varios años activos en desarrollo y parecía estar bien ubicado para beneficiarse del creciente rumor que rodeaba a "NoSQL".

Membase fue otro sistema no relacional que experimentó una captación significativa durante este período.

La base de datos Membase proporcionó una variación persistente en el extremadamente popular "framework Memcached". Memcached es un caché de objetos distribuidos de solo lectura que comúnmente se implementa en conjunción con MySQL para reducir la carga de la base de datos.

Los objetos se distribuyen a través de múltiples nodos de Memcached, y se puede ubicar mediante una búsqueda de clave hash. Si los datos están en un servidor Memcached, se evita una lectura de la base de datos.

Membase proporcionó una solución compatible con Memcached en la que también se podían modificar los datos y persistir en el disco.

Membase era, por lo tanto, particularmente atractivo para aquellos que tenían una inversión existente en tecnología Memcached, que ofrece la posibilidad de que una aplicación se pueda convertir a Membase de Memcached / MySQL.

Membase fue inicialmente conocido como la base de datos subyacente de Zynga, un popular juego en línea de Farmville. Mientras tanto, a pesar de los muchos logros técnicos de CouchDB, la base de datos CouchDB y la empresa CouchOne parecía estar luchando por establecer un nicho comercial.

CouchDB carecía de una arquitectura de escalamiento viable. A principios de 2011, se anunció una fusión entre Membase y CouchOne.

La compañía resultante fue llamaba CouchBase. CouchBase donó el código CouchDB existente a la comunidad Apache, y se embarcó en un nuevo esfuerzo para fusionar las capacidades de CouchDB y MemBase.

En el resultado el servidor Couchbase, el motor JSON de CouchDB se fusionaron con la capa clave-valor compatible con Memcached de MemBase.

Couchbase heredó la interfaz MapReduce de CouchDB para crear consultas y vistas, pero Couchbase 4.0 introdujo una capa similar a SQL para acceso a documentos llamada N1QL (Lenguaje de consulta de formularios normal no primordial).

Consulta de datos y el paradigma map/reduce

Para la recuperación de datos, un almacén de documentos es totalmente diferente de una base de datos relacional. No por la potencia y la complejidad de estas últimas, sino por tener algunas herramientas que pueden ser utilizadas por el desarrollador para recuperar un conjunto específico de datos.

Una de esas herramientas es el paradigma map/reduce. Se basa en dos operaciones y está optimizado para aprovechar la programación paralela y el clúster múltiple de la base de datos. La función de mapeo se ejecuta en cada documento y devuelve todo el documento o un subconjunto del mismo.

Este enfoque permite que la consulta se ejecute simultáneamente en procesos múltiples y en algún nodo múltiple de la base de datos. Todos los resultados se agrupan en un conjunto de datos temporales.

Después de la operación del mapeo, la base de datos mezcla los datos, los reorganiza, agregándolos en unos pocos conjuntos de datos y luego llama al método de reducción.
La reducción de los datos es una operación que combina los datos usando una regla especificada por el usuario y genera el resultado. Se realiza una reducción en cada conjunto de datos producido por el mapeo por lo que esta operación también se realiza en paralelo. Después de eso, también se puede reducir el resultado si es necesario.

El resultado final del mapeo y la reducción se almacena en un documento especial llamado vista. Una vista se puede usar directamente después de una consulta como en una base de datos relacional o se puede mantener en la memoria durante una cantidad de tiempo especificada.

Esto permite a la base de datos pre-computar consultas pesadas que no necesitan el recurso y caché actualizados o mantener una solicitud que a menudo se llama. Esto permite reducir la cantidad de trabajo en el servidor mediante la negociación de la coherencia de los datos.

MongoDB

En 2007, los fundadores e ingenieros senior de DoubleClick, la empresa de publicidad en línea, que acababa de ser adquirido por Google, establecieron una nueva empresa llamada 10gen.

La compañía pretendía crear una PaaS (Plataforma como servicio) para ofrece una oferta similar a Google App Engine.
La plataforma requería un sistema escalable y un motor de almacenamiento de datos elástico.

En ausencia de un candidato existente adecuado, el equipo creó su propia base de datos, que llamaron MongoDB.

En 2008, 10gen decidió centrarse exclusivamente en MongoDB, y en 2009, lanzó el producto bajo una licencia de código abierto junto con una distribución empresarial comercial.

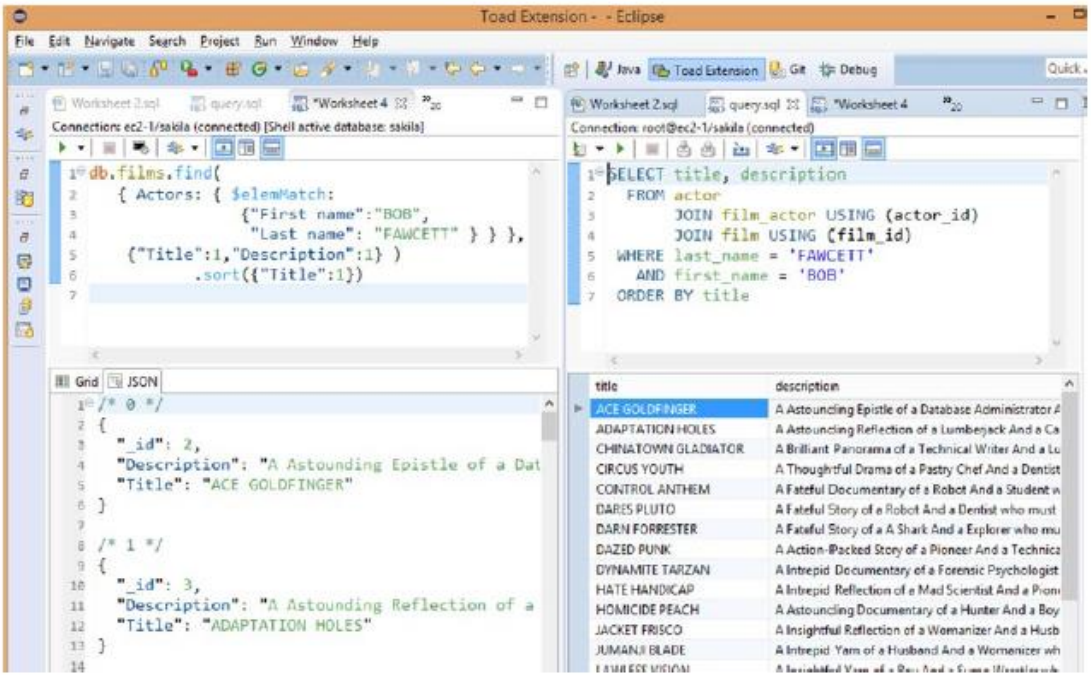
MongoDB es una base de datos documental orientada a JSON, aunque internamente utiliza una variante codificada en binario de JSON llamado BSON.

El formato BSON admite una carga de análisis más baja que JSON, así como una mayor compatibilidad para tipos de datos tales como fechas y datos binarios.

MongoDB está principalmente optimizada para la velocidad y puede ser más de 25 veces más rápido que CouchDB, pero no ofrece la misma fiabilidad. Además esta base de datos no cumple con ACID porque solo ofrece atomicidad en cada documento y no al nivel de transacciones.

Sin embargo, MongoDB tiene una ventaja ya que proporciona una capacidad de consulta basada en JavaScript que es razonablemente fácil de aprender, al menos en comparación con los enfoques de Map/Reduce tales como los requeridos inicialmente en CouchDB.

La figura siguiente compara una consulta de JavaScript de MongoDB para recuperar películas que cuentan con un actor específico con el equivalente SQL en MySQL.



MongoDB estableció una fuerte ventaja en el espacio de las bases de datos NoSQL al proporcionar un ecosistema y una arquitectura amigable para los desarrolladores.

Los desarrolladores que buscan una alternativa no relacional, típicamente a MySQL u Oracle, encuentran relativamente fácil comenzar con MongoDB.

La adopción liderada por desarrolladores de MongoDB fue robusta, y hoy MongoDB puede afirmar que es la base de datos no relacional más utilizada.

En muchos aspectos, el aumento de MongoDB hoy se asemeja al ascenso de MySQL en la última década.

En ambos casos, estas bases de datos ingresaron a la organización no como resultado de algún tipo de plan de tecnología estratégica, sino como resultado de su popularidad entre los desarrolladores.

Así como MySQL se convirtió en la base de datos predeterminada para las aplicaciones de la LAMP (Linux-Apache-MySQL-PHP) a principios de la década de 2000, parece que MongoDB ha logrado un puesto similar dentro de la comunidad moderna de desarrollo web.
MongoDB puede carecer de algunas de las capacidades de escalabilidad y rendimiento de otras ofertas de NoSQL más actuales como Cassandra o HBase, aunque la implementación del motor de almacenamiento WiredTiger en la versión 3.0 proporciona el motor base con un aumento significativo en la capacidad.

Sin embargo, ha impulsado a un alto nivel a muchos sitios web de gran escala, y parece que va a ser una base de datos NoSQL líder para el futuro inmediato.

Otra soluciones con código cerrado

Lotus Note: Note (ahora IBM Note)

Es un software cliente-servidor que permite la administración corporativa y colaborativas de funciones entre empleados con correos, calendario, lista de tareas y otros. El núcleo componente del servidor es una base de datos documental.. Debido a su naturaleza patentada, no se sabe mucho sobre sus detalles de implementación.

DocumentDB

Document DB es una base de datos patentada desarrollada por Microsoft que se ejecuta en su plataforma en la nube. Está principalmente basado en un almacén de datos de tipo JSON.

SimpleDB

SimpleDB es la versión desarrollada por Amazon de Document DB. Es casi equivalente, pero se ejecuta en el servidor de Amazon.

Bibliografía utilizada y sugerida

Libros y otros manuscritos

Catherine M. Ricardo, Susan D. Urban, Databases Illuminated, USA New York, Jones & Bartlett Learning 2017.

Grant Allen, Beginning DB2, Beginning DB2 from Novice to Professional, USA New York, APRESS 2008.

Guy Harrison, Next Generation Databases, USA New York, APRESS 2016

Raul F. chong, Xiaomaei Wang, Michael Dang, Dwaine R. Snow, Understanding DB2 Learning with Examples, Second Edition, IBM PRESS 2007.

Steve Hoberman, Data Modeling for MongoDB, USA New Jersey, Technics Publications 2014.

Lo que vimos:

En esta Unidad vimos los conceptos de bases de datos documentales y sus variantes XML y JSON. Comparamos ejemplos concretos de código que podrían producir un resultado equivalente entre las variantes relacionales y no SQL. También, abordamos distintas herramientas presentes en el mercado.

Lo que viene:

En la próxima Unidad vamos a ver de qué se tratan las bases de datos que funcionan en memoria, sus ventajas y desventajas. Además, vamos a comprender conceptualmente cuándo conviene usarlas.