# 2.3.- BASE DE DATOS NATIVA XML: XINDICE.

## 2.3.1. Introducción.

Una base de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso, es decir, nos permite almacenar información para usarla posteriormente. Por otro lado, la tecnología XML permite estructurar la información de manera sencilla gracias al uso de etiquetas.

Si se unen ambos conceptos, la utilidad de las bases de datos y el extenso abanico de posibilidades que proporciona XML, se tiene las bases de datos XML. Dentro de éstas se encuentran las Bases de Datos Nativas XML que son una elección acertada para el caso de almacenamiento masivo de contenido nativo XML. Existen numerosas aplicaciones en las que este tipo de almacenamiento está especialmente indicado como portales de información corporativa, datos personales, catálogos e intercambio de información de negocio. Un ejemplo es el almacenamiento de periódicos en formato XML, como es el caso de “El heraldo de Aragón” o “Las Provincias”, en Valencia. [10] [11]

Su elección en este proyecto está justificada dado que la premisa es que se dispone de un elevado número de Guías de Práctica Clínica a gestionar, las cuáles se proporcionarán en formato XML.

Las Bases de Datos Nativas XML (NXD -Native XML Database) definen el modelo lógico de un documento XML, y almacena y recupera los documentos según ese modelo. Estas bases de datos tienen como unidad fundamental de almacenamiento lógico el documento XML, tal como una base de datos relacional tiene una fila en una tabla como su unidad fundamental de almacenamiento lógico.

Con todo esto, existen tres puntos principales a recordar de las NXD:

- La base de datos NXD se especializa en almacenar datos XML y almacena todos los componentes del modelo XML de forma intacta.
- La base de datos almacena y recupera documentos XML.
- Una NXD puede que realmente sea una base de datos independiente.

Las NXD no representan un nuevo modelo de base de datos de bajo nivel, y no pretenden sustituir las bases de datos existentes. Son simplemente una herramienta que facilita la manipulación de documentos XML y proporciona un almacenamiento robusto.

### 2.3.2. Almacenamiento de datos XML.

Los datos contenidos en un documento XML pueden ser almacenados según dos modelos:

- **Centrados en los datos (data-centric):** este tipo de documentos suelen tener una estructura bien definida cuyos datos pueden extraerse del documento e indexarse con alguna base de datos convencional. Estos datos son altamente estructurados con tamaño limitado y reglas poco flexibles para campos opcionales y contenidos. Se dice que se trata de datos de grano fino ya que la unidad independiente de datos más pequeña es el nivel de PCDATA.

  Los documentos data-centric son documentos que utilizan XML para el transporte de datos. Algunos ejemplos: datos científicos, planificaciones de vuelos, datos de Stock u órdenes de pedidos.

- **Centrados en el documento (document-centric):** tienen una estructura irregular y de gran importancia, lo que implica una mayor dificultad a la hora de realizar consultas ya que éstas se harán tanto del contenido como de la estructura del docuemento. Los datos son más impredecibles en tamaño y contenido que los anteriores. Se dicen que son datos de grano grueso ya que la unidad independiente más pequeña puede ser incluso un documento completo. Algunos ejemplos: libros, correos electrónicos o anuncios.

La mayoría de los sistemas de almacenamiento enfocan a servir uno de esos formatos mejor que el otro. Las bases de datos llamadas XML-enabled BD son típicamente mejores al tratar con requerimientos centrados en los datos, mientras que las Bases de Datos Nativas son mejores para aquellos centrados en el documento.

### 2.3.3. Almacenamiento de documentos XML.

Las Bases de Datos XML permiten tres tipos de almacenamiento de documentos XML:

- Almacenamiento no estructurado: los documentos XML se almacenan directamente en formato de texto como atributo de tipo fichero y se deben proporcionar funciones adicionales para poder acceder a la información dentro de los documentos XML.

- Almacenamiento estructurado: la estructura de un documento XML se convierte a un esquema de la base de datos que hay por debajo.

- Mapeo: el contenido de documentos XML se mapea en esquemas de bases de datos específicamente diseñado para este contenido. Si la estructura del documento XML no es compatible con la estructura de la base de datos, el documento debe ser transformado para ajustarlo a la estructura de la base de datos antes de almacenarlo

## 2.3.4. Tipos de Bases de Datos XML.

Para poder distinguir los diferentes tipos de bases de datos XML que existen se debe estudiar el tipo de almacenamiento y gestión de documentos XML que hacen. De esta forma, se denominan XML-enabled BD (Bases de datos que permiten XML) a aquellas bases de datos que mapean el esquema del documento en un esquema de base de datos y transfiere los datos acorde a ese mapeo; mientras que las Bases de Datos Nativas XML utilizan un conjunto de estructuras fijas que nos permiten almacenar cualquier documento XML de forma nativa. [11] [12]

Estos tipos de bases de datos son:

A. **XML-enabled BD:** estas bases de datos tienen su propio modelo de datos (relacional u orientadas a objeto) y mapean instancias del modelo de datos XML en instancia de su propio modelo de datos. Por tanto, requieren de un modelo previo (modelo entidad-relación) sobre el cual se modela la estructura existente en un tipo de documento XML en particular. Alguno de estos modelos son:

   - Bases de datos relacionales: se basan en las bases de datos relaciones (tablas bidimensionales) como único medio para representar los datos del mundo real.

   - Bases de datos Orientadas a Objetos: soportan un modelo de objetos puro, en la medida de que no están basados en extensiones de otros modelos más clásicos como el relacional.

   Este tipo de bases de datos realizan 2 tipos de procesos : “Shredding” y “Publishing”: obtener los datos descritos en un documento XML, mapearlos a sus correspondientes “tablas” y posteriormente, tratar de reconstruir el documento XML original a través de la obtención de los datos que se almacenaron en dichas “tablas”. Una vez desglosan la información de un documento XML en su correspondiente esquema relacional o de objetos no se garantiza la reconstrucción del documento XML original.

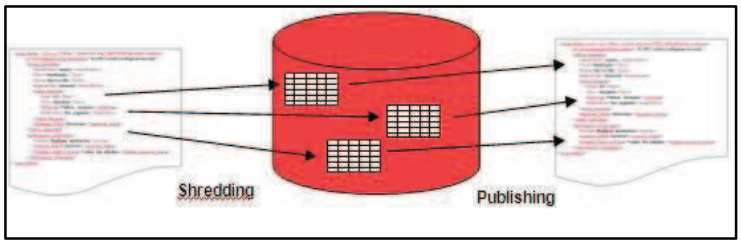


Ilustración 6.-Shredding y Publishing para XML-enabled BD.

Las XML-enabled BD son útiles cuando se quiere publicar datos existentes como XML o importar datos desde un documento XML en una BD existente. No para almacenar documentos XML completos. Requiere tiempo de diseño para el mapeo de esquemas. No pueden almacenar documentos cuyo esquema no sea conocido en el tiempo de diseño.

B. **BD Nativas XML:** utiliza el modelo de datos directamente. El aspecto principal es el almacenamiento de los documentos XML de forma nativa. Entre sus características que se verán posteriormente se encuentran:

- Define un modelo (lógico) para un documento XML (para el documento, no para los datos).
- Almacena y recupera documentos de acuerdo a ese modelo.
- Como mínimo, el modelo debe incluir elementos, atributos, manejo de PCDATA (abreviatura de *"datos de carácter analizados"* que significa que el elemento va a contener texto) y orden dentro del documento.

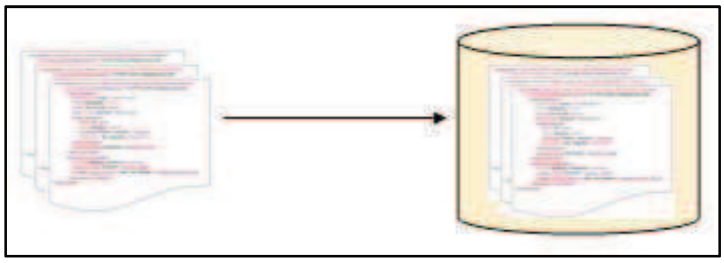

**Ilustración 7.- Almacenamiento de un documento XML en una BD Nativa.**

En la siguiente figura se puede observar la diferencia entre ambos tipos mostrando cómo queda el documento XML al ser almacenado en cada tipo de base de datos:

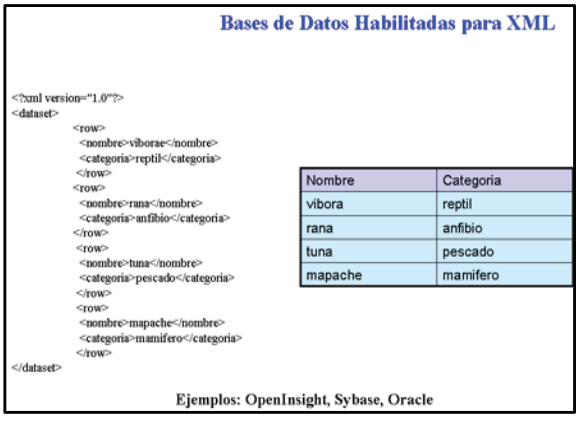


**Ilustración 8.- Almacenamiento en XML-enabled DB.**

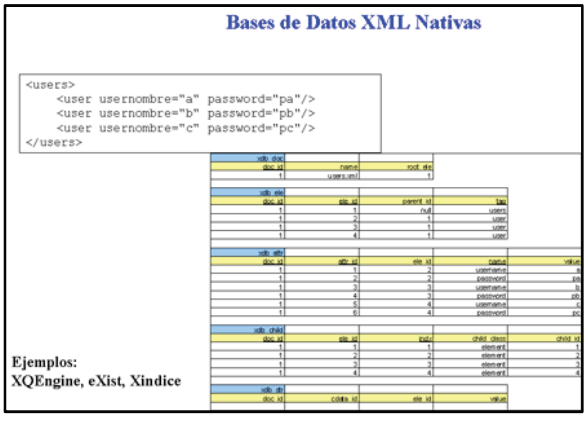


**Ilustración 9.- Almacenamiento en Bases de Datos Nativas XML**

A continuación se muestra una tabla con las distintas posibilidades de elección de una Base de Datos XML según su tipo de almacenamiento:

| | BD XML nativas | Extensiones de BD | | |
|---|---|---|---|---|
| | | Almacenamiento no estructurado | Almacenamiento estructurado | Mapeo |
| **Comercial** | eXcelon XIS<br>GoXML DB<br>Infonyte-DB<br>Tamino<br>TEXTML<br>X-Hive/DB | IBM DB2 XML Extender<br>Microsoft SQLXML<br>Oracle XML DB | | Oracle XML DB/ Mapeo Estructurado |
| **Código abierto** | dbXML<br>eXist<br>Xindice | | Ozone/XML (OO) | |
| **Investigación** | Lore<br>Natix<br>PDOM<br>TIMBER | | Monet XML<br>Shimura et al.<br>XML Cartridge | |

Tabla 10.- Tabla de Bases de Datos XML.

## 2.3.5. Bases de Datos Nativas XML.

### 2.3.5.1. Introducción.

El término “base de datos nativa XML” (NXD) se refiere a un sistema de base de datos cuyo propósito principal es el de almacenar documentos XML. La organización *XML:DB Initiative for XML Databases* describe una base de datos de este tipo como un: “modelo lógico para documentos XML que almacena y recupera estos documentos de acuerdo a dicho modelo”.

Por tanto, la unidad básica es un documento XML y se trata de un almacenamiento centrado en documento. Son capaces de almacenar la información con su estructura lógica intacta, así como, recuperarla en su formato orginial.

Estas bases de datos ganan el término nativas porque almacenan los datos estructurados como XML sin la necesidad de traducir los datos a una estructura relacional o de objeto.

### 2.3.5.2. Características.

Las características[8][10][11] que se puede encontrar en el uso de una base de datos Nativa y que la diferencian de otros tipos son las siguientes:

a) Emplean como unidad lógica fundamental de almacenamiento el documento XML: se define un modelo (lógico) para un documento XML (para el documento, no para los datos).

b) Hacen uso del modelo que propone XML, el cual es una estructura en árbol. Con este modelo es posible almacenar y recuperar documentos de acuerdo a ese modelo.

c) Representa como “nodos” los elementos, atributos, instrucciones de procesamiento, comentarios y cualquier otro elemento constituyente de un documento XML.

d) Preservan el orden del documento, las instrucciones de procesamiento, los comentarios, las secciones PCDATA y las entidades. Como mínimo, el modelo debe incluir elementos, atributos, manejo de PCDATA y orden dentro del documento.

e) Soportan lenguajes de consulta XML.

f) Permiten almacenar cualquier tipo de documentos XML (bien formados), sin la necesidad de tener un modelo previo ligado a cada tipo de documentos XML que se quisiera almacenar.

g) No tienen ningún modelo de almacenamiento físico subyacente concreto. Pueden ser construidas sobre bases de datos relacionales, jerárquicas, orientadas a objetos o bien mediante formatos de almacenamiento propietarios.

h) Almacenamiento de documentos en colecciones:

- Las colecciones juegan en las bases de datos nativas el papel de las tablas en las bases de datos relacionales.

- Los documentos se suelen agrupar, en función de la información que contienen, en colecciones que a su vez pueden contener otras colecciones.

i) Validación de los documentos.

j) Consultas: la mayoría de las BD XML soportan uno o más lenguajes de consulta. Uno de los más populares es XPath.

k) Indexación XML: se ha de permitir la creación de índices que aceleren las consultas realizadas frecuentemente.

l) Creación de identificadores únicos: cada documento XML tiene asociado un identificador único por el que será reconocido dentro del repositorio.

m) Actualizaciones y borrados: las BD nativas tienen una gran variedad de estrategias para actualizar y borrar documentos. Entre alguna de ellas, muchos sistemas soportan XUpdate para llevar a cabo esta funcionalidad.

### 2.3.5.3. Tipos de Base de Datos Nativas XML.

Existen dos tipos de Bases de Datos Nativas XML. Estos se muestran a continuación:

A. **Almacenamiento basado en texto (ficheros de texto):** almacena el documento XML entero en forma de texto y proporciona alguna funcionalidad de base de datos para acceder a él.

   Aplican (como mucho) técnicas de compresión para reducir el espacio de almacenamiento y mantienen índices adicionales para aumentar la eficiencia en el acceso a la información. Pueden definirse sobre BD o sistemas de ficheros existiendo dos posibilidades:

   - Posibilidad sencilla: almacenan el documento como un BLOB (Binary Large Object) en una base de datos relacional, o mediante un fichero, y proporcionar algunos índices sobre el documento que aceleren el acceso a la información.

   - Posibilidad sofisticada: almacenan el documento en un almacén adecuado con índices, soporte para transacciones, etc.

B. **Almacenamiento basado en modelo:** definen un modelo de datos lógico (como DOM) para la estructura jerárquica de los documentos XML y almacenan los documentos de acuerdo con ese modelo usando el modelo de almacenamiento físico que se desee (mapeo a BD relacional, Persistent DOM, etc.). Veamos alguna de estas posibilidades:

   - Posibilidad 1: Traducir el DOM a tablas relacionales como Elementos, Atributos, Entidades, etc.

   - Posibilidad 2: Traducir el DOM a objetos en una base de datos orientada a objetos.

## 2.3.5.4. Limitaciones.

Una de las limitaciones que se pueden encontrar al hacer uso de este tipo de base de datos se centra al realizar búsquedas. Esto se debe a que se utiliza XPath como lenguaje de consulta y éste no permite realizar búsquedas muy complicadas, como son el caso de ordenamiento y *cross join*, ya que XPath no fue creada realmente para búsquedas en bases de datos, sino para búsquedas en un documento.

Esta limitación se traslada a que cada vez que se quiera realizar una búsqueda dentro de una colección que contenga un conjunto de documentos XML, se tendrá que realizar una búsqueda individual de cada documento XML, donde se recorra uno por uno el documento completo. Esta búsqueda implica que se tarde demasiado tiempo en obtener el resultado.

Puede verse como una limitación que el uso de Base de Datos Nativa XML sólo permite el almacenamiento de documentos XML y no de otro tipo de

documentos, por lo que se vió anteriormente que se tratan de bases de datos centradas en documentos XML.

#### 2.3.5.5. Conclusiones.

Actualmente se está produciendo una "saturación" de información relacionada con XML. El uso de esta tecnología está siendo cada vez más empleado por las empresas para organizar sus datos y su información. A su vez esta información debe gestionarse de la forma más óptima posible, y esa forma es mediante una Base de Datos Nativa XML.

Como su propio nombre indica, son bases de datos, y como tales soportan transacciones, acceso multi-usuario, lenguajes de consulta, etc., diseñadas especialmente para almacenar documentos XML

La necesidad de capacidades de almacenamiento y obtención de datos de forma rápida y flexible que muchas aplicaciones intensivas en XML solicitan es cubierta por este tipo de base de datos.

En definitiva, las bases de datos "tradicionales", lineales y divididas en columnas y filas no pueden almacenar correctamente una información y datos mucho más complejos, enjundiosos y muy diferentes a los datos para los que estas bases de datos estaban en un principio pensadas.

### 2.3.6. XIndice.

#### 2.3.6.1. Introducción.

XIndice es una base de datos nativa XML de código abierto desarrollada por Apache Software Foundation. Almacena e indexa documentos XML para proveer a otras aplicaciones datos con muy poco procesamiento en el lado del servidor, además provee de funcionalidades que son únicas de los documentos XML y que difícilmente son reproducidas por bases de datos relacionales. Facilita mecanismos de indexación y de optimización de consultas para aumentar el rendimiento. Los documentos XML en Xindice son organizados en estructuras tipo árbol.

Una de las ventajas que ofrece XIndice consiste en que los documentos son almacenados en colecciones o recopilaciones que pueden ser consultadas como un todo.

Aún con las limitaciones que se han visto sobre las Bases de Datos Nativas, XIndice proporciona unas ventajas que han hecho que sea una buena elección para este proyecto. Es apropiado para el almacenamiento masivo de contenido nativo XML, crucial para el caso de la gestión de Guías de Práctica Clínica en el que existen un gran número de GPC en XML a gestionar.

El uso de otro tipo de base de datos para la gestión de GPC en XML implicaría que en el almacenamiento se deba hacer un mapeo a otro modelo lógico lo que se traduce en un mayor tiempo empleado además de que existe el riesgo de no poder recuperar el documento XML como se recibió en un principio. Así también hay que destacar que la velocidad de recuperación con XIndice es mayor ya que se almacena todo el documento físico de forma contigua o unido por punteros físicos (más que lógicos). Esto permite que la recuperación de los documentos, tanto con uniones como sin ellas, se haga de forma más rápida que la utilizada por uniones lógicas, como ocurre en las bases de datos relacionales.

Además XIndice suministra una serie de herramientas que permiten la validación de un documento XML respecto a un esquema, la consulta del documento o datos de éste mediante XPath (entre otros leguajes de búsqueda) y la modificación o actualización de los datos con XUpdate.

El resultado es una optimización de los servicios basados en el almacenamiento y procesamiento de la información junto con una estandarización en el formato de los datos.

### 2.3.6.2. Características.

Las Bases de Datos Nativas XML es una tecnología reciente siendo XIndice un proyecto todavía en desarrollo. XIndice es una base de datos semiestructurados que provee de gran flexibilidad para el almacenamiento de los datos. Esto se debe a que el servidor actualmente soporta el almacenamiento de documentos XML bien formados, es decir, que no dispone de ningún tipo de esquema que indique o filtre los documentos XML que puedan ser almacenados.

Actualmente, XIndice es una poderosa herramienta de gestión de documentos XML de la que destacan las siguientes características:

a) Colecciones de documentos: los documentos XML son almacenados en colecciones sobre las cuales pueden realizarse consultas tanto individualmente, en un documento XML, como en la colección completa.

b) Lenguaje de consulta XPath: la consulta de documentos pertenecientes a una colección se realiza de forma flexible mediante el lenguaje XPath.

c) Indexación XML: se permite definir índices sobre los elementos y los valores de los atributos con el fin de mejorar el rendimiento de las consultas sobre un gran número de documentos XML. Esto disminuirá el tiempo de respuesta de dicha consulta.

d) Implementación de XML:DB XUpdate: permite la modificación de los datos de un documento XML almacenado en una colección sin recuperar la totalidad del documento. Es un lenguaje basado en XML que permite

realizar modificaciones sobre los datos de un documento o incluso de una colección completa.

e) Implementación de la API XML:DB para Java: permite que la base de datos pueda ser utilizada por aplicaciones Java. La intención de esta API es dar portabilidad a las bases de datos XML.

f) Herramientas de administración desde la línea de comandos: XIndice permite realizar todas las operaciones de la API XML:DB desde la línea de comandos. Proporciona al administrador una herramienta con un completo juego de órdenes para la gestión de la base de datos.

g) Arquitectura modular: el servidor de XIndice dispone de una arquitectura modular, permitiendo añadir o quitar fácilmente elementos para adaptar al servidor a un entorno particular o embebido en otra aplicación.

### 2.3.6.3. APIs de XIndice.

Para acceder al sistema de base de datos, XIndice ofrece tres APIs[10] [12] [13] a tener en cuenta:

1) API Core Server: es el nivel más bajo de la API interna, como su propio nombre indica. Esta API se utiliza para acceder al núcleo del sistema de la base de datos. Sólo está disponible para el software que se ejecuta en la misma Java Virtual Machine (JVM) que el motor de la base de datos.

2) API XML-RPC: construida por encima de la API Core Server, se usa para acceder a bases de datos nativas XML con lenguaje diferente a Java.

3) API XML:DB para Base de Datos XML: ofrece acceso a la API Core Server para desarrollar aplicaciones en Java. Existen dos tipos de implementaciones de esta interfaz: uno basado en la API XML-RPC y la otra usa directamente la Máquina Virtual Java (JVM). Es la principal interfaz de programación para las bases de datos nativas XML.

   El concepto de colecciones es introducido en esta API XML:DB. Cada colección reúne un número de documentos XML. Estas colecciones se organizan siguiendo una jerarquía parecida a la estructura de directorios de un sistema operativo. Un recurso es un documento dentro de una colección. Y un servicio es una facilidad que permite realizar acciones como búsquedas, inserciones,…

   De la especificación de la API XML:DB cabe destacar las siguientes interfaces:

   - DatabaseManager: permite el acceso a la base de datos.
   - Collection: representa una colección, conjunto de recursos, de una base de datos XML.

- XMLResource: proporciona acceso a los recursos XML almacenados en una base de datos.
- CollectionManagementService: servicio que permite la gestión básica de las colecciones de una base de datos: crear y borrar colecciones.
- XPathQueryService: servicio que permite la ejecución de consultas XPath en el contexto de una colección completa o de un solo recurso XML.
- XUpdateQueryService: servicio que permite la ejecución de consultas XUpdate en el contexto de una colección completa o de un solo recurso XML.

| Paquete | Interfaz |
|---|---|
| org.xmldb.api | DatabaseManager |
| org.xmldb.api.base | Collection |
| | Configurable |
| | Database |
| | Resource |
| | ResourceIterator |
| | ResourceSet |
| | Service |
| org.xmldb.api.modules | BinaryResource |
| | CollectionManagementService |
| | TransactionService |
| | XMLResource |
| | XPathQueryService |
| | XUpdateQueryService |

**Tabla 11.- Especificación API XML:DB.**

Otros servicios a tener en cuenta ofrecidos por Xindice pero que no forman parte de la XML:DB API son:

- DatabaseInstanceManager: permite controlar las operaciones del servidor utilizando código.
- CollectionManager: permite crear y configurar colecciones desde el código, en lugar de utilizar el servidor

### 2.3.6.4. Rutas de localización.

Una colección de una base de datos se encarga de reunir un conjunto de documentos XML. Su nombre original es *"collection"*. Pero estas colecciones no sólo se encargan de almacenar documentos XML, sino también objetos XML (elemento, atributo, comentario, instrucción de procesamiento o elemento de texto XML) o incluso otras colecciones. Por tanto, estas colecciones pueden crearse de forma anidada. [11]

XIndice dispone de un contexto/colección principal del que colgarán el resto de colecciones, se trata de *"/db"*. Luego, cada uno de estos objetos almacenados en una colección pueden ser referenciados por una ruta o path.

Por ejemplo, en el caso que se tenga una colección creada en *"/db"* llamada *"mi_colección"* y a su vez otra creada dentro de ésta que se llame *"mi_colección_hija"*. Se podrá acceder a ella con la siguiente ruta:

*/db/mi_colección/mi_colección_hija*

En el caso que se almacene un documento XML llamado *"mi_documentoXML"* en la colección *" mi_colección_hija"* , la forma de acceder a él sería:

*/db/mi_colección/mi_colección_hija/mi_documentoXML*

Hay que tener en cuenta que en el caso que exista una colección, un objeto y un conjunto de documentos con el mismo nombre entonces se establece una preferencia cuando se ejecuta la ruta o el path: primero la colección, depués el objeto XML y por último el documento XML. De esta forma, si se tiene una colección y un documento XML con el mismo nombre y la misma ruta, el documento XML no se podría recuperar.

También es posible acceder a una colección desde un lugar externo a la ubicación de dicha colección. En este caso habría que especificar el host y el puerto del servidor:

*myhost.domain.com:8888/db/mi_colección/mi_colección_hija/mi_documentoXML*

### 2.3.6.5. Operaciones con colecciones.

En este apartado se muestran las diferentes operaciones que se pueden realizar con las colecciones dentro de una base de datos:

1) Añadir una colección:

| **xindice add_collection -c (or context) -n (or name) [-v (or )]** | | |
|---|---|---|
| **Acción** | add_collection | Parámetros:<br>-c: colección donde se crea la nueva colección.<br>-n: el nombre de la nueva colección.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | ac | |
| **Acceso** | administrador | |
| **Ejemplos:** | | |
| **xindice add_collection -c /db -n Gestion_GC** | | |
| **xindice ac -c /db/Gestion_GC –n Pediatría** | | |

**Tabla 12.- XIndice: añadir una colección.**

2) Eliminar una colección:

| xindice delete_collection -c (or context) -n (or name) [-v (or )] | | |
|---|---|---|
| **Acción** | delete_collection | Parámetros:<br>-c: colección donde está la colección a borrar.<br>-n: el nombre de la colección a borrar.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | dc | |
| **Acceso** | administrador | |

**Ejemplos:**

**xindice delete_collection -c /db/Gestion_GC -n Pediatría**
**xindice dc -c /db –n Gestion_GC**

**Tabla 13.- XIndice: eliminar una colección.**

Hay que tener en cuenta que al eliminar una colección se eliminan las colecciones hijas, documentos XML y objetos XML que dependan de ella.

3) Listar una colección:

| xindice list_collections -c (or context) [-v (or )] | | |
|---|---|---|
| **Acción** | list_collections | Parámetros:<br>-c: colección padre de la que se quiere listar sus colecciones hijas.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | lc | |
| **Acceso** | usuario | |

**Ejemplos:**

**xindice list_collections -c /db**
**xindice lc -c /db/Gestion_GC**

**Tabla 14.- XIndice: listar una colección.**

### 2.3.6.6. Operaciones con documentos.

En este apartado se muestran las diferentes operaciones que se pueden realizar con los documentos XML almacenados en una colección:

1) Añadir un documento:

| xindice add_document -c (or context) -f (or file path) [-n (or key to assign to document)] [-v (or )] | | |
|---|---|---|
| **Acción** | add_document | Parámetros:<br>-c: colección donde se almacenará el documento<br>-f: ruta del documento a añadir.<br>-n: nombre a asignar al documento<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | ad | |
| **Acceso** | usuario | |

**Ejemplos:**

**xindice add_document -c /db/Gestion_GC -f /tmp/guía_gripe.xml -n Gripe**
**xindice ad -c /db/Gestion_GC –f /tmp/guía_gripe.xml**

**Tabla 15.- XIndice: añadir un documento a una colección.**

Añadir un documento a una colección requiere de dos parámetros: la colección donde se añadirá el documento (-c) y la ruta del documento a guardar (-f), además de

la extensión. En caso que no se especifique el nombre (-n) con el que se quiere guardar, se almacenará con el nombre que tiene por defecto. No se podrán añadir documentos a colecciones inexistentes.

De esta forma, en el primer ejemplo se añade un documento a la colección *"Gestion_GC"* (colección hija del contexto principal de XIndice), de nombre *"guía_gripe.xml"* y se almacena con el nombre de "*Gripe"*. El segundo ejemplo es exactamente igual que el primero, sólo que al no especificar el nombre con el que se quiere almacenar se le asigna el que tiene por defecto, *"guía_gripe.xml".*

2) Añadir varios documentos:

| **xindice add_multiple_documents -c (or context) -f (or Directory to use ) [-e (or file extenstion )] [-v (or )]** | | |
|---|---|---|
| **Acción** | add_multiple_documents | Parámetros:<br>-c: colección donde se añadirán los documentos<br>-f: ruta de los documentos a añadir.<br>-e: extensión de los documentos a añadir.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | addmultiple | |
| **Acceso** | Usuario | |

**Ejemplos:**

**xindice add_multiple_documents -c /db/Gestion_GC -f /tmp/guías**

**xindice addmultiple -c /db/Gestion_GC -f /tmp/guías -e xml**

**Tabla 16.- XIndice: añadir varios documentos a una colección.**

Esta operación permite añadir varios documentos con una única instrucción. Requiere de dos argumentos: la colección donde se van a añadir y la ruta donde se encuentran almacenados los documentos. Estos documentos se guardarán con el nombre que tienen. Con el parámetro *"-e"* se importa únicamente aquellos documentos con una extensión determinada.

En el primer ejemplo se almacenan todos los documentos que se encuentran en la ruta *"/tmp/guías"* en la colección *"Gestion_GC"* (colección hija de *"/db"*). Mientras que en el segundo sólo se almacenarán aquellos que tengan extensión *"xml".*

3) Eliminar un documento:

| **xindice delete_document -c (or context) -n (or document key) [-v (or )]** | | |
|---|---|---|
| **Acción** | delete_document | Parámetros:<br>-c: colección de donde se eliminará el documento.<br>-n: nombre del documento a borrar.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | dd | |
| **Acceso** | Usuario | |

**Ejemplos:**

**xindice delete_document -c /db/Gestion_GC -n Gripe**

**xindice dd -c /db/Gestion_GC -n guía_gripe.xml**

**Tabla 17.- XIndice: eliminar un documento XML almacenado en una colección.**

4) Recuperar un documento:

| xindice retrieve_document -c (or context) -n (or document key) -f (or file path and name ) [-v (or )] | | |
|---|---|---|
| **Acción** | retrieve_document | Parámetros:<br>-c: colección donde está el documento.<br>-n: nombre del documento a recuperar.<br>-f: ruta donde almacenar el documento.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | rd | |
| **Acceso** | Usuario | |

**Ejemplos:**

**xindice retrieve_document -c /db/Gestion_GC -n Gripe -f /tmp/gripe_guia.xml**
**xindice rd -c /db/Gestion_GC -n guía_gripe.xml –f /tmp/guia_gripe.xml**

**Tabla 18.- XIndice: recuperar un documento almacenado en una colección.**

Esta operación transfiere un documento de una colección y lo almacena en la ruta especificada, donde se debe indicar también el nombre y la extensión con la que se quiere almacenar. En caso que la ruta no exista se creará, y si en esa ruta existe un documento con el mismo nombre se sobrescribirá automáticamente.

5) Importar un directorio: esta operación está diseñada para coger un directorio y crear una colección con su nombre. Los subdirectorios también serán creados como colecciones hijas, duplicando la estructura de directorios.

   Los archivos que se encuentran dentro de los directorios son convertidos en documentos y almacenados en la colección creada. El identificador de cada archivo es su propio nombre.

| xindice import -c (or context) -f (or file path and name ) [-e (or file extenstion )] [-v (or ) ] | | |
|---|---|---|
| **Acción** | import | Parámetros:<br>-c: ruta de la colección donde importar.<br>-f: ruta del directorio a importar.<br>-e: extensión de los documentos a añadir.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | import | |
| **Acceso** | administrador | |

**Ejemplos:**

**xindice import -c /db/Gestion_GC -f /tmp/Pediatría**
**xindice import -c /db/Gestion_GC -f /tmp/Pediatría –e xml**

**Tabla 19.- XIndice: importar un directorio.**

Con la opción *"-e" se* importarán únicamente los documentos que tengan la extensión especificada. Si hay una recopilación con el mismo nombre ésta se sobrescribirá.

6) Exportar un directorio: con esta operación se crea un directorio a partir de una colección, incluyendo las colecciones hijas como subdirectorios. Requiere dos argumentos, la colección y el directorio a exportar.

| xindice export -c (or context) -f (or directory path) [-v (or )] | | |
|---|---|---|
| **Acción** | export | Parámetros:<br>-c: colección a exportar<br>-f: ruta donde se almacenará la colección.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | export | |
| **Acceso** | usuario | |
| **Ejemplos:** | | |
| **xindice export -c /db -f /tmp/Gestion_GC** | | |
| **xindice export -c /db/Gestion_GC/Pediatría -f /tmp/Pediatría** | | |

**Tabla 20.- XIndice: exportar un directorio.**

## 2.3.6.7. Operaciones para indexar una colección.

Indexar una colección consiste en realizar un índice de los documentos XML, objetos XML y colecciones que tiene almacenados para permitir la realización de búsquedas de forma más rápida y eficiente.

En este apartado se muestran las operaciones necesarias para indexar una colección.

1) Añadir un índice a una colección:

| xindice add_indexer -c (or context) -n (or index name ) -p (or index pattern) [--pagesize (or pagesize)] [--maxkeysize (or max key size)] [-t (or indextype)] [-v (or )] | | |
|---|---|---|
| **Acción** | add_indexer | Parámetros:<br>-c: colección a la que se añade el índice.<br>-n: nombre del índice.<br>-p: patrón usado para crear el índice.<br>--pagesize: tamaño de página para el índice.<br>--maxkeysize: tamaño máximo del índice.<br>-t: tipo de dato del índice creado.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | ai | |
| **Acceso** | administrador | |
| **Ejemplos:** | | |
| xindice add_indexer -c /db/Gestion_GC -n indiceGC1 –p Author | | |
| xindice ai -c /db/Gestion_GC -n indiceGC2 –p Diagnostic | | |

**Tabla 21.- XIndice: añadir un índice a una colección.**

2) Permite poner:

- tamaño de página de índice (por defecto 4096).
- tamaño máximo del índice (por defecto 0=ninguno).
- tipo de datos del índice: Los posibles valores son: entero corto, entero largo, en coma flotante de 32 bits y de 64 bits. Caracteres de 8 bits con signo, de 16 bits con signo y 8 bits booleanos.

3) Eliminar un índice de una colección:

| xindice delete_indexer -c (or context) -n (or index name) [-v (or )] | | |
|---|---|---|
| **Acción** | delete_indexer | Parámetros:<br>-c: colección a la que se eliminar el índice.<br>-n: nombre del índice a eliminar<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | di | |
| **Acceso** | administrador | |
| **Ejemplos:** | | |
| xindice delete_indexer -c /db/Gestion_GC -n indiceGC1 | | |
| xindice di -c /db/Gestion_GC -n indiceGC2 | | |

**Tabla 22.- XIndice: eliminar un índice de una colección.**

Aunque se elimine el índice la colección se mantiene.

4) Listar los índices de una colección:

| xindice list_indexers -c (or context) [-v (or )] | | |
|---|---|---|
| **Acción** | list_indexers | Parámetros:<br>-c: colección de la que se quiere listar sus índices.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | li | |
| **Acceso** | administrador | |
| **Ejemplos:** | | |
| xindice list_indexers -c /db/Gestion_GC | | |
| xindice li -c /db/Gestion_GC | | |

**Tabla 23.- XIndice: listar índices de una colección.**

Muestra los índices que tiene una recopilación en una base de datos. Su forma de listar es parecida a la operación de listar recopilaciones.

### 2.3.6.8. XPath.

XPath (XML Path Language) es un lenguaje que permite buscar información dentro de un documento XML. Para ello define una sintaxis para establecer partes en el documento XML, permitiendo navegar a través de sus elementos y atributos, además de manipular de forma básica booleanos, números y cadenas. XPath es uno de los elementos principales del estándar XSLT del W3C.

XPath incluye:

- Una sintaxis para definir las partes de un documento XML.
- Un conjunto de expresiones para seleccionar partes de un documento XML.
- Un conjunto de funciones estándar para tratamiento de cadenas, fechas, valores numéricos, etc.

En el apartado dedicado a XPath se explicará más en detalle esta herramienta.

XIndice implementa un subconjunto de Xpath como lenguaje para la realización de consultas tanto a nivel de documento XML como de colección.

| xindice xpath -c (or context) -q (or query) -s (or prefix=namespace) [-v (or ) ] | | |
|---|---|---|
| **Acción** | xpath | Parámetros:<br>-c: colección sobre la que se va a realizar la consulta.<br>-q: consulta a realizar.<br>-s: indica el espacio de nombres.<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | xpath | |
| **Acceso** | usuario | |

**Tabla 24.- XIndice: consultas XPath.**

A continuación se mostrarán las consultas Xpath soportadas y cuáles son sus resultados. Para ello se hará uso del siguiente documento XML, en el que se realizarán las consultas, y que estará almacenado en una colección llamada "*addressbook*":

```
<?xml version="1.0"?>
<person>
        <fname>John</fname>
        <lname>Smith</lname>
        <phone type="work">563-456-7890</phone>
        <phone type="home">534-567-8901</phone>
        <phone type="cell">345-678-9012</phone>
        <email type="home">jsmith@somemail.com</email>
        <email type="work">john@lovesushi.com</email>
        <address type="home">34 S. Colon St.</address>
        <address type="work">9967 W. Shrimp Ave.</address>
</person>
```

**Ilustración 10.- Ejemplo de documento XML para consultas XPath.**

1) Consultar un documento: en este tipo de consultas se obtienen documentos siguiendo algún criterio. Haciendo uso del ejemplo de documento XML de la Ilustración 10, se quiere encontrar a todas las personas con teléfono móvil, entonces el tipo de consulta a realizar sería:

```
xindice xpath -c /db/addressbook -q "/person/phone[@type='home']"
```

2) Consultar un elemento: estas consultas se realizan para extraer algunos elementos de los documentos. Si se quiere consultar los números de teléfono de casa de todas las personas se indicaría de la siguiente forma:

```
xindice xpath -c /db/addressbook -q "/person[phone/@type='cell']"
```

3) Consultar por String: XPath también soporta las expresiones con ristras de caracteres. En el caso que se quiera el primer número de teléfono de todos las personas de la colección cuyo nombre sea John:

```
xindice xpath -c /db/addressbook –q "string(/person[fname='John']/phone)"
```

4) Consultar por número: para realizar consultas con expresiones numéricas. Si se quiere consultar cuántos números de teléfonos tienen cada una de las personas de la colección:

```
xindice xpath -c /db/addressbook –q "count(/person/phone)"
```

### 2.3.6.9. XUpdate.

XUpdate es el lenguaje que implementa XIndice para la realización de modificaciones a la base de datos (inserciones, extracciones, actualizaciones). Estas actualizaciones deberán ir embebidas en código java para después ejecutarlo conforme al API de Xindice.

| **xindice xupdate -c (or context) -f (or file path) -n (or name) [-v (or ) ]** | | |
|---|---|---|
| **Acción** | xupdate | Parámetros:<br>-c: colección sobre la que se va a realizar la consulta.<br>-f: ruta del documento que contiene la consulta.<br>-n: nombre del documento<br>-v: muestra detallada de la salida producida. |
| **Acción abreviada** | xupdate | |
| **Acceso** | usuario | |
| **Ejemplos:** | | |
| xindice xupdate -c /db/test -f /path/to/xupdate.xml | | |
| xindice xupdate -c /db/test -n document-to-update.xml -f /path/to/xupdate.xml | | |

**Tabla 25.- XIndice: consultas XUpdate.**

Una modificación en un documento XML, es decir, una consulta XUpdate, es representada por el elemento: *xupdate:modifications*. La construcción quedaría similar a la que se muestra:

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
        <xupdate:insert-after select="/addresses/address[1]">
                        ...
        </xupdate:insert-after>
</xupdate:modifications>
```

**Ilustración 11.- XUpdate: estructura de *"xupdate:modifications".***

*xupdate:modifications* define la consulta y puede contener alguno de los siguientes tipos de elementos:

1) *xupdate:insert:* inserta nodos en la estructura de árbol del documento XML. Se utilizan las siguientes instrucciones: *xupdate:insert-before* y *xupdate:insert-after*.

- *xupdate:insert-before:* inserta el nodo antes del nodo padre.
- *xupdate:insert-after*: los inserta después del nodo padre.

Ambas instrucciones requieren de un atributo *select* que especifica el nodo padre a seleccionar.

Las instrucciones: *xupdate:insert-before* y *xupdate:insert-after* pueden contener alguno de estos tipos de elementos: *xupdate:element, xupdate:attribute, xupdate:text y xupdate:commen.*

| Elemento | Función |
|---|---|
| **xupdate:element** | Inserta un nuevo elemento en el documento. |
| **xupdate:attribute** | Inserta un atributo a un elemento del documento. |
| **xupdate:text** | Inserta texto a un elemento del documento. |
| **xupdate:commen** | Inserta un comentario al documento. |

**Tabla 26.- XUpdate: elementos de la instrucción *"insert"*.**

2) *xupdate:append*: añade un elemento al árbol. Requiere de un atributo *select* que especifica el padre del nuevo nodo. La expresión *child* especifica la ubicación del nuevo nodo. *xupdate:append* al igual que el anterior caso puede contener los siguientes tipos de elementos: *xupdate:element, xupdate:attribute, xupdate:text y xupdate:comment.*

3) *xupdate:update*: actualiza el contenido de un nodo existente. Debe tener un atributo *select* que especifique el nodo a actualizar.

4) *xupdate:remove*: elimina un nodo. Debe tener un atributo *select* que especifica el nodo a extraer.

5) *xupdate:rename*: permite renombrar un elemento o un atributo.

Para una mejor comprensión del funcionamiento de XUpdate se verá el siguiemte ejemplo:

Se supone que se tiene el documento XML:

```
<?xml version="1.0"?>
<addresses version="1.0">
        <address id="1">
                <fullname>Andreas Laux</fullname>
                <born day='1' month='12' year='1978'/>
                <town>Leipzig</town>
                <country>Germany</country>
        </address>
</addresses>
```

**Ilustración 12.- Ejemplo XUpdate: documento XML.**

Al que se desea realizar una actualización a través de XUpdate. La actualización consiste en insertar una nueva dirección después de la primera de ellas:

```
<?xml version="1.0"?>
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
        <xupdate:insert-after select="/addresses/address[1]" >
                <xupdate:element name="address">
                        <xupdate:attribute name="id">2</xupdate:attribute>
                        <fullname>Lars Martin</fullname>
                        <born day='2' month='12' year='1974'/>
                        <town>Leizig</town>
                        <country>Germany</country>
                </xupdate:element>
        </xupdate:insert-after>
</xupdate:modifications>
```

**Ilustración 13.- Ejemplo XUpdate: archivo XML con la consulta XUpdate.**

El resultado de aplicar esta consulta XUpdate al documento XML es el que se muestra:

```
<?xml version="1.0"?>
<addresses version="1.0">
        <address id="1">
                <fullname>Andreas Laux</fullname>
                <born day='1' month='12' year='1978'/>
                <town>Leipzig</town>
                <country>Germany</country>
        </address>
        <address id="2">
                <fullname>Lars Martin</fullname>
                <born day='2' month='12' year='1974'/>
                <town>Leizig</town>
                <country>Germany</country>
        </address>
</addresses>
```

**Ilustración 14.- Ejemplo Xupdate: documento XML resultado de la consulta XUpdate.**