**Escuela Colombiana de Ingeniería Julio Garavito**

**Carrera / Semestre:** Ingeniería de Sistemas / 5to Semestre

**Asignatura:** Modelos y Bases de Datos (MBDA)

**Nombre:** Andrés Chavarro, Jesús Pinzón

**Fecha:** 2024/05/09

# Autoestudio 6 – XML Básico

## INVESTIGACIÓN

1. **Tipo de Datos**

* ¿Cómo se define un atributo para que contenga datos en XML?

**R/** Para que un atributo contenga datos en XML se define dentro de una **etiqueta** utilizando la sintaxis **nombre = "valor".** Por ejemplo, a continuación, se define la etiqueta “persona” con los atributos “nombre”, “edad” y “género”:

<Persona nombre="Juan" edad="30" genero="masculino">

… contenido …

</Persona>

* ¿Cómo se inserta una tupla con un valor *en* XML*?*

**R/** En **XML no hay un concepto directo de** **"tupla"** como en las bases de datos relacionales. En su lugar, XML se estructura mediante elementos y atributos. Para insertar un nuevo valor, se necesita crear un nuevo elemento o atributo dentro de la estructura existente. Por ejemplo, a continuación, se muestra cómo agregar un nuevo elemento al siguiente código:

<datos>

    <persona>

        <nombre>Juan</nombre>

        <edad>30</edad>

        <genero>masculino</genero>

    </persona>

</datos>

Para agregar una nueva "tupla", en este caso una nueva persona, simplemente se debe **añadir otro elemento** <persona> dentro del elemento <datos>, con sus respectivos elementos <nombre>, <edad>, y <genero>.

Por ejemplo, para agregar una nueva persona llamada "Pedro" con 28 años y género masculino, se podría añadir así:

<datos>

    <persona>

        <nombre>Juan</nombre>

        <edad>30</edad>

        <genero>masculino</genero>

    </persona>

    <persona>

        <nombre>Pedro</nombre>

        <edad>28</edad>

        <genero>masculino</genero>

    </persona>

</datos>

1. **Consultas**

* ¿Cómo se integra XPath a las expresiones SQL?

**R/** Una consulta **XPath** (Lenguaje de Rutas XML) puede especificarse como **parte de una dirección URL** o **dentro de una plantilla**. El esquema de asignación determina la estructura de este fragmento resultante y los valores se recuperan de la base de datos. Este proceso es conceptualmente similar a crear vistas utilizando la instrucción ***CREATE VIEW*** y escribir consultasSQL en ellas.

Suponiendo que se tiene una tabla llamada “Productos”, la cual contiene una columna llamada “DatosXML”, donde se almacenan los datos de cada producto en formato XML; para recuperar información específica de esos documentos usando XPath, se podría hacer lo siguiente en una BD racional:

SELECT

    XMLQuery('/producto/nombre/text()' PASSING p.DatosXML RETURNING CONTENT) AS nombre,

    XMLQuery('/producto/precio/text()' PASSING p.DatosXML RETURNING CONTENT) AS precio

FROM Productos p

WHERE p.codigo = 'G45';

En este ejemplo:

* ***XMLQuery()*** es una función que ejecuta una consulta XPath en un documento XML.
* ***/producto/nombre/text()*** y ***/producto/precio/text()*** son las expresiones XPath que especifican la ubicación de los datos que se desean extraer dentro del documento XML.
* ***PASSING p.DatosXML*** indica el documento sobre el que se ejecutará la consulta ***XPath***. En este caso, se refiere a la columna DatosXML de la tabla Productos.
* ***RETURNING CONTENT*** especifica que se desea devolver el contenido del nodo seleccionado por la consulta XPath.

Es importante consultar la documentación específica del sistema de gestión de bases de datos para obtener información detallada sobre cómo integrar XPath con SQL en ese entorno particular.

* ¿Qué hacen las funciones *extract, extractValue, existsNode*?

**R/** Las funciones *extract, extractValue y existsNode* son funciones que se utilizan para trabajar con datos XML en bases de datos Oracle. A continuación, se describe cada una de ellas:

* + ***EXTRACT:*** Esta función permite extraer partes específicas de un documento XML, como valores de elementos o atributos. Por ejemplo, una sentencia que usa es esta función es:

SELECT EXTRACT(columna\_xml, '/ruta/a/elemento') AS valor

FROM tabla;

Esta consulta extraería el valor del elemento ‘<elemento>’ que está anidado bajo la ruta ‘/ruta/a/’ en el documento XML almacenado en la columna ‘columna\_xml’ de la tabla ‘tabla’.

* + ***EXTRACTVALUE:*** Esta función era una forma más simplificada de usar ***EXTRACT*** para extraer valores de un documento XML en versiones anteriores de Oracle. A partir de Oracle DB 10g, ***EXTRACTVALUE*** **ha sido marcada como obsoleta**, y se recomienda usar ***XMLQUERY*** o ***XMLTABLE*** en su lugar. La sintaxis de una consulta usando esta función era de la forma:

SELECT EXTRACTVALUE(columna\_xml, '/ruta/a/elemento') AS valor

FROM tabla;

Esta consulta extraería el valor del elemento ‘<elemento>’ bajo la ruta ‘/ruta/a/’ en el documento XML almacenado en la columna ‘columna\_xml’ de la tabla ‘tabla’

* + ***EXISTSNODE*:** Esta función comprueba si un nodo o conjunto de nodos especificados por una ruta XPath existe en un documento XML. Devolverá 1 si el nodo existe y 0 si no existe. Una consulta de ejemplo usando esta función podría ser:

SELECT EXISTSNODE(columna\_xml, '/ruta/a/elemento') AS existe

FROM tabla;

Esta consulta verificaría si existe un nodo ‘<elemento>’ bajo la ruta ‘/ruta/a/’ en el documento XML almacenado en la columna ‘columna\_xml’ de la tabla ‘tabla’, devolviendo 1 si existe y 0 si no existe.

Es importante tener en cuenta que las versiones más recientes de Oracle están migrando hacia el estándar **SQL/XML**, por lo que es posible que en el futuro se prefiera el uso de funciones y métodos basados en este estándar en lugar de estas funciones específicas de Oracle.

* ¿Qué hace la función *XMLTable* y *dbms\_xmlgen.getxmltype*?

**R**/ La función ***XMLTable*** y el paquete ***DBMS\_XMLGEN*** son herramientas que se utilizan para trabajar con datos XML en Oracle. Algunas de sus principales utilidades son:

* + ***XMLTable:*** Función en Oracle que se utiliza para descomponer un documento XML en filas y columnas, permitiendo así tratar el contenido XML como si estuviera almacenado en una tabla relacional. Permite ejecutar consultas XPath sobre un documento XML y devolver los resultados en un formato tabular. La sintaxis de esta función es:

SELECT \*

FROM XMLTable('/ruta/a' PASSING columna\_xml COLUMNS elemento VARCHAR2(100));

* + ***DBMS\_XMLGEN.getXMLType:*** Esta función es parte del paquete ***DBMS\_XMLGEN*** de Oracle y se utiliza para generar un tipo de datos XML a partir de resultados de una consulta SQL. En esencia, la función ejecuta una consulta SQL y devuelve los resultados en forma de un objeto XML. Esto es útil cuando se requiere convertir datos relacionales en un formato XML, por ejemplo, para exportarlos o manipularlos como datos XML en lugar de datos relacionales. La sintaxis de la función es:

SELECT DBMS\_XMLGEN.getXMLType('SELECT \* FROM tabla') AS xml\_data

FROM dual;

1. **Validación**

* ¿Cómo se verifica si un dato XML es bien formado?

**R/** Para verificar si un dato XML es bien formado, es decir, si cumple con las reglas de sintaxis XML y está estructurado correctamente, se pueden utilizar diversas herramientas y métodos:

1. **Validación Manual:** Revisar el XML manualmente para asegurarse de que todas las etiquetas estén correctamente cerradas, que los atributos estén en el formato adecuado y que la estructura general del XML sea coherente.
2. **Analizadores XML:** Existen herramientas y librerías que pueden analizar y validar datos XML automáticamente como por ejemplo **XML Parsers.**
3. **Editores XML**: Muchos editores de texto y herramientas de desarrollo integrado **(IDEs)** tienen funcionalidades integradas para validar documentos XML en tiempo real mientras se editan. Estas herramientas pueden resaltar errores de sintaxis y proporcionar sugerencias para corregirlos. Por ejemplo, si estamos trabajando en un entorno de desarrollo como **Eclipse**, **Atom** o **Visual Studio Code** **(VSC)**, estos suelen tener plugins o extensiones que facilitan la edición y validación de archivos en XML.

* ¿Cómo se verifica si un dato XML cumple un esquema DTD definido?

**R/** Para verificar si un dato **XML** cumple con un esquema **DTD** (Document Type Definition) definido, se puede utilizar herramientas que permitan validar el documento XML contra el DTD. En general, se pueden seguir los siguientes pasos para validar el esquema:

1. **Definir el DTD:** Primero, se debe tener un archivo DTD que defina la estructura y las reglas de validación para el documento XML. El DTD especifica los elementos, atributos y estructuras permitidas en el documento XML, así como las relaciones entre ellos.
2. **Referenciar el DTD en el documento XML:** El documento XML debe hacer referencia al DTD utilizando una declaración DOCTYPE al principio del documento. Por ejemplo:

<!DOCTYPE raiz SYSTEM "ruta/al/dtd">

Donde "raiz" es el nombre del elemento raíz del documento XML y "ruta/al/dtd" es la ruta al archivo DTD.

1. **Utilizar un validador XML:** Se puede utilizar herramientas y programas que ofrezcan funcionalidades de validación XML para verificar si el documento XML cumple con el DTD definido. Algunas opciones comunes incluyen:
   * **Validadores en Línea:** Hay varios validadores XML en línea que permiten cargar un documento XML y un DTD para realizar la validación como, por ejemplo [W3C DTD Validator](https://validator.w3.org/#validate_by_input) y [TRUUGO XML Validator](https://www.truugo.com/xml_validator/).
   * **Herramientas de Comandos:** Algunas herramientas de línea de comandos, como [XmlLint](https://xmllint.com/), permiten validar documentos XML localmente contra un DTD.
   * **Herramientas de Desarrollo Integrado (IDE):** Muchos IDEs de desarrollo tienen funcionalidades integradas para validar documentos XML contra DTDs definidos, lo que permite una validación en tiempo real mientras se edita el XML.
2. **Interpretar Resultados:** Una vez que se haya validado el documento XML contra el DTD, se recomienda interpretar los resultados para determinar si el documento cumple con las reglas definidas en el DTD. Si no hay errores, el documento XML es válido según el DTD. Si hay errores, se deben corregir según las especificaciones del DTD.

En resumen, para verificar si un dato XML cumple un esquema DTD definido, se requiere definir el DTD, referenciarlo en el documento XML, utilizar un validador XML para realizar la validación y luego interpretar los resultados para corregir cualquier error encontrado.

## PRÁCTICA

En este autoestudio vamos a adicionar a la información de la **tabla extras (extra)** de [GuestHouse](https://sqlzoo.net/wiki/Guest_House_Assessment_Easy) un nuevo atributo que permita tener información sobre los siguientes elementos:

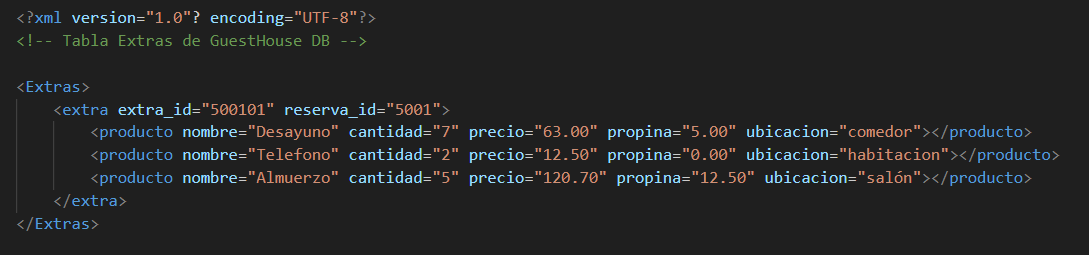
1. **Productos:** Productos consumidos en el extra. De cada uno de ellos se indica el nombre, la cantidad y el precio.
2. **Propina:** Valor de la propina, si existe.
3. **Ubicación:** Lugar donde se atendió el extra: comedor, piscina, habitación, salón.
4. **Definiendo y Poblando**

Adicionar esta nueva información:

1. Expliquen la estructura que van a usar para representar la información con un ejemplo completo.

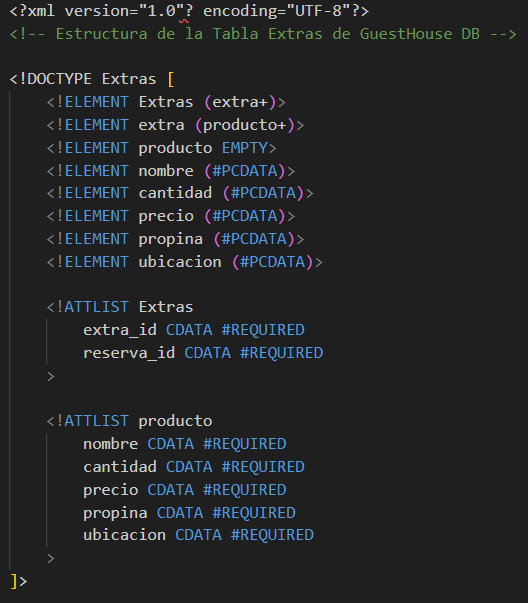
**R/** Para representar la información de la tabla extra de [GuestHouse](https://sqlzoo.net/wiki/Guest_House_Assessment_Easy) en un archivo XML, vamos a considerar a “Extras” como la raíz del documento, la cual va a tener un elemento “extra” con atributos “extra\_id” y “reserva\_id” para representar cada uno de los servicios solicitados por los huéspedes del hotel. Así mismo, cada “extra” va a tener uno o varios elementos “producto” con atributos “nombre”, “cantidad”, “precio”, “propina” y “ubicacion”, describiendo las características de cada pedido.

A continuación, se muestra un fragmento del código en XML de un extra con tres productos y sus respectivos atributos:



1. Modifiquen la creación de las tablas considerando este nuevo atributo.

**Código DTD:**



**NOTA:** Revisar el código completo en [*[GuestHouse DTD.dtd]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Autoestudios%20•/Auto.%206%20•/GuestHouse%20DTD.dtd)

1. Modifiquen las instrucciones para poblar la tabla considerando esta información.

**Código XML:**

****

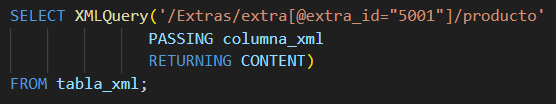
**NOTA:** Revisar el código completo en [*[GuestHouse XML.xml]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Autoestudios%20•/Auto.%206%20•/GuestHouse%20XML.xml)

1. **Consultando**

Escriba las sentencias SQL que permitan conocer:

1. Los productos de un extra específico. (Con toda la información)

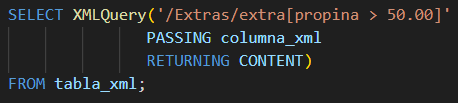
**Código SQL:**



**NOTA:** Revisar el código completo en [*[Consultas.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Autoestudios%20•/Auto.%206%20•/Consultas.sql)

1. Los extras que lograron propinas mayores a un valor dado.

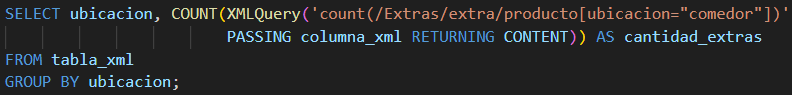
**Código SQL:**



**NOTA:** Revisar el código completo en [*[Consultas.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Autoestudios%20•/Auto.%206%20•/Consultas.sql)

1. El número de extras por ubicación.

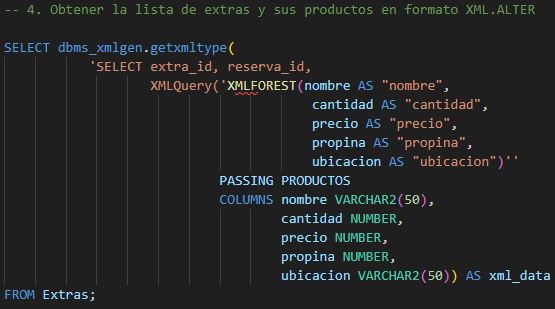
**Código SQL:**

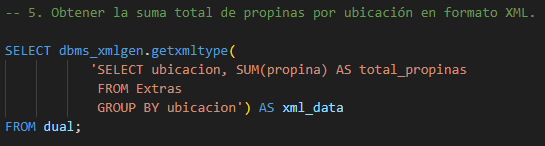
****

**NOTA:** Revisar el código completo en [*[Consultas.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Autoestudios%20•/Auto.%206%20•/Consultas.sql)

1. Propongan e implementen otras dos consultas en las que usen las funciones ***XMLTable*** y ***dbms\_xmlgen.getxmltype.***

**Códigos SQL:**

****

****

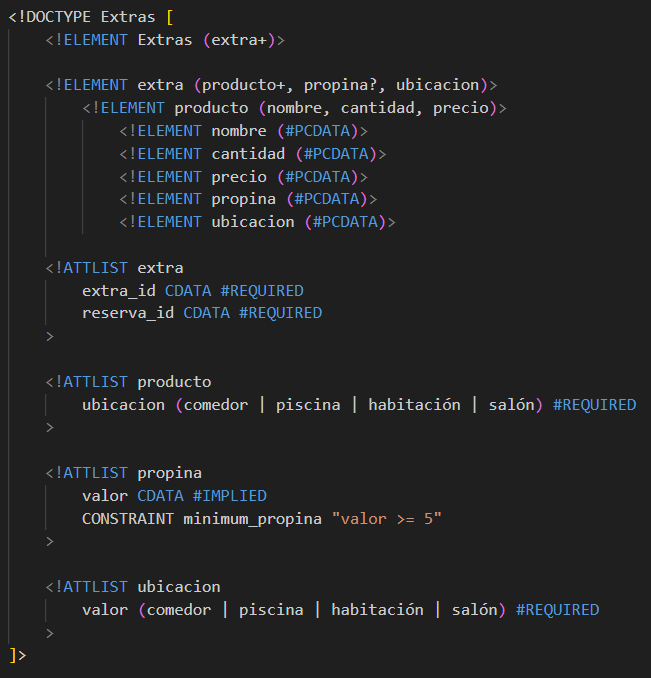
**NOTA:** Revisar el código completo en [*[Consultas.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Autoestudios%20•/Auto.%206%20•/Consultas.sql)

1. **Definiendo Esquema**

Escriba un esquema DTD que garantice que:

1. Los extras deben tener mínimo dos productos.
2. Los extras pueden no tener propinas.
3. La ubicación de los extras sólo puede una de las especificadas.
4. Propongan e implementen otras dos restricciones.
   1. Los atributos “nombre”, “cantidad” y “precio” dentro de <producto> son obligatorios y no pueden estar vacíos.
   2. Las propinas deben ser mínimo de 5.

**Código DTD:**

****

**NOTA:** Revisar el código completo en [*[GuestHouse DTD.dtdl]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Autoestudios%20•/Auto.%206%20•/GuestHouse%20DTD.dtd)

## Bibliografía

McLibre. (2015). *DTD: Definición de Tipo de Documento*. Consultado de: <https://www.mclibre.org/consultar/xml/lecciones/xml-dtd.html>

Microsoft Learn. (2024). *Introducción al uso de consultas XPath (SQLXML 4.0).* Consultado de: <https://learn.microsoft.com/es-es/sql/relational-databases/sqlxml-annotated-xsd-schemas-xpath-queries/introduction-to-using-xpath-queries-sqlxml-4-0?view=sql-server-ver16>

SQLZoo. (2017). *Guest House*. Consultado de: <https://sqlzoo.net/wiki/Guest_House>

Truugo. (2024). *XML Validator.* Página Web: <https://www.truugo.com/xml_validator/>

Tutorialspoint. (2024). *XML – Los Atributos.* Consultado de: <https://www.tutorialspoint.com/es/xml/xml_attributes.htm>

W3C. (2024). *Markup Validation Service.* Página Web: <https://validator.w3.org/#validate_by_input>

XmlLnit. (2020). *XML-DTD Validator.* Página Web: <https://xmllint.com/>