Sólo XML. CATALOGO DE CURSOS

**A. Explorando**

1. Explore el archivo usando un navegador y un editor. ¿Cuáles son las ventajas de uno u otro?

El navegador nos da una vista más intuitiva, ya que nos muestra la separación de las diferentes etiquetas de manera clara y concisa, en cambio el editor no hace esto, el editor, por ejemplo, si nosotros no lo tabulamos, este no se organizará por sí solo, también el navegador separa con colores, las etiquetas de atributos, puede desplegar o plegar las sub-etiquetas que posea, cosa que no hace el editor.

1. Describa la información que encuentra en el archivo.

Es un catálogo de cursos, que son divididos por departamentos; un departamento posee sus diferentes cursos, y estos a su vez tienen diferentes profesores, con sus respectivos nombres y descripción de estos mismos.

**B. Consultas iniciales**

Implemente las siguientes consultas:

1. ¿De cuáles departamentos se tiene información? [a. Nombre con etiqueta] [b. Nombre sin etiqueta] [c. Código]
2. /Course\_Catalog/Department/Title
3. /Course\_Catalog/Department/Title/text()
4. /Course\_Catalog/Department/@Code
5. ¿Cuál es la información del curso Introduction to Databases? [a. Toda la información] [b. Todos los atributos] [c. Todas las etiquetas]

/Course\_Catalog/Department[@Code = 'CS']/Course[@Number= "CS145"]

/Course\_Catalog/Department[@Code = 'CS']/Course[@Number= "CS145"]/@\*

/Course\_Catalog/Department[@Code = 'CS']/Course[@Number= "CS145"]/\*

1. ¿Cuántos cursos están detallados? ¿Cuáles son? [Nombre]

/Course\_Catalog/sum(Department/count(Course[count(/\*) >= 1]))

/Course\_Catalog/Department/Course[count(/\*) >= 1]/Title/text()

1. ¿Cuál es el total de inscripciones de los cursos?

/Course\_Catalog/sum(Department/sum(Course/@Enrollment))

1. ¿De cuáles cursos no se conoce el número de inscritos? [Nombre]

/Course\_Catalog/Department/Course[not(@Enrollment)]/Title/text()

1. ¿Cuáles cursos son iniciales (no tienen prerrequisitos)? [Nombre]

/Course\_Catalog/Department/Course[not(Prerequisites)]/Title/text()

1. ¿Cuáles cursos son finales (no son prerrequisitos de ninguno)? [Nombre]

/Course\_Catalog/Department/Course[not(@Number = /Course\_Catalog/Department/Course/Prerequisites/Prereq)]/Title/text()

1. ¿Cuáles cursos tienen como instructores al director de su departamento? [Nombre] (Los profesores se pueden identificar por su apellido)

/Course\_Catalog/Department/Course[Instructors/Professor/Last\_Name/text() = /Course\_Catalog/Department/Chair/Professor/Last\_Name/text() ]/Title/text()

1. ¿Cuáles cursos de un departamento tienen el mismo profesor? [Nombre] (Los profesores se pueden identificar por su apellido)

/Course\_Catalog/Department/Course[count(Instructors/Professor/Last\_Name/text() = /Course\_Catalog/Department/Course/Instructors/Professor/Last\_Name/text())>1]/Title

1. ¿Cuáles directores de departamento no colaboran en cursos?[Toda la información] (Los profesores se pueden identificar por su apellido)

/Course\_Catalog/Department[not(Course/Instructors/Professor/Last\_Name/text() = /Course\_Catalog/Department/Chair/Professor/Last\_Name/text()) ]/Chair

**B. Consultas quiz Standford**

1. Return all Title elements (of both departments and courses). (16)

/Course\_Catalog//Title/text()

1. Return last names of all department chairs. (3)

/Course\_Catalog/Department/Chair/Professor/Last\_Name/text()

1. Return titles of courses with enrollment greater than 500. (2)

/Course\_Catalog/Department/Course[@Enrollment> 500]/Title/text()

1. Return titles of departments that have some course that takes "CS106B" as a prerequisite. (2)

/Course\_Catalog/Department[Course/Prerequisites/Prereq = 'CS106B']/Title/text()

1. Return last names of all professors or lecturers who use a middle initial. Don't worry about eliminating duplicates. (5)

/Course\_Catalog/Department/Course/Instructors/(Professor|Lecturer)[Middle\_Initial]/Last\_Name/text()

1. Return the title of courses that have a cross-listed course (i.e., that have "Cross-listed" in their description). (2)

/Course\_Catalog/Department/Course[contains(Description, 'Cross-listed')]/Title/text()

1. Return the enrollment of all courses in the CS department. (10)

/Course\_Catalog/Department[@Code = 'CS']/Course/@Enrollment

1. Return last names of instructors teaching at least one course that has "system" in its description and enrollment greater than 100. (2)

/Course\_Catalog/Department/Course[contains(Description, 'system') and @Enrollment>100]/Instructors//Last\_Name

1. Return the title of the course with more than 2 instructors (\*)(1)

/Course\_Catalog/Department/Course[count(Instructors/\*)>2]/Title/text()

1. Return course numbers of courses that have the same title as some other course. (Hint: You might want to use the "preceding" and "following" navigation axes for this query, which were not covered in the video or our demo script; they match any preceding or following node, not just siblings.) (2)
2. Return the courses numbers of courses that have no lecturers as instructors. (\*)(10)

/Course\_Catalog/Department/Course[not(Instructors/Lecturer)]/@Number

1. Return titles of courses taught by the chair of a department. For this question, you may assume that all professors have distinct last names. (1)

/Course\_Catalog/Department/Course[Instructors/Professor/Last\_Name/text() = /Course\_Catalog/Department/Chair/Professor/Last\_Name/text() ]/Title/text()

1. Return titles of courses taught by a professor with the last name "Ng" and by a professor with the last name "Thrun". (\*)(1)

/Course\_Catalog/Department/Course[self::Course/Instructors/\*/Last\_Name="Ng" and self::Course/Instructors/\*/Last\_Name="Thrun"]/Title/text()

1. Return course numbers of courses that have a course taught by Eric Roberts as a prerequisite. (4)

/Course\_Catalog/Department/Course[Prerequisites/Prereq=/Course\_Catalog/Department/Course[Instructors/\*[First\_Name="Eric" and Last\_Name="Roberts"]]/@Number]/@Number

1. Return last names of instructors teachiing at least one course that has system in its description and enrollment greater than 100.(2)

/Course\_Catalog/Department/Course[contains(Description, 'system') and @Enrollment>100]/Instructors//Last\_Name

**C. Consultas propias**

Propongan e implementen cinco consultas propias

1. El nombre de todos los profesores.

/Course\_Catalog/Department/Course/Instructors/Professor/First\_Name

1. El nombre de todos los lectores.

/Course\_Catalog/Department/Course/Instructors/Lecturer/First\_Name

1. El número de todos los cursos.

/Course\_Catalog/Department/Course/@Number

1. El código de los departamentos.

/Course\_Catalog/Department/@Code

1. Cursos dictados por un solo instructor.

/Course\_Catalog/Department/Course[count(Instructors/\*)=1]/Title/text()

**D. Esquema**

Proponga un DTD para estos datos. Para verificar use la herramienta que aparece en moodle.

Realizado en DTD05.xml

**E. Nuevos datos**

Incluya en el archivo la información de tres cursos de la ESCUELA. Verifique que cumple el esquema definido anteriormente

<Course\_Catalog>

<Departament Code = "SE">

<Title> System Engineering </Title>

<Chair>

<Professor>

<First\_Name>Oswaldo</First\_Name>

<Last\_Name>Castillo</Last\_Name>

</Professor>

</Chair>

<Course Number="CS110" Enrollment="2000">

<Title>Modelos Matematicos</Title>

<Description>

Introduccion a la logica matematica.

</Description>

<Instructors>

<Professor>

<First\_Name>Patricia</First\_Name>

<Last\_Name>Alvarez</Last\_Name>

</Professor>

<Professor>

<First\_Name>Wilmer</First\_Name>

<Last\_Name>Garzon</Last\_Name>

</Professor>

</Instructors>

</Course>

<Course Number="CS111" Enrollment="20001">

<Title>Programacion imperativa basica</Title>

<Description>

Introduccion a la programacion y al diseño de algoritmos para poder desarrollar problemas.

</Description>

<Instructors>

<Professor>

<First\_Name>Patricia</First\_Name>

<Last\_Name>Castañeda</Last\_Name>

</Professor>

<Professor>

<First\_Name>Wilmer</First\_Name>

<Last\_Name>Garzon</Last\_Name>

</Professor>

</Instructors>

</Course>

<Course Number="CS112" Enrollment="2002">

<Title>Programacion imperativa modular</Title>

<Description>

Profundizacion en la programacion utilizado diferentes metodos logrando asi una mayor eficacia.

</Description>

<Instructors>

<Professor>

<First\_Name>Carlos</First\_Name>

<Last\_Name>Pinzon</Last\_Name>

</Professor>

<Professor>

<First\_Name>Diego</First\_Name>

<Last\_Name>Satoba</Last\_Name>

</Professor>

</Instructors>

<Prerequisites>

<Prereq>CS111</Prereq>

</Prerequisites>

</Course>

</Departament>

</Course\_Catalog>

**PUNTO DOS. (XML – Oracle) Linked-us. Formación**

Este punto vamos a perfeccionar uno de los atributos de la base de datos.

1. Proponga la estructura XML necesaria para tener la información de las justificaciones de las validaciones. DTD y ejemplos XML OK y XML NoOK. Explique.

Realizado en DTD05.xml

1. Actualice la tabla y las validaciones adicionados a la base de datos. (PoblandoOK, ProblandoNoOK)

Realizado en LAB05.sql

1. Implemente la consulta definida para el Perfil: Consultar mis validaciones

1. Extienda la información de las justificaciones (TJustificación-DTD) y proponga una nueva consulta que ilustre la pertinencia de la información registrada en XML. (Diseño e implementación)

Retrospectiva:

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Hombre)

8/Carlos Murillo, 6/Miguel Sánchez

2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

Está completo.

3. ¿Cuál consideran fue el mayor logro? ¿Por qué?

Hacer todo, ya que era nuestra meta.

4. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

La sintaxis de las consultas en XML, que resolvimos siendo meticulosos en las consultas

5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

Ayudarnos, y trabajar en equipo, seguiremos trabajando duro :3