# Databases (XML & Webtechnologie)



## Databases (XML & Webtechnologie)

| | |
|---|---|
| Richting | Informatica |
| Jaar | Bachelor Informatica Keuzevakken |

## Bespreking

Net zoals bij het vak databases 1 dat in tweede bachelor wordt gegeven volgt dit vak een vrij typische Theorie/Praktijk/Project onderverdeling. De theorie van dit vak gaat eerst en vooral over de structuur van XML bestanden, nadien worden er manieren bestudeerd om de XML boomstructuur te doorlopen. Nadien wordt Xquery, een taal om informatie op te vragen uit XML-databases bekeken en validatietechnieken. Uiteindelijk wordt er nog een ander type query taal gezien, namelijk LiXQuery. Deze taal is een beperkte Xquery taal die dan semantisch en syntactisch bestudeerd wordt.

In de praktijk worden de onderwerpen die in de theorie bekeken werden behandeld. Het project van dit vak houdt in dat een web-applicatie gemaakt moet worden gebruik makend van XML-technieken, AJAX, XSLT, enzovoorts. Wat er precies gemaakt moet worden mag door de studenten zelf gekozen worden zolang de bestudeerde technieken maar toegepast worden.

Het examen zelf is niet echt een theoretisch exmen maar eerder gewoon een groot theoretisch oefeningenexamen (het jaar voor ons was het examen echter wel heel theoretisch dus dit kan blijkbaar variëren). Het examen op zich is ook niet echt moeilijk. Er moet echter wel opgepast worden voor strikvragen aangezien er zo wel een aantal in het examen zitten.

## Puntenverdeling

Theorie + Praktijk: 15/20. Project: 5/20.

## Examenvragen

### Academiejaar 2010 - 2011 - 1ste zittijd

Dit examen bestond uit 5 pagina's :

- Pagina 1
- Pagina 2
- Pagina 3

## Academiejaar 2009 - 2010 - 2de zittijd

Beschouw volgende let-assignments.

```
let $Serves := <Serves>
<t bar="Kelder" beer="Duvel">
<t bar ="Nobel" beer = "Leffe">
<t bar ="Nobel" beer = "Duvel">
</Serves>

let $Visits := <Visits>
<t drinker:="Jan" bar= "Kelder">
<t drinker:="Tim" bar= "Kelder">
<t drinker:="Tim" bar= "Nobel">
</Visits>
```

1. Formuleer in het nederlands de query die uitgedrukt wordt door de functie q1.

   ```
   define function q1($Serves,$Visits)
   {
   $Visits/*[$Serves/*[@beer="Duvel"]/@bar=@bar]/@drinker
   }

   return q1
   Serves,$Visits)
   ```

2. Formuleer volgende query in het nederlands

   ```
   define function q3($Serves,$Visits)
   {
   let $x:= $Serves/*[every $z in $Serves/* satisfies ($z/@bar!= @bar or
   $z/@beer =
   "Duvel")]/@bar
   return $Visits/*[$x=@bar]/@drinker
   }

   return q3($Serves, $Visits)
   ```

3. Vul volgende <?> in. De functie is selectTextnodes() moet de text nodes selecteren van de sequentie $s

   ```
   define function selectTextNodes($s)
   {
   for $e in $s
   return typeswitch ($e)
   case text return <?>
   default return <?>
   }
   ```

4. Vul volgende <?> in. De volgende functie inDocOrder plaatst $e in de sequentie $s, $s en het resultaat zijn in document order.

```
define function inDocOrder($e,$s)
{
if (empty($s))
then $e
else
if ($e is $s[1])
then <?>
else
if ($e << $s[1])
then <?>
else (<?>, inDocOrder($e, <?>))
}
```

5. Schrijf een XSLT template die voor een gegeven (integer) parameter nn de faculteit n!=1∗2∗..∗nn!=1∗2∗..∗n berekent
6. Schrijf een XSLT stylesheet die twee gegeven tree fragmenten merget zodat de elementen met dezelfde naam samengevoegd worden. Je mag er hierbij van uitgaan dat een node geen twee kinderen heeft met dezelfde naam. De volgorde van de kinderen in het eindresultaat is niet van belang. Voorbeeld

```
<a i="1">
<b>
<c/>
<b>
<d/>
</a>
+
<a j="2">
<b>
<e/>
<b>
</a>
=
<a i="1", j="2">
<b>
<c/>
<e/>
<b>
<d/>
</a>
```

7. Schrijf een XML Schema voor een Movie Database. Deze database bevat Movies, Actors en Studio's. Actors hebben een naam, leeftijd en nationaliteit; Movies hebben een titel, jaartal,lengte en een genre; Studio's hebben een naam en een adres. Verder moet kunnen aangeduid worden welke acteurs in welke films spelen (many-to-many), en welke studio's welke films gedraaid hebben(one-to-many). Je mag er van uitgaan dat er geen twee personen met dezelfde naam of twee films met dezelfde titel bestaan. Verder moeten er ook (logische) restricties gelden op onder andere de leeftijd van een acteur en de lengte van een film.

## Academiejaar 2008 - 2009 - 1ste zittijd

1. Schrijf een query in XQuery die als resultaat 3 getallen geeft:
    1. het aantal elementen in het document op de file “tentamen.xml” met naam “subpart”;
    2. het aantal elementen in het document op de file “tentamen.xml” met naam “subpart” die geen kindelen hebben met naam “subpart”;
    3. het aantal elementen in het document op de file “tentamen.xml” die minstens 1 kind hebben met naam “subpart”;

2. Geef het resultaat van de volgende query

```
define function tree2($x)
{
element
  {"tree2"}
  {
    for $p in $x/p
    for $c in $p/ch
    return
      element
        {"pair"}
        { attribute {"parent"} {$p/@parent},
          attribute {"child"} {$c/text()}
        }
  }
}

define function tree3($x)
{
element
  {"tree3"}
  {
    for $p in $x/p
    for $c in $p/ch
    return
      element
        {"p"}
        { attribute {"child"} {$c/text()},
          for $pa in $x/p
          where $pa/ch/text() = $c/text()
          return element {"par"} {$pa/@parent}
        }
  }
}

define function tree4($x)
{
element
  {"tree4"}
  {
    for $p in $x/p
    for $c in $p/ch
    where $c[1] is ($x/p/ch[string( /text())=string($c/text())])[1]
    return
      element
        {"p"}
        { attribute {"child"} {$c/text()},
          for $pa in $x/p
          where $pa/ch/text() = $c/text()
          return element {"par"} {$pa/@parent}
        }
  }
}

let $tree1 :=
<tree1>
  <p parent="John">
```

```
    <ch>Iim</ch>
    <ch>Harry</ch>
  </p>
  <p parent="Nicole">
    <ch>Iim</ch>,
    <ch>Harry</ch>
    <ch>Karel</ch>
  </p>
  <p parent="Iim">
    <ch>An</ch>
  </p>
  <p parent="Vera">
    <ch>An</ch>
    <ch>Bert</ch>
  </p>
  <p parent="An">
    <ch>Carol</ch>
  </p>
</tree1>
return (tree2($tree1), tree3($tree1), tree4($tree1))
```

3. De volgende functie **except2** neemt het 2e2e text kind van het argument $e weg:

```
define function except2($e)
{
  let $sec := $e/text()[2]
  return element
    {name ($e)}
    {for $k in $e/(*|@*|text()) where not($k is $sec) return $k}
}
```

   Verander deze functie zodat ze alle text kinderen van $e wegneemt die dezelfde text bevatten als de voorlaatste text knoop van $e.

4. Wat doet de volgende stylesheet in XSLT? Wat doet deze stylesheet als men de laatste `<xsl:apply-templates/>` wegneemt?

```
<xsl:stylesheet version="2.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<xsl:output method = "xml"/>

<xsl:template match="/">
  <xsl:apply-templates select="*"/>
</xsl:template>

<xsl:template match="*">
  <xsl:element name="{name()}">
    <xsl:for-each select="@*">
        <xsl:element name="{name()}">
          <xsl:value-of select=" "/>
        </xsl:element>
    </xsl:for-each>
    <xsl:apply-templates/>
  </xsl:element>
</xsl:template>

</xsl:stylesheet>
```

5. Onderstaande XSLT stylesheet somt de elementen in een document tree op in pre-order. Pas deze stylesheet aan zodat de elementen

    1. in post-order opgesomd worden ,

    2. in in-order opgesomd worden

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml"/>

<xsl:template match="/">
  <xsl:apply-templates select="*"/>
</xsl:template>

<xsl:template match="*">
  <xsl:element name="name()"/>
  <xsl:apply-templates select="*"/>
</xsl:template>

</xsl:stylesheet>
```

6.

1. Wat is het resultaat van onderstaande XSLT transformatie op het gegeven XML document?

2. Waaraan moet een XML document voldoen opdat de omgekeerde transformatie kan plaatsvinden?

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method = "xml"/>

<xsl:template match="/">
  <xsl:element name="partList">
    <xsl:apply-templates select=".//*"/>
  </xsl:element>
</xsl:template>

<xsl:template match="*">
  <xsl:if test="name()='part'">
    <xsl:element name="part">
      <xsl:attribute name="partId">
        <xsl:value-of select="@partId"/>
      </xsl:attribute>
      <xsl:if test="name(..)='part'">
        <xsl:attribute name="partOf">
          <xsl:value-of select="../@partId"/>
        </xsl:attribute>
      </xsl:if>
    </xsl:element>
  </xsl:if>
</xsl:template>

</xsl:stylesheet>

<?xml version = "1.0"?>
<intList>
<part partId="1">
  <part partId="2">
  </part>
  <part partId="3">
    <part partId="4">
    </part>
  </part>
</part>
<part partId="5">
  <part partId="6">
  </part>
</part>
</intList>
```

7. Beschouw het onderstaande XML document. Geef een mogelijk XML Schema waaraan dit document voldoet. Maak gebruik van keys en references.

```
<?xml version="1.0"?>
<catalog>
  <product>
    <name>cupboard</name>
    <number>3421</number>
    <dimension unit="cm">
      <width>120</width>
      <heigth>180</heigth>
      <depth>34</depth>
    </dimension>
    <descr>oak veneer</descr>
  </product>
  <product>
    <name>mirror</name>
    <number>3455</number>
    <dimension unit="m">
       <heigth>1,5</heigth>
       <width>1</width>
    </dimension>
    <descr>blue frame</descr>
    <supplier>Schmid Co</supplier>
    <supplier>Mayer GmbH</supplier>
  </product>
  <suppliers>
    <supplier>Schmid Co</supplier>
    <supplier>Mayer GmbH</supplier>
  </suppliers>
</catalog>
```

## Academiejaar 2007 - 2008 - 1ste zittijd

1. Geef het verschil tussen de gegeven 2 queries op een XML-document met een gegeven DTD (queries en DTD worden gegeven).
2. 
    1. Geef het resultaat van bovenstaande query (query wordt gegeven).
    2. Welke DTD is niet van toepassing op het gegeven XML-document (document wordt gegeven).
    3. Hoeveel keer komt er Toyota voor in het resultaat van volgende query (query wordt gegeven).
3. Verklaar in LiXQuery de text{} constructor zoals in de paper gegeven (formele semantiek van text{} constructor wordt gegeven).
4. Gegeven een XML-schema. Hoe passen we dit XML-schema aan zodat slechts 3 van de 5 elementen mogelijk voorkomen (is dit mogelijk? Bespreek).