Data & Informatie
Toets 4

SQL/XML, SPARQL, RESTXQ, 'De wiskunde erachter' en Security

Bij de toets mogen geen boeken, aantekeningen of elektronische apparaten worden gebruikt. Er zijn 4 vragen met in totaal 9 punten. Je krijgt 1 punt voor het verschijnen bij de toets.
NB 1: Maak opgaven 1, 2 en 3 op één blad en opgave 4 op een apart blad.
NB 2: Hulp bij de syntax van SQL/XML en SPARQL is gegeven aan het eind van dit document.

Opgave 1: SQL/XML (1,5 punt)

Beschouw het jullie welbekende databaseschema

```
movie (mid:integer, name:text, year:numeric(4,0), plot_outline:text, rating:numeric(2,1))
directs (mid:integer, pid:integer)
person (pid:integer, name:text)
writes (mid:integer, pid:integer)
```

Vraag (a)

Geef een SQL/XML query die een tabel oplevert met één kolom met daarin XML-fragmenten "<movie>…</movie><year>…</year>" voor alle movies ouder dan 2000 met in de plaats van '…' de naam van de movie en het jaar van de movie. Voorbeeld van het beoogde resultaat:

xmlelement
<movie>Godfather, The</movie><jaar>1972</jaar>
<movie>Shawshank Redemption, The</movie><jaar>1994</jaar>
<movie>Schindler's List</movie><jaar>1993</jaar>
<movie>Casablanca</movie><jaar>1942</jaar>

Antwoord

```
SELECT XMLFOREST(name AS naam, year AS jaar)
FROM movie
WHERE year < 2000
```

of

```
SELECT XMLCONCAT(XMLELEMENT(NAME naam, name), XMLELEMENT(NAME jaar, year))
FROM movie
WHERE year < 2000
```

Vraag (b)

Geef een SQL/XML query die een tabel oplevert met één kolom met daarin voor alle jaren een XML-element 'filmsuit' met daarin de namen van alle films uit dat jaar. Het jaar zelf is een attribuut van filmsuit. Voorbeeld van het beoogde resultaat.

xmlelement
<filmsuit jaar="1990"><naam>Goodfellas</naam><naam>Miller's Crossing</naam></filmsuit>
<filmsuit jaar="1948"><naam>Treasure of the Sierra Madre, The</naam><naam>Ladri di biciclette</naam></filmsuit>
<filmsuit jaar="1945"><naam>Enfants du paradis, Les</naam></filmsuit>
<filmsuit jaar="1992"><naam>Reservoir Dogs</naam><naam>Unforgiven</naam><naam>Player, The</naam></filmsuit>

Antwoord

```
SELECT
   XMLELEMENT(NAME filmsuit,
      XMLATTRIBUTES(year AS jaar),
      XMLAGG(XMLELEMENT(NAME naam, name))
   )
FROM movie
GROUP BY year
```

Opgave 2: SPARQL (1,5 punt)

Gegeven dezelfde data set als bij het practicum over koninklijke families. Hieronder nogmaals alle mogelijke relaties in de data.

Type van object	*Naam van relatie*	*Type van waarde / object*
a:Birth	a:date	
a:Birth	a:place	
a:Death	a:date	
a:Death	a:place	
a:Family	a:divorce	a:Divorce
a:Family	a:marriage	a:Marriage
a:Individual	a:birth	a:Birth
a:Individual	a:death	
a:Individual	a:name	
a:Individual	a:sex	
a:Individual	a:title	
a:Individual	a:childIn	a:Family
a:Individual	a:spouseIn	a:Family
a:Marriage	a:date	
a:Marriage	a:place	
elk object	rdf:type	

Vraag (a)

Gegeven de onderstaande SPARQL query

```
SELECT ?d WHERE
{       ?m a:date ?d .
        ?m a:place ?n .
        FILTER regex(?n, "Netherlands")
}
```

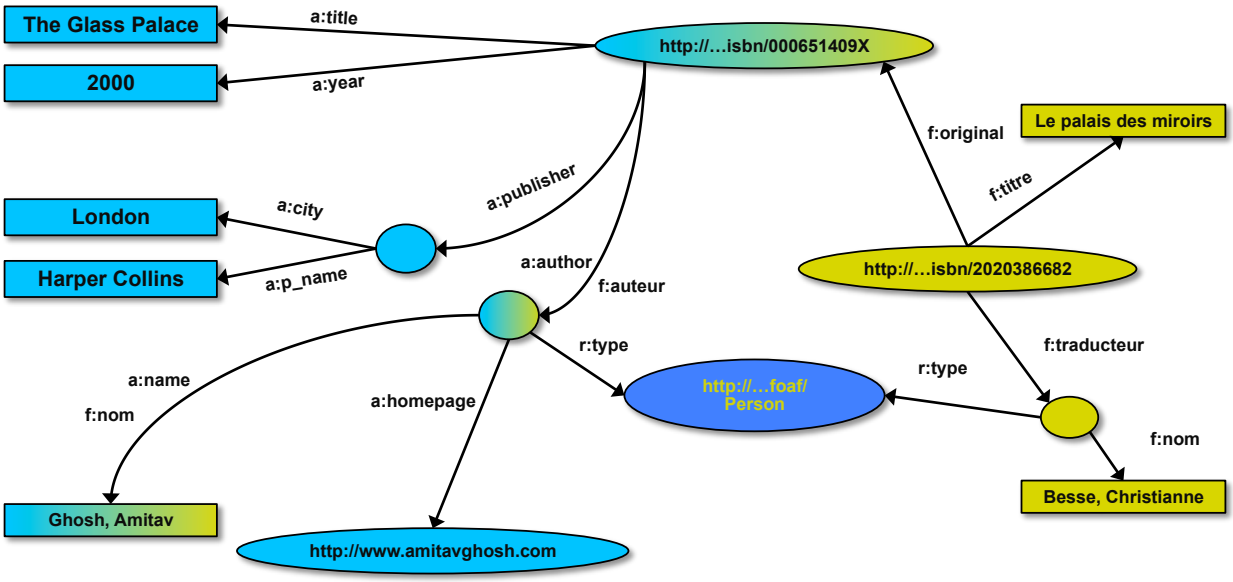
De query levert een tabelletje op met een aantal datums erin. Leg zo exact mogelijk uit waarvan dit datums zijn.

Antwoord

Dit zijn datums van geboortes, sterfgevallen en huwelijken in Nederland. De objecten ?m moeten namelijk een relatie a:date en a:place hebben en dat geldt alleen voor types a:Birth, a:Death en a:Marriage.

Vraag (b)

Op ’t college is de onderstaande RDF-data aan de orde geweest over een boek “The Glass Palace” geschreven in 2000 door “Amitav Ghosh” en de vertaling daarvan in ’t Frans “Le palais des mirroirs” vertaald door “Christianne Besse”.



Schrijf een SPARQL-query die gegeven deze data de titels van alle Franse vertalingen van het boek “The Glass Palace” oplevert.

Antwoord

```
SELECT ?fransetitel WHERE {
   ?fransboek f:titre ?fransetitel .
   ?fransboek f:original ?origineelboek .
   ?origineelboek a:title “The Glass Palace”
}
```

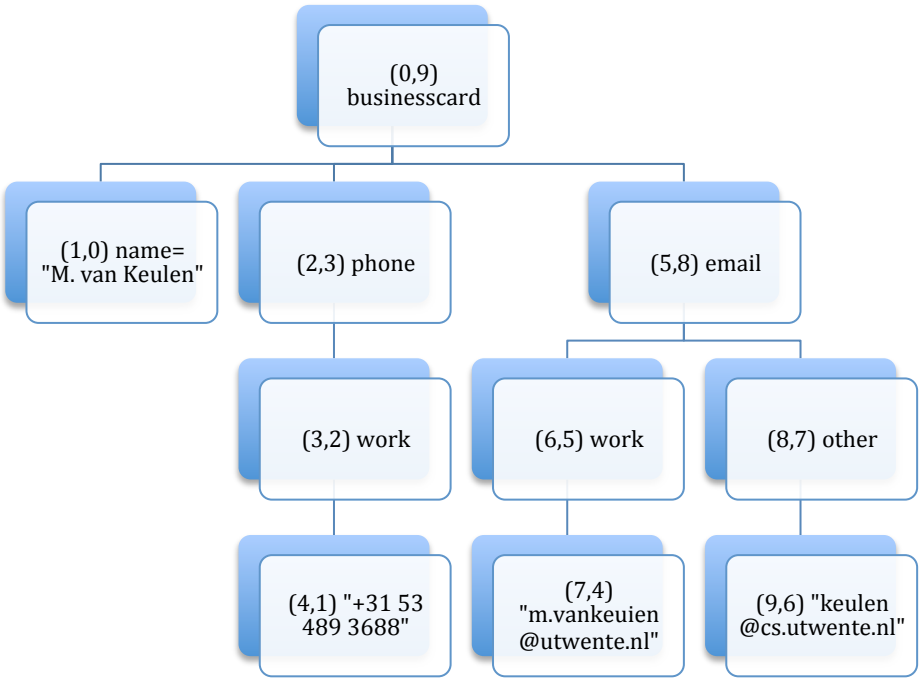
Opgave 3: ‘De wiskunde erachter’ (1,5 punt)

Vraag (a)

Gegeven het onderstaande XML-documentje. Teken de bijbehorende boom. Teken ook in de boom alle pre-order en post-order ranks.

```
<businesscard name=”M. van Keulen”>
<phone><work>+31 53 489 3688</work></phone>
<email><work>m.vankeulen@utwente.nl</work><other>keulen@cs.utwente.nl</other></email>
</businesscard>
```

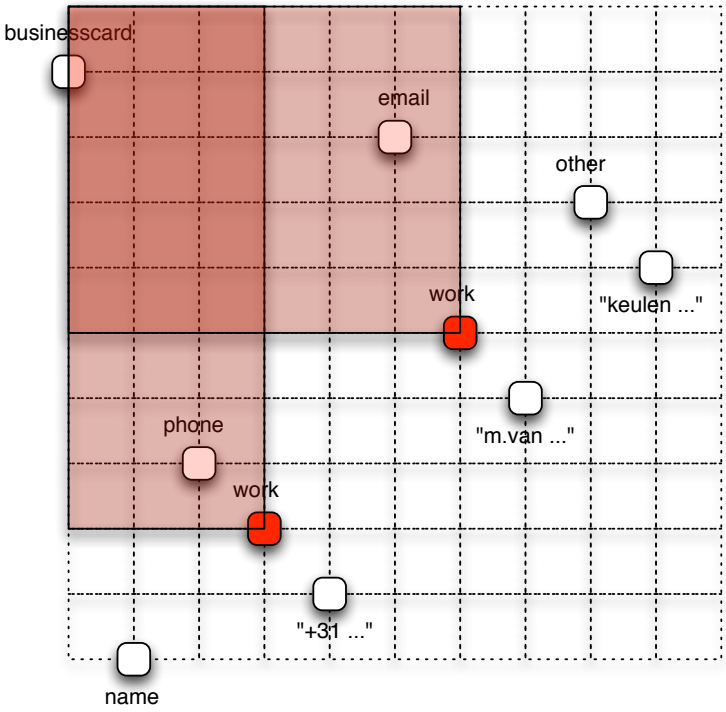
Antwoord



Let op: aan attribuut met samen het haar waarde één node in de boom. Een element met tekst ertussen zijn twee nodes: een element-node en een text-node. Alle nodes in de boom tellen meer voor de preorder- en postorder-nummering.

Vraag (b)

Teken de pre/post-plane voor dit documentje en arceer daarin het gebied waar je alle *ancestors* van alle `work`-elementen kunt vinden.



Vraag (c)

Gegeven de onderstaande drie documenten. Bereken P('hoedje' | d_i) voor elk van de drie documenten.

d_1: een, twee, drie, vier
d_2: hoedje van, hoedje van
d_3: een, twee, drie, vier, hoedje van papier

Antwoord

P('hoedje' | d_1) = 0/4 = 0
P('hoedje' | d_2) = 2/4 = 1/2
P('hoedje' | d_3) = 1/7

Vraag (d)

Gegeven concepten Man, Vrouw en relaties heeftKind en heeftOuder. We willen de concepten van Vader, Moeder, Kind en Persoon definiëren. Hieronder een poging daartoe die nog verre van compleet is.

Vader ≡ Man ∩ ∃ heeftKind.Kind ∩ ∀ heeftKind.Kind
Moeder ≡ Vrouw ∩ ∃ heeftKind.Kind ∩ ∀ heeftKind.Kind
Kind ≡ Persoon ∩ ∃ heeftOuder.Persoon
Persoon ≡ Man ∪ Vrouw ∪ Kind

(i) Laten bovenstaande definities een familie toe waarbij beide ouders van hetzelfde geslacht zijn? Leg uit waarom.

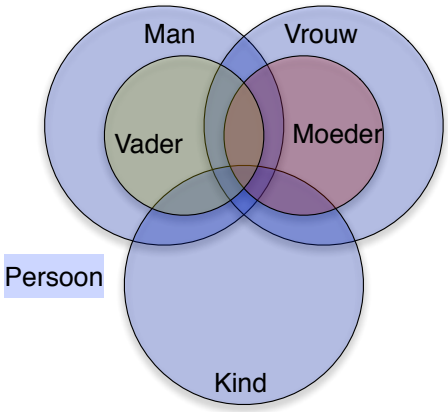
Antwoord

Een kind moet minimaal één ouder hebben van type Persoon. Er staat nergens precies hoeveel ouders dat kind mag hebben, dus een één-ouder gezin, als ook een gezin met 42 ouders is volgens deze specificatie prima. Bovendien is de beperking dat er één ouder van type Persoon is, dus als er van die 42 ouders er eentje een man of een vrouw is, dan is 't al goed, maw een gezin met een twee vaders, twee moeders of zelfs een man en 41 kikkers en eenden is correct volgens deze specificatie.

(ii) Men heeft nagelaten om te definiëren dat Vader en Moeder subklassen zijn van Persoon. Leg uit waarom dit uit de bovenstaande definities volgt.

Antwoord

Hieronder staat mbv het cirkel-diagram de situatie geïllustreerd rondom Persoon, Man, Vrouw, Kind, Vader, Moeder. Alles wat een blauwe achtergrond heeft is een Persoon, d.w.z. alle instanties die in deze verzamelingen vallen zijn allemaal van type Persoon. Zoals je ziet zijn de verzamelingen van alle Vaders en van alle Moeders deelverzameling van de verzameling van alle Personen, maw het zijn subklassen.
Merk op dat er nergens nog gespecificeerd staat dat iemand niet tegelijkertijd Man en Vrouw kan zijn en andere belangrijke kennisregels.



Opgave 4: Security (4,5 punt)
NB: MAAK DEZE OPGAVE OP EEN APART BLAD!

Vraag (a)

For each of the following fuzzers:

A. a mutation-based fuzzer;
B. a protocol fuzzer;
C. a symbolic execution based fuzzer.

In a few lines, (i) describe a system that you would fuzz using this fuzzer, and (ii) explain why this fuzzer is a good choice.

Vraag (b)

Given the following code:

```
1: void foo (signed char x, signed char y) {
2:   signed char buf[5], z;
3:
4:   if (x < 0 | y < 0){
5:     return;
6:   }
7:   z = x + y + 5;
8:   if (z <= 5){
9:     z = z * 2;
10:    buf[z] = y;
11:  }
12:}
```

A signed char can obtain any value from [-127,127].

(i) Write down the assertion that you would add just before line 10 in order to ensure memory safety.
(ii) Provide the symbolic values and the path predicate for symbolic execution that can be used to test memory safety at line 10.
(iii) Provide an assignment to x and y that causes a buffer overflow.

Vraag (c)

For the SQL code below, write down a statement to change the email jan@myweb.com with hacker's email, hacker@mycompany.com, assuming that the hacker knows that there is a table called "table" with a field "email".

```
SELECT data
          FROM table
              WHERE Emailinput = '$email_input';
```

Instead of a proper email address, an attacker could enter the following code:
X';Update table set email='hacker@mycompany.com' where email=' jan@myweb.com';

As long as the idea how to achieve the goal is clear, the syntax of the SQL statement does not matter.

Vraag (d)

Explain how hash functions are used in signatures by writing down 2 properties of hash functions.

Hash functions are functions that generate fixed output for inputs of any size. The output is random looking and unique in the sense that it is extremely difficult to find another input that produces the same output. Any small change in the input changes the output (the has value) significantly.

Hash functions are used for digital signatures to check the integrity of the documents. The hash of the document is signed. Upon receipt of the signature and the document, the hash value in the signature and the hash value of the received document are compared. If they are same, this means that the document is not altered.

Syntax for SQL/XML

In de SELECT-clause van een SQL-query kun je de volgende functies gebruiken om XML-structuren te construeren (versimpeld).

xml ::= elem | attr | forest | aggr
elem ::= `XMLELEMENT ( [NAME` name `,`] (sql | xml)* `)`
attr ::= `XMLATTRIBUTES ( [NAME` name `,`] sql+ `)`
forest ::= `XMLFOREST (` (sql | xml)+ `)`
aggr ::= `XMLAGG (` xml `)`

'sql' staat voor een SQL-variabele, naam van een SQL-attribuut of een willekeurige SQL-query. name is de naam voor het te construeren element of attribuut. Als er geen NAME-clause gespecificeerd is, dan wordt naam van de sql-expressie gebruikt als default.

Syntax voor SPARQL

De syntax voor een SPARQL-query is als volgt (versimpeld).

query ::= prefix* select
prefix ::= `PREFIX` name`:` `<` url `>`
select ::= `SELECT` [`DISTINCT`] var* `WHERE {` pattern+ `}`
var ::= `?`name
qname ::= (name | url)`:`name
pattern ::= triplepattern | filter | optional | union
triplepattern ::= (var | qname) (var | qname) (var | qname | value) .
optional ::= `OPTIONAL {` pattern+ `}`
union ::= `{` pattern+ `}` `UNION {` pattern+ `}`
filter ::= `FILTER (` expr `)`
expr ::= comparison | regex
comparison ::= (var | qname | value) compop (var | qname | value)
compop ::= `=` | `!=` | `<` | `>` | `<=` | `>=`
regex ::= `regex(` var, value `)`

'url' staat voor een willekeurige URL of URI. 'value' staat voor een waarde, dwz een getal of een string.