Data & Informatie voorbeeldvragen voor Toets 4

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

Bij de toets mogen geen boeken, aantekeningen of elektronische apparaten worden gebruikt. De onderstaande vragen zijn representatieve voorbeeldvragen zoals je die op de 'echte' toets zou mogen verwachten. Merk wel op dat het er meer zijn dan op de 'echte' toets, dwz de lengte is niet representatief.
NB: Hulp bij de syntax van SQL/XML en SPARQL is gegeven aan het eind van dit document. Op de daadwerkelijke toets krijg je deze hulpinformatie er ook bij.

Opgave 1: SQL/XML

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-elementen "<director>…</director>" voor alle directors (dwz. personen die in de directs-tabel voorkomen) met in de plaats van '…' de naam van de director. Voorbeeld van het beoogde resultaat:

xmlelement
<director>Francis Ford Coppola</director>
<director>Frank Darabont</director>
<director>Steven Spielberg</director>
<director>Michael Curtiz</director>

Vraag (b)

Geef een SQL/XML query die het bovenstaande oplevert als een tabel met één rij met daarin een compleet XML-fragment met root 'directors' en daarbinnen alle directors zoals hierboven. Voorbeeld van het beoogde resultaat:

xmlelement
<directors><director>Francis Ford Coppola</director><director>Frank Darabont</director><director>Steven Spielberg</director></directors>

Vraag (c)

Geef een SQL/XML query die een tabel oplevert met één kolom met daarin voor alle directors XML-elementen 'director' die de zowel de naam van de director (element 'name') als de titels van alle movies die hij heeft gedirect (element 'movie'). Voorbeeld van het beoogde resultaat.

xmlelement
<director><name>Francis Ford Coppola</name><movie>Godfather, The</movie><movie>Godfather: Part II, The</movie></director>
<director><name>Frank Darabont</name><movie>Shawshank Redemption, The</movie><movie>Green Mile, The</movie></director>
<director><name>Steven Spielberg</name><movie>Schindler's List</movie><movie>Raiders of the Lost Ark</movie><movie>Saving Private Ryan</movie></director>

<director><name>Michael Curtiz</name><movie>Casablanca</movie><movie>Adventures of Robin Hood, The</movie></director>

Opgave 2: SPARQL

Gegeven dezelfde data set als bij het practicum over koninklijke families.

Vraag (a)

De onderstaande SPARQL query

```
SELECT ?n WHERE
{  ?beatrix a:name "Beatrix of Netherlands //" .
    ?beatrix a:spouseIn ?family .
    ?husband a:spouseIn ?family .
    ?husband a:name ?n
}
```

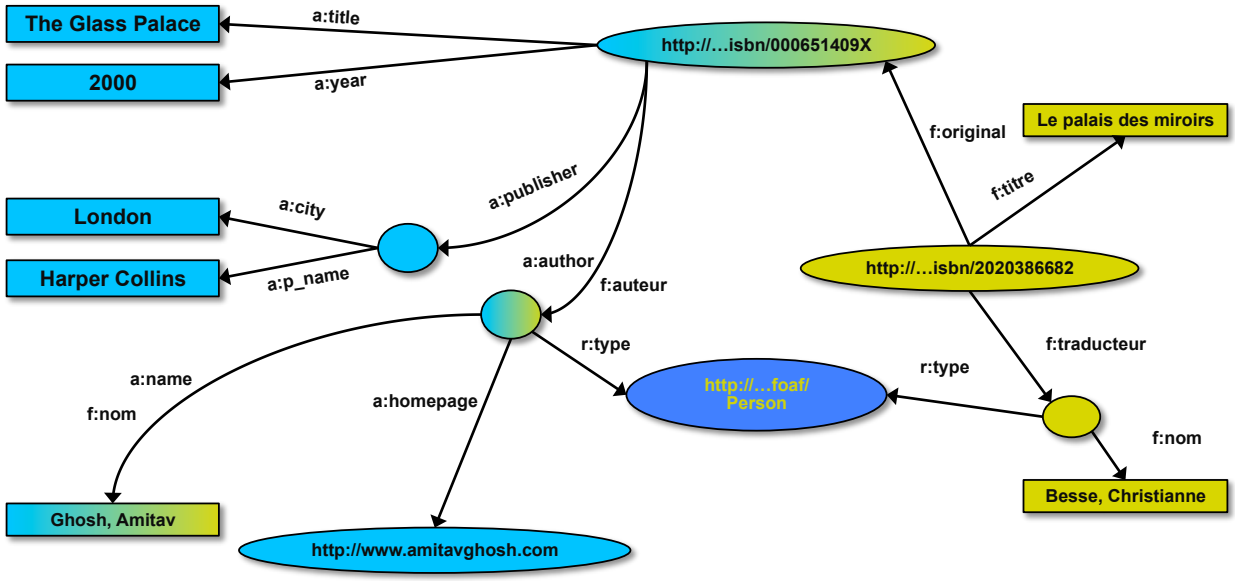
is een poging om de naam van de man van Beatrix te produceren. De query produceert echter het onderstaande resultaat:

xmlelement
"Beatrix of_Netherlands //"
"Claus /von_Amsberg/"

Verklaar waarom ook Beatrix in het resultaat zit en niet alleen haar man Claus; '?n' is toch alleen maar gerelateerd aan '?husband' en niet aan '?beatrix'?

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 boeken van het jaar 2000 oplevert met bij elk een eventuele Franse vertaling als die er is.

Opgave 3: RESTXQ

Op het college is de onderstaande webservice aan de orde geweest:

```
declare %rest:path("test/{$a}/{$b}")
```

```
        %rest:GET
        %output:method("xhtml")
function page:hello($a as xs:integer, $b as xs:integer)
{
<html>
<h1>Optelling</h1>
<p>De som van {$a} en {$b} is {$a+$b}.</p>
</html>
};
```

Stel onze BaseX database bevat een document van de volgende vorm:

```
<getallen>
   <getal naam="foo">42</getal>
   <getal naam="bar">43</getal>
   <getal naam="baz">15</getal>
</getallen>
```

Wat moet er aan de RESTXQ webservice worden veranderd, zodat je op basis van een URL http://localhost:8984/test/foo/bar de onderstaande output krijgt?

Optelling

De som van foo en bar is 42+43=85.

Opgave 4: 'De wiskunde erachter'

Vraag (a)

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

```
<todolist name="lijst">
<item status="done">Boodschappen doen</item>
<item status="todo">Kadootje kopen voor <b>Henk</b>!!!</item>
</todolist>
```

Vraag (b)

Stel een element heeft pre-order rank 14 en postorder rank 5. Aan welke conditie(s) mbt pre-order en post-order rank moeten nodes voldoen om een *ancestor* te zijn van dit element?

Vraag (c)

Gegeven de onderstaande drie documenten (bron NOS teletekst, pagina 801). Bereken P('zege' | d_i) voor elk van de drie documenten.

d_1: Ruime zege Duitsland op Portugal
d_2: Nigeria en Iran scoren niet
d_3: VS neemt wraak met zege op Ghana

Vraag (d)

Stel we willen in het model opnemen dat de belangrijkheid van een document groter is naarmate het meer woorden met een hoofdletter bevat. Document d_1 met 3 hoofdletters is dus belangrijker dan d_2 en d_3 die elk 2 woorden met hoofdletter hebben. In welk van de onderstaande kansen is dit verdisconteerd?

$$P(D|T_1,\ldots,T_n),\ P(T_1,\ldots,T_n|D),\ P(D),\ P(T_1,\ldots,T_n)$$

Vraag (e)

Gegeven concepten Man, Kind en relatie heeftKind, willen we het concept van Vader definiëren. Welke van de onderstaande formules is een correcte beschrijving van wat wij onder een vader verstaan? Leg uit waarom.

(i) $\text{Vader} \equiv \text{Man} \cap \exists\ \text{heeftKind.Kind}$
(ii) $\text{Vader} \equiv \text{Man} \cup \exists\ \text{heeftKind.Kind}$
(iii) $\text{Vader} \equiv \text{Man} \cap \forall\ \text{heeftKind.Kind}$
(iv) $\text{Vader} \equiv \text{Man} \cup \forall\ \text{heeftKind.Kind}$
(v) $\text{Vader} \equiv \text{Man} \cap \exists\ \text{heeftKind.Kind} \cap \forall\ \text{heeftKind.Kind}$
(vi) $\text{Vader} \equiv \text{Man} \cup \exists\ \text{heeftKind.Kind} \cup \forall\ \text{heeftKind.Kind}$

Vraag (f)

Kies één van de andere formules van vraag (e) dan je gekozen hebt voor je antwoord. Geef een voorbeeld van een object of persoon uit de werkelijkheid die wel aan je gekozen formule voldoet, maar geen 'Vader' is.

Opgave 5: Security

Vraag (a)

Give a brief description of the structure of a certificate. Explain how an expired certificate can be exploited by a malicious entity.

Vraag (b)

Imagine that hashes will be computed for each car part identifier in a database using a function that generates a hash of 80 bits. Elaborate **briefly** on the maximum number of car part hashes that can be stored in the database without collusion.

Vraag (c)

Given the follow code:

```
1: void foo (unsigned char x, unsigned char y) {
2:   unsigned char buf[3], z;
3:
4:   z = x + 5;
5:   z = z * 2;
6:   if (z <= 4){
7:     buf[z] = y;
8:   }
9:}
```

An unsigned char can obtain any value from [0,255].

(i) Write down the assertion that you would add just before line 7 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 7.
(iii) Provide an assignment to x and y that causes a buffer overflow.

Vraag (d)

Given the following strategy for mutation-based fuzzing:

1. Google for .pdf
2. Crawl pages to build a test set

3. Use mutation-based fuzzing tool to:
    a) Load pdf file
    b) Mutate the file
    c) Feed to program
    d) Record if it crashed and what crashed it

(i) Add steps to the strategy that use the code of "program" to improve the fuzzing based on what has been discussed in the lecture.

(ii) Explain what this improves, how it is improved, and a possible downside of this new strategy.

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.