# Data & Informatie voorbeeld van Toets 3
# Antwoorden

## Triggers, Transactions, Web Services and XML

## Antwoord 1

***Antwoord a)***

Person ( id , name , primary key( id) )

Student (id, name, foreign key(id) references Person (id) on delete cascade)

Het deel '*on delete cascade*' zorgt voor actie 1, het deel '*foreign key( id ) references Person ( id )*' zorgt voor actie 2. Er werd niet gevraagd om in *Student* de *id* tot *primary key* te verklaren, en ook niet om de actie die opgeroepen wordt door '*on update cascade*' toe te voegen aan de foreign key constraint.

***Antwoord b)***

```
create trigger OnPersonDelete
        after delete on Person
        referencing old as DeletedPerson
        for each row
        delete from Student s where s . id = DeletedPerson . id ;

create trigger OnStudentInsert
        before insert on Student
        referencing new as NewStudent
        for each row
        when ( NewStudent . id not in (select id from Person ))
        abort;
```

Kleine variaties in de syntaxis worden niet fout gerekend. Varianten met statement level granularity ('for each state ment') zijn ook mogelijk:

```
create trigger OnPersonDelete2
        after delete on Person
        referencing old table as DeletedPersons
        for each statement
        delete from Student s
        where s . id in (select id from DeletedPersons );

create trigger OnStudentInsert 2
        before insert on Student
        referencing new table as NewStudents
        for each statement
        when (exists ((select id from NewStudents ) except (select id from Person )))
        abort;
```

# Antwoord 2

***Antwoord a)*** Een interleaving van T1 en T2 die er zo uit ziet:

… read2(x) … write1(x) … write2(x) …

of

… read1(x) … write2(x) … write1(x) …

***Antwoord b)*** Bijvoorbeeld:

read1(x); read1(y); write1(x) … read2(x); … abort1 (abort/rollback van T1)

De toevoeging *abort*1 is niet noodzakelijk voor een goed antwoord. Wanneer ‘abort/rollback’ wordt vervangen door ‘commit’, dan is er in feite nog steeds sprake van een dirty read, hoewel die in dat geval niet erg is omdat de executievolgorde dan serialiseerbaar is.

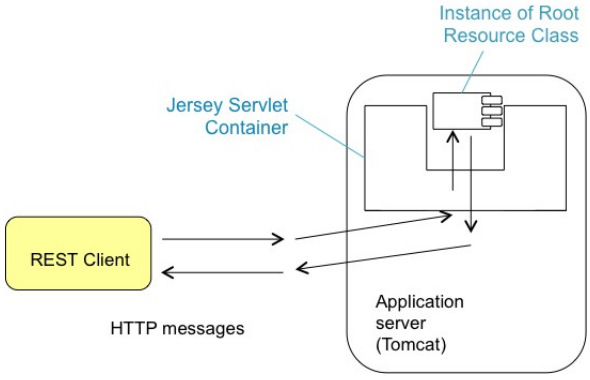
***Antwoord c)*** Zet het isolatieniveau van T1 op READ COMMITTED of sterker.

# Antwoord 3

***Antwoord a)*** Een web applicatie die alleen HTML pagina's oplevert stuurt naar de web client informatie samen met de formattering waarmee de informatie aan de eindgebruiker (een mens) vertoond kan worden. In tegenstelling, de nadruk bij een RESTful service is op de informatie en machine-to-machine communicatie, dus in principe stuurt een RESTful service een stuk informatie, gecodeerd op een bepaald manier zodat een client (een computer programma) in staat is om deze informatie te interpreteren en verwerken, niet per se om aan een mens te vertonen.

Een voorbeeld is de bookQuote applicatie van het college. In de web applicatie versie, de BookQuote servlet stuurt een HTML pagina naar de client (een browser), maar in de RESTful service implementatie de BookQuote service stuurt de prijs van het boek als antwoord naar de client (een programma), in een bericht die in XML of JSON gecodeerd wordt.

***Antwoord b)*** Met Jersey wordt een RESTful service geïmplementeerd als een web applicatie, waarin een speciale Servlet de HTTP request opvangt en naar de juiste object stuurt. In Jersey is deze structuur aangeduid als een 'Jersey Servlet container'. De RESTful web applicatie wordt in Tomcat geïnstalleerd ('deployed') en draait met gebruik van de Tomcat runtime environment.



***Antwoord c)*** Jersey maakt gebruik van annotaties. De klasse met de methodes die aangeroepen worden voor het afhandelen van een HTTP request (de zogenaamde 'root resource class') is geannoteerd met de Path annotatie, waarin de URL pad naar deze klasse wordt gedefinieerd. Sommige methodes in deze 'root resource class' worden dan geannoteerd met de HTTP methode die ze afhandelen (GET, PUT, POST of DELETE) en de codering die ze begrijpen (Consumes) en produceren (Produces). Jersey gebuikt de Path informatie om de juiste klasse (object) te bepalen, door te kijken naar de URL van de HTTP request. Voor het bepalen van de juiste methode van de 'root resource class' die aangeroepen moet worden, gebruikt Jersey de HTTP methode in de HTTP request, en de coderingen van de data (inhoud van HTTP request en verwachte codering van de HTTP response). Wanneer een match ontstaat tussen de inhoud van de HTTP request en een methode van de 'root resource class' wordt deze methode door Jersey aangeroepen.

## Antwoord 4

***Antwoord a)*** Ja, het is well-formed. Alle opening-tags hebben een bijbehorende closing-tag en ze zijn netjes genest.

***Antwoord b)*** Dit is een voorbeeld van meer data-centric XML, want de structuur is vrij strikt; het zou zo uit een relationele database kunnen komen.

***Antwoord c)*** `//game[@teama="GER" and @teamb="CRC"]`

***Antwoord d)***

```
for $g in //game[@teama="GER" or @teamb="GER"]
let $score :=
  if ($g/@teama="GER")
  then ($g/result/@teama, $g/result/@teamb)
  else ($g/result/@teamb, $g/result/@teama)
let $resultaat :=
  if ($score[1] > $score[2]) then "Gewonnen"
  else if ($score[1] = $score[2]) then "Gelijkspel"
  else "Verloren"
 group by $resultaat
 return <groep resultaat="{$resultaat}">{$g}</groep>
```

***Antwoord e)***

```
replace node //game[@teama="ECU" and @teamb="GER"]/notplayed with (<result
teama="1" teamb="1"/>,<review>foo bar</review>
```

***Antwoord f)***

```
//game[. contains text {"Germans","beat"} all not in "not beat"]
```

Het gaat bij deze antwoorden niet om de volledig correcte syntax, maar wel om de juiste denkwijze. Het antwoord op (d) is een beetje lang, normaal zou ik niet zulke lange query's vragen, maar ik wilde jullie deze oplossing niet onthouden.