Data & Informatie voorbeeld van Toets 3

Triggers, Transactions, Web Services and XML

De toets bevat XX vragen op YY pagina's die gezamenlijk 100 punten opleveren. Bij het tentamen mogen geen boeken, aantekeningen of elektronische apparaten worden gebruikt.

Opgave 1

Beschouw het volgende databaseschema:

Person (id, name)

Student(id, university)

De bedoeling is dat iedere student ook een persoon is. Daarom willen we dat voortdurend de volgende acties worden uitgevoerd:

1. Wanneer een rij wordt verwijderd uit tabel Person , dan wordt de rij met hetzelfde id ook verwijderd uit tabel Student .
2. Wanneer er gepoogd wordt een rij toe te voegen aan tabel Student , dan wordt getest of er een rij met hetzelfde id bestaat in Person ; is dat niet het geval, dan vindt de toevoeging niet plaats.

Vraag a) Voeg key constraints toe aan het schema waardoor de gewenste acties automatisch door het databasesysteem worden uitgevoerd. Ter herinnering, key c onstraints zien er als volgt uit:

PRIMARY KEY (x1 , x2 , …)

FOREIGN KEY (x1, x2, …) REFERENCES *T*(y1, y2, …) ON *event action* ON *event' action'*

Hierbij staan x1, x2, y1, y2 voor attribuutnamen, *T* voor een tabelnaam, *event* voor een gebeurtenis (DELETE, UPDATE) en action voor een actie (SET NULL, SET DEFAULT, NO ACTION, CASCADE)

Vraag b) Geef nu, voor het oorspronkelijk gegeven schema, een of meer *triggers* waardoor de gewenste acties automatisch door het databasesysteem worden uitgevoerd. Ter herinnering, de syntaxis van een trigger creatie luidt als volgt:

create trigger *trigger-name*
{ *before* | *after* } {insert | delete | update [of *column-name-list*] } on *table-name*
[referencing [old as *var-to-refer-to-old-tuple*]
[new as *var-to-refer-to-new-tuple*]
[old table as *name-to-refer-to-old-table*]
[new table as *name-to-refer-to-new-table*]]
[for each { row | statement }]
[when (*precondition*)]
statement-list

Hierbij staat **{ *x* | ... }** voor een keuze uit *x* , … en **[*x*]** staat voor een keuze uit *x* of niets.

Opgave 2

Gegeven zijn twee transacties waarvan de executievolgorde van de individuele operaties er schematisch als volgt uit zien:

T_1: $read_1(x)$; $read_1(y)$; $write_1(x)$; $write_1(y)$.

T_2: $read_2(x)$; $write_2(x)$.

Hierbij staan x en y voor verschillende objecten uit de database. De read -operaties lezen de waarde van het object, en de write-operaties schrijven een nieuwe waarde in het object.

Vraag a) Geef een niet-serialiseerbare executievolgorde aan voor de operaties van deze transacties:

Vraag b) Geef een executievolgorde voor de operaties van deze transa cties aan zo dat T_2 een *dirty read* doet.

Vraag c) We willen de gelijktijdige executie van transacties T_1 en T_2 zo min mogelijk beperken, maar wel zo dat T_1 geen dirty read s zal doen. Welke transactie(s) moeten we dan op welk isolatieniveau zetten? (Herinner je dat de isolatieniveaus, in opklim mende sterkte van beperkingen, de volgende zijn: READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIAL IZABLE .)

Opgave 3

Vraag a) Wat is het essentiële verschil tussen een web applicatie die alleen HTML pagina's oplevert aan de web clients, en een web service net als de RESTful services behandeld in deze module? Geef een eenvoudig voorbeeld om dit verschil te illustreren.

Vraag b) Wat is de rol van Servlets in de implementatie van RESTful services? Maak een schema waarin deze rol duidelijk is afgebeeld.

Vraag c) Hoe weet een web service geïmplementeerd met Jersey welke methode van welke klasse aangeroepen moet worden voor het afhandelen van een HTTP request bericht?

Opgave 4

Het document `soccer.xml` bevat informatie over het wedstrijdschema en resultaten van een voetbaltoernooi. Het staat een voorbeeld van zo'n document. Neem aan dat in de praktijk dergelijke documenten veel meer data zullen bevatten op dezelfde manier gestructureerd.

```
<schema>
	<game teama="GER" teamb="CRC">
		<group>A</group>
		<result teama="3" teamb="1"/>
		<review>A great opening game in which the Costa Ricans gave
			good performance, but could not beat the Germans.</review>
	</game>
	<game teama="ECU" teamb="GER">
		<group>A</group>
		<notplayed/>
	</game>
</schema>
```

Vraag a) Is dit document well-formed? Leg uit waarom ’t al-dan-niet well-formed is.

Vraag b) Is dit een voorbeeld van zogenoemde data-centric XML of van document-centric XML? Leg uit waarom.

Geef een XQuery voor de onderstaande vragen. Maak ze zo kort mogelijk.
NB: Je query’s moeten niet alleen een correct antwoord geven voor het voorbeelddocument, maar voor alle documenten die op dezelfde manier gestructureerd zijn.

Vraag c) Geef alle informatie over de wedstrijd tussen Duitsland (GER) en Costa Rica (CRC).

Vraag d) Produceer een overzicht voor Duitsland (GER) met drie groepen: alle gewonnen wedstrijden, alle verloren wedstrijden en alle gelijke spelen.

Vraag e) De wedstrijd tussen ECU en GER is gespeeld en de uitslag is 1-1. Update het document met deze uitslag en met een bijbehorende ‘review’ (verzin maar wat onzintekst voor die review).

Vraag f) Vind alle reviews mbv XQuery Full-Text-faciliteiten, die de woorden “Germans” en “beat” bevatten, maar niet in de context van “not beat”.