SE42 Opdracht 1 – Rick Rongen

# Inleiding

Voor SE42 moeten wij in deze opdracht gaan communiceren met een database via JPA. Hierbij communiceer je met de database via objecten in plaats van waardes. Dit doen wij met behulp van EclipseLink.

Ik ga enkele opdrachten bij elkaar beantwoorden omdat deze antwoorden erg op elkaar lijken.

# Entity Manager

In dit gedeelte is de generated value strategy in identity modus.

Als een account word aangemaakt word er wel standaard een accountnummer en een balans, maar geen ID. Dit omdat dit pas word bepaald als het object naar de database gaat. Dus zodra het object gepresist is en de bijbehorende transactie is gecommit dan heeft het object een ID. Als er een rollback word gedaan op de transactie word er dus ook geen ID ingesteld.

Door tijdens een transactie flush aan te roepen worden de huidige wijzigingen naar de database gepushed. Maar er word geen “commit;” gestuurd naar de database. Dit houd dus in dat wijzigingen nog niet permanent zijn, maar gegenereerde waardes wel worden ingesteld. Als er later na een rollback word gedaan heeft dit wel effect op de database maar niet op de lokale objecten.

Als een object gepresist is dan word het op het einde van een commit opnieuw geïnspecteerd op wijzigingen. Alle nieuwe wijzigingen worden dan mee gecommit.

Met refresh kun je van een object nieuwe gegevens uit de database halen. Dit bijvoorbeeld als een object extern bewerkt is of door een andere entity manager.

Merge laat een lokaal bewerkt object dat losgekoppeld was van de entity manager zijn wijzigingen in de database zetten. Hierbij word wel een nieuw object gemaakt dat gepersist is, dus het object wat erin gestopt word, word gekopieerd en de kopie word gepersist.

Met find kan je objecten vinden in de database, hiervoor moet je de klasse van het object wat je wil opvragen mee geven en de primary key. Hierna krijg je een object terug van die klasse die meteen gepersist is.

Met clear kun je alle gepersiste objecten binnen een entity manager detachen. Hierdoor worden gene wijzigingen van die objecten meer bij gehouden.

Door remove aan te roepen kun je een object uit de database halen. Als je dit doet is hij ook helemaal weg uit de database en kan hij niet meer terug gehaald worden. Het lokale object blijft wel bestaan zolang er een referentie naar is.

# Generation Type

Je kan bij een primary key instellen welke Generation Type deze heeft(geld alleen voor numerieke typen). Er zijn 3 verschillende generation types, Identity, Secuence en Table.

## Identity

Bij identity word de id colom van de tabel gebruikt om een volgend nummer te kiezen. Dus als de laatste colom id 10 heeft heeft de volgende 11.

## Sequency

Bij sequence word er een globale sequence gebruikt voor een id te genereren. Als er dus meerdere tabellen zijn kunnen er dus gaten ontstaan omdat de zelfde sequence voor alle tabellen gebruikt word.

Er kan ook zelf een sequence gespecificeerd worden, hierdoor kun je een sequence maken die uniek is voor een tabel of voor enkele tabellen, denk bijvoorbeeld aan subtypering.

## Table

Bij table word er een apparte tabel aangemaakt voor een sequence bij te houden. Hierbij reserveert de entity manager een aantal id’s en increment de colom met een bepaald aantal. Het voordeel hierbij is dat de id direct word toegevoegd zodra je een object gaat presisten, er hoeft dus niks gecommit te worden.

# Statements

Bij transacties word als eerste START TRANSACTION aangeroepen. Hierdoor zal er bij een fout of rollback geen wijzigingen worden opgeslagen in de database. Zodra de flush word aangeroepen word alles wat in de cache zit alvast naar de database gestuurd, ook de START TRANACTION dus. Zodra je commit of rollback aanroept worden de betreffende statements naar de database gestuurd en worden de wijzigingen verwerkt of verworpen.

Bij het aanroepen van persist, word eerst gekeken of een entiteit al bestaat in de database. Er word dus een SELECT uitgevoerd op het id van het object als het niet null is en op alle UNIQUE colommen. Als er geen dubbele regels zijn dan word het object geinsert met een INSERT statement van het object. Als de tabel nog niet bestaat word deze eerst aangemaakt door een gegenereerd CREATE script.

Find voert alleen maar een SELECT uit op de meegeleverde primary key.

Refesh voert ook alleen maar een SELECT uit op de primary key van het object. Hierbij haalt hij de meest recente versie op.

Met merge worden alle wijzigingen van een gedetached object naar de database weg geschreven en word er een nieuwe gepersiste instantie gemaakt met de juiste waarden. Er word een SELECT uitgevoerd op het object in de database om te kijken of deze bestaat en wat de huidige waarden zijn, daarna word een UPDATE uitgevoerd om de gewijzigde waarden naar de database weg te schrijven. Het nieuwe object bevat dus de meest recente versie van dat object met eventueel ongewijzigde data door databaserestricties etc.

Bij clear worden er geen querys uitgevoerd, er word alleen gestopt met het bijhouden van wijzigingen.

Bij remove word er een DELETE uitgevoerd op de primary key van het object.