Praktikumsbericht

Anmerkung: Wir können unser Projekt nicht mit dem Lokalen Docker-Container bzw. der Persistenceunit auf Git pushen. Dort schlägt jedes Mal die Pipeline fehl, deswegen ist auf Git die „fbi-postgresPu“ hinterlegt. Alle Messdaten beziehen sich aber auf die Verwendung des lokalen Dockers.

Massendatengenerierung

Wir generieren für jeden Spieler zufällig einen Namen, mit der Methode „generateRandomPlayer“. Das nicht alle Spieler am selben Tag spielen, generieren wir mit der Methode „getTimeStamp“ unterschiedliche Daten. Weiterhin wird eine zufällige Anzahl von Fragen gespielt, die zwischen 10 und 20 liegt. Dabei können alle Kategorien drankommen.

Wie verwenden Sie flush(), clear(), etc. und warum?

Wir verwenden flush() und clear() jede 10 Spieler, sprich 1000 Spiele.

```
if ((i % 10 == 0) && (i > 0)) {
  em.flush();
  em.clear();
}
```

Der Speicher kann bei den großen Datenmengen schnell aufgebraucht sein. Um dies zu verhindern nutzten wir flush() und clear().

Wie haben Sie eine schnelle Erzeugung der Daten bewirkt?

Durch Batch-Writing werden einzelne Anfragen gesammelt und gruppiert übertragen. Wir verwenden eine Batchsize von 2000. Dies entspricht die Anzahl der Persistaufrufe vor jedem flush und clear.

Wie lange dauert die Massendatengenerierung bei Ihrer Anwendung?

Mit Batch-Writing:

```
The duration is: 16 minutes and 31 seconds
Number of Players:10000
Number of played games: 1000000
Played questions : 15005387
```

Ohne Batch-Writing: Nach 90 min haben wir den Test hier abgebrochen.

Wie benutzen Sie Transactions und warum?

Wir öffnen nur eine Transaction zu Beginn der Datengenerierung und warten bis das Programm mit der Generierung fertig ist, danach schließen wir die Transaction. Vor dem Commit benutzen wir ein clear(), weil wir die Daten nicht weiter im Persistenzkontext benötigen.

Queries

```
@NamedQueries({
    @NamedQuery(name = "Game.count",
        query = "SELECT COUNT(g) FROM Game g"),
    @NamedQuery(name = "PlayedQuestions.count",
        query = "select count (g.givenAnswers) from Game g ")
})
```

Game.count gibt uns die Anzahl der Spiele in der Datenbank zurück.

PlayedQuestions.count gibt uns die Anzahl der gespielten Fragen zurück.

```
@NamedQuery(name = "Player.count",
    query = "select count (p) from Player p")})
```

Player.count gibt uns die Anzahl der Spieler in der Datenbank zurück.