Dokumentation Praktikumsbericht zu Praktikum 4

Testen und dokumentieren Sie, welchen Performancegewinn Sie erzielen, wenn Sie die folgenden Einstellungen für Batch Writing in der persistence.xml einstellen:

```
<property name="eclipselink.jdbc.batch-writing" value="JDBC"/>
<property name="eclipselink.jdbc.batch-writing.size" value="1000"/>
```

Dokumentieren Sie ebenfalls, was Batch Writing datenbankseitig bewirkt und warum Sie hier eine signifikante Verbesserung der Performance mit Batch Writing beobachten:

Dadurch, das nicht ständig wieder eine einzelne Anweisung kommt, sondern die Zugriffe gesammelt kommen, können sie durch Puffern optimiert werden. Batch-Writing hilft dabei, das zeitaufwendige Daten laden auf der Datenbank-Seite zu minimieren. Die Bearbeitung wird also optimiert.

Wie lange dauert die Massen-Datengenerierung bei Ihrer Anwendung?

Die Generierung von Massendaten benötigt ca. 7 - 8 Minuten.

1.) Ganz ohne Optimierung: ca 27min
2.) mit Batch Writing: ca 12min
3.) mit ID-Generator (pooled): ca 10min
4.) mit reWriteBatchedInserts: ca 7min

(abhängig vom Rechner usw.)

Testen Sie das Anlegen Ihrer Massendaten durch geeignete JPQL-Queries

Mit der JPQL-Query "SELECT COUNT(c) FROM Game c" bekommen wir die Anzahl von Game-Entities aus der Datenbank.
Mit der JPQL-Query "SELECT COUNT(c) FROM Player c" erhalten wir die Anzahl von Player-Entities in der Datenbank.
Mit der SQL-Query "SELECT COUNT(*) FROM gamequestion" bekommen wir die Anzahl der Assoziationen der einzelnen Spiele zu ihrer Fragen.

Wie haben Sie eine schnelle Erzeugung der Daten bewirkt?

1.) Batch Writing
2.) ID-Generator
3.) Inserts in einem Batch bündeln zu einem einzigen (reWriteBatchedInserts)

Wie benutzen Sie Transactions und warum?

Wir benutzen Transaktions für die Persistierung bzw Änderung einer Entity.
Um einen Rollback über die gesamten zusammenhängenden Daten machen zu können, falls etwas schief geht.

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

Flush führt die gecachten SQL-Anweisungen direkt aus und sendet die Daten direkt an die Datenbank. Flush wird alle 1000 Daten ausgeführt, um dann den Cache mit clear() wieder zu leeren, ohne eine Transaktion zu unterbrechen.

Clear leert den Cache, so kann verhindert werden, dass der Cache gigantisch anwächst. Der Clear wird direkt nach dem Flush ausgeführt, so werden die gerade abgearbeiteten Daten aus dem Cache entfernt und müllen ihn nicht zu. Wird dieser Befehl vergessen, läuft irgendwann der RAM über.

Darüber hinaus wurden die ID's von 'Game' - und 'Player' -Entities mit SEQUENCE statt IDENTITY erzeugt. Durch SEQUENCE und einem definierten generator können wir uns 1000(Batchgröße) IDs auf einmal von der Datenbank holen und sparen Zeit.