DB Praktikumsbericht

Hammann, Wilpert

June 2020

Inhaltsverzeichnis

1 Einleitung

Innerhalb dieses Praktikum-Berichts dokumentieren wir die Ergebnisse unserer Generierung der Massendaten. Dabei gehen wir auch auf die Verwendung der flush() und clear() Methoden ein, sowie des Perfomance-Gewinn von Batch Writing.

2 Allgemeines Vorgehen

Um die Massendaten zu generieren arbeiten wir mit 2 for Schleifen um jeweils die 10.000 Player und die 100 Games pro Player zu erstellen. Jedes Game bekommt eine zufällige Anzahl an Kategorien im Bereich von 5-7. Jede Kategorie wird mit 2 - 4 Fragen hinzugefügt.

2.1 Ergebnis

Wir haben mit unserer Optimierung eine Zeitspanne von 6:52 min gebraucht um alle Daten in die Datenbank zu persistieren.
Unsere Queries um das erfolgreiche persistieren zu testen liefern 10.000 Player, 1.000.000 Games und ca 11.050.000 GameQuestions zurück. Diese Werte entsprechen auch unseren Erwartungen.
Eine Beobachtung die wir gemacht haben ist, dass die Funktion des Persistieren über die Zeit keine lineare Funktion ist, sondern am Anfang etwas steiler ist als Ende, also am Ende für die selbe Menge an Objekten länger braucht als am Anfang. Zusätzlich sind die Messungen von Schwankungen betroffen, das heißt das die selbe Batch Größe nicht immer zu der selben Zeit führt.

Dennis:

AMD Ryzen 5 2600X@4.05Ghz,

16GB DDR 4 Ram@3200MHz,

HDD@9 MB/s

Batchgröße: 10.000:
10:34 min
GameQuestion: 11.057.768

Batchgröße 100.000:
10:26 min
GameQuestion: 11.051.921

Batchgröße: 100:
11:16 min
GameQuestion: 11.057.896

Batchgröße: 1000:
10:27 min
GameQuestion: 11.056.257

Batchgröße: 500:
10:29 min
GameQuestion: 11.054.151

Batchgröße: 300:
10:24 min
GameQuestion: 11.057.520

Rene:

3900x@4,2 Ghz

16Gb Ram@3200Mhz,

SSD

Batchgröße: 10.000:
7:09 min
GameQuestion: 11052659

Batchgröße 5000:
7:19 min
GameQuestion: 11053605

Batchgröße: 2500:
7:36 min
GameQuestion: 11055663

Batchgröße: 1000:
7:34 min
GameQuestion: 11057030

Batchgröße: 500:
7:32 min
GameQuestion: 11055019

Batchgröße: 300:
7:17 min
GameQuestion: 11055359

2.2 commit

Ein commit() schreibt alle Änderungen die noch nicht mit einem flush() in die Datenbank geschrieben wurde, in die Datenbank.
Wir nutzen einen commit() nachdem ein Spieler und die 100 dazugehörigen Games persistiert wurden. Dies hat den Grund, damit wir öfters ein clear aufrufen können um den persistance context zu leeren.

2.3 Transaction

Wir nutzen eine einzige Transaktion um alle Datensätze zu persistieren, da wir die Anzahl an offenen und geschlossen Transaktionen möglichst gering halten wollen.

2.4 flush() und clear()

flush() schreibt die vorher mit persist() persistierten Daten in die Datenbank, dies ist jedoch noch nicht für alle sichtbar, clear() löscht den persistance context, das alle vorher verwalteten Entities abhängt. Änderungen ohne ein flush() werden nicht persistiert.
Wir nutzen jeweils ein clear() nach dem commit() von den 100 Games und einem Player, um den persistance context regelmäßig zu löschen.

2.5 Batch Writing

Beim Batch Writing werden mehrere Anfragen auf die Datenbank erst gesammelt um dann gemeinsam an die Datenbank übertragen zu werden.
Wir nutzen eine Batch Writing größe von x, um die Anzahl an Übertragungen an die Datenbank vernünftig klein zu halten aber gleichzeitig die Größe der Sammlung in einem gewissen Rahmen zu halten um diese nicht zu groß zu machen.
Bei uns ist jedoch die Problematik aufgetaucht, dass wir anhand der Werte nicht genau bestimmen konnten welche Batch-Größe besser oder schlechter sei.