Datenbanken 2 - Praktikum 4: Bericht

• Wie lange dauert die Massendatengenerierung bei Ihrer Anwendung?

Versuch 1:	**Dauer:** ca. 21min	**Beschreibung:** 1. 1. Transaktion öffnen (einmal) 2. 10000 Personen anlegen mit jeweils 100 Matches 3. Transaktion schließen (einmal) Zusätzlich: in Player: cascade = {CascadeType.PERSIST} vor matchList	**Beobachtung:** Commit und insert erst am Ende der Laufzeit **Folgen:** Hoher Speicherplatzbedarf
Versuch 2: Mit Batch	**Dauer:** 7,3min	**Beschreibung:** **Zu Versuch 1 – Erweiterungen:** Änderungen in der persistence.xml: <property name="eclipselink.jdbc.batch-writing" value="JDBC"/> <property name="eclipselink.jdbc.batch-writing.size" value="1000"/>	**Beobachtung:** Sehr viele Commits am Anfang und Inserts nur am Ende der Laufzeit **Folgen:** Hoher Speicherplatzbedarf geringer Laufzeit
Versuch 3: Mit Batch, flush(), clean()	**Dauer:** 6,5min	**Beschreibung:** **Zu Versuch 1 – Erweiterungen:** Änderungen in der persistence.xml: <property name="eclipselink.jdbc.batch-writing" value="JDBC"/> <property name="eclipselink.jdbc.batch-writing.size" value="1000"/> Mehrere kleinere Transaktionen (statt eine große) Jede Transaktion mit flush() und clear() (siehe bei Zusätzlich)	**Beobachtung:** Synchrone commits und inserts in gewissen zeitlichen Abständen **Folgen:** geringer Speicherplatzbedarf geringer Laufzeit

• Wie haben Sie eine schnelle Erzeugung der Daten bewirkt?
(Siehe oben)

• Wie benutzen Sie Transactions und warum?
Mehrere Transactions mit flush() und clear() um Speicherplatz zu sparen, da der Persistenzkontext sonst zu groß werden würde und sich dies somit negativ auf die Laufzeit auswirken würde.

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

flush() und clear() innerhalb einer Transaktion zur regelmäßigen Bereinigung des Persistenzkontextes:
„Flush“ sorgt für eine Synchronisation mit der verbundenen Datenbank.
„Clear“ leert anschließend den Persistenzkontext und überführt alle aktuell gemanagten Entities in den Zustand „detached“.
Vorteil: Die Entities müssen somit nicht bei jeder Änderung mit der Datenbank abgeglichen und bearbeitet werden.

Zusätzlich:

Wann öffnen und schließen Sie Transaktionen?
Reihenfolge pro Transaktion:

1. begin()
2. „Objekt erzeugen und initialisieren“
3. flush()
4. clear()
5. commit()

(wird mehrfach durchlaufen)

Performanceoptimierung:
Durch die Nutzung von Batch-Writing kann eine deutliche Leistungssteigerung bei der Erstellung und Persistierung der Massendaten erreicht werden.

Batch Writing basiert auf einer **Bündelung von SQL Anfragen**, sodass **alle Anfragen gemeinsam übermittelt** werden können und homogene **Anfragen nicht doppelt** in der Datenbank verarbeitet werden müssen. Hierbei macht man sich beispielsweise parametrisierte SQL statements zu Nutze, die einmal generiert und mehrfach mit unterschiedlichen Parametern aufgerufen werden, anstatt ein separates SQL statement zu generieren für jeden Datenbankzugriff.