2013/14

KLUNE Alexander

TGM

Aufgabe 06

Replikation

Inhalt

[Aufgabenstellung 2](#_Toc380078429)

[Aufgabenaufteilung 3](#_Toc380078430)

[Zeiteinteilung 3](#_Toc380078431)

[Geschätzte Zeit 3](#_Toc380078432)

[Tatsächliche Zeit 3](#_Toc380078433)

[Designüberlegung 4](#_Toc380078434)

[Files 4](#_Toc380078435)

[Problem 5](#_Toc380078436)

[Problemstellungen 5](#_Toc380078437)

[Problemstellungen 7](#_Toc380078438)

[Resultate 8](#_Toc380078439)

[Niederlagen 8](#_Toc380078440)

[Quellen 9](#_Toc380078441)

# Aufgabenstellung

Eine Handelsgesellschaft, die mehrere Filialen hat, betreibt eine Online-Plattform für den Verkauf der Produkte. Der Webshop wird mit Hilfe einer Datenbank betrieben. Bei dem Verkauf der Produkte werden Rechnungen in Form von PDF-Dokumenten erzeugt.  
  
Aufgabenstellung:  
Die Daten (Datenbank, Rechnungen) sollen stets auf die Filialrechner repliziert werden, damit die Sachbearbeiter vor Ort diese einsehen und bearbeiten können.

* Entwickle ein vereinfachtes Datenbankmodell für den Webshop
* Wähle ein Konsistenzprotokoll Deiner Wahl (siehe Theorie bzw. Tanenbaum)
* Implementiere einen Replikationsmanager in Java (JDBC, Sockets, o.ä. ...) für Datenbank und Rechnungen
* alle Transaktionen im Zuge der Replikation sollen protokolliert werden (zum Beispiel mit Log4J)

Beispiel fuer Log-Eintrag:  
Replikation Rechnungen München -> Berlin OKAY  
Replikation DB MÜnchen -> Berlin FEHLGESCHLAGEN  
  
Problemstellungen:

* Wie oft wird repliziert?
* Wie erfolgt der Aufruf des Replikationsmanager bzw. läuft der Replikationsmanager stets im HIntergrund?
* Was passiert im Fehlerfall?
* Welche Probleme koennen auftreten?  
  - Dateien mit gleichen Namen  
  - Dateien mit gleichen Namen und unterschiedlicher Größe  
  - Datensatz mit gleichem Schlüssel

Meilensteine (16Pkt): Gruppengröße 2 Personen

* Erstelle ein Replikationskonzept für diese Handelsgesellschaft (4 Punkte)
* Implementiere dieses Konzept für zwei Rechner (6 Punkte)  
  mind. 10 Datensätze pro Tabelle, mind. 10 Rechnungen
* Implementierung Logging (2 Punkte)
* Dokumentiere drei Fehler-/Problemfälle und entsprechende Lösungvorschläge (4 Punkte)

# Aufgabenaufteilung

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| Datenbankmodell | X |  |
| Implementierung Konsistenzmodell |  | X |
| Replikationsmanager |  | X |
| Loging | X |  |
| Problemstellungen | X | X |
| Testing | X | X |
| Gesamt | 4 | 4 |

# Zeiteinteilung

## Geschätzte Zeit

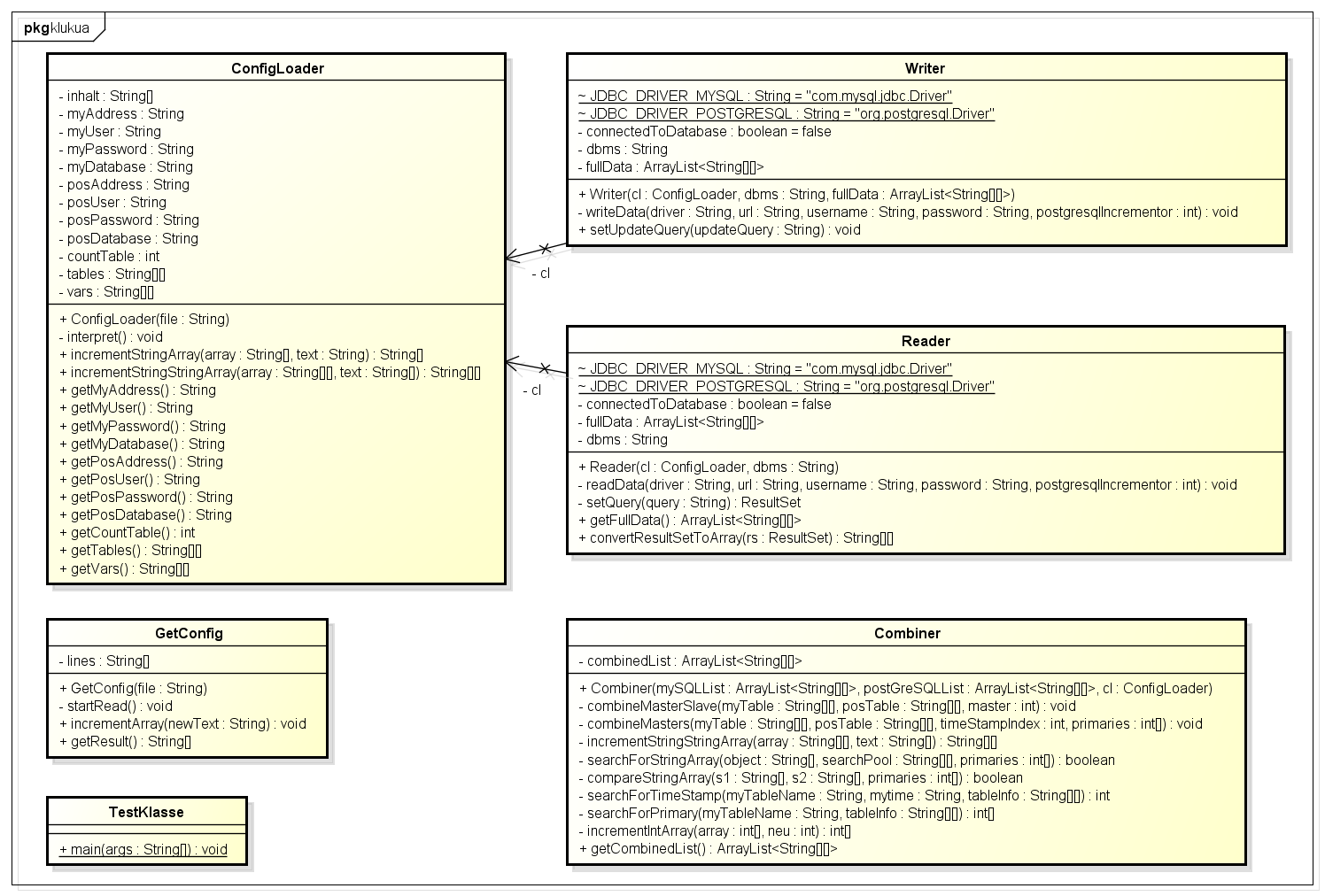
|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| Datenbankmodell | 1 h |  |
| Implementierung Konsistenzmodell |  | 2 h 30 min |
| Replikationsmanager |  | 4 h |
| Logging | 3 h | 1 h |
| Problemstellungen | 2 h | 2 h |
| Testing | 2 h 30 min | 1 h |
| Gesamt | 8 h 30 min | 10 h 30 min |

## Tatsächliche Zeit

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| Datenbankmodell | 30 min |  |
| Implementierung Konsistenzmodell |  |  |
| Replikationsmanager |  |  |
| Logging | 1 h |  |
| Problemstellungen |  |  |
| Testing |  |  |
| Gesamt |  |  |

# Designüberlegung

## Klassendiagramm DBMS Replikation



Konsistenzmodell:

Urbildbasiertes Protokoll: Entfernte Schreibvorgänge

Datenbank PostgreSQL?

Replikationsverwaltung mit slony?

<http://wiki.postgresql.org/wiki/Slony>

<http://slony.info/>

PostGres 9.0 beinhaltet ebenfalls replication

## Files

Die Übertragung und Kommunikation über RMI.

Es wird dabei an das Konzept von Dropbox festgehalten. Das heißt es gibt nur einen Ordner der stets synchron gehalten wird. Dieser muss beim starten des Programmes angegeben werden. (Hier muss überprüft werden ob der Ordner überhaupt existiert.) Danach werden die Dateien Synchronisiert. Wenn gleiche Dateien enthalten sind in beiden Ordnern beim starten diese aber eine Unterschiedliche Größe haben werden sie unbenannt in „Dateinname In Konflikt stehende Datei.xx“

Bei der Implemenatation handelt es sich um eine Master-Slave Architektur. Es wurde das Urbildbasierte Protokoll für lokales Schreiben implementiert. Das heißt es ist auf beiden Rechnern ein schreibender Zugriff möglich und dabei wird die Datei auf allen anderen Nodes gesperrt. Danach werden die Dateien synchronisiert und die Sperren aufgebhoben.

Verwendet wird für die File bearbeitung die Java Lybarie: java.nio.file diese ist ab Version 7 verfügbar und ist eine Verbesserung der Lybarie java.io.File. [1]

Es wird ein Watchdog implementiert dieser schaut ob Änderungen stattfinden. Dieser Watchdog implementiert Runnable und wird als Thread gestartet. Er besitzt folgenden Algorithmus in seiner run() Methode:

Path path = Paths.get(URI.create("file:/C:/Temp/"));

WatchService watchService = path.getFileSystem().newWatchService();

WatchKey watchKey = path.register(watchService, StandardWatchEventKinds.ENTRY\_CREATE);

while(true) {

  for(WatchEvent<?> event : watchKey.pollEvents()) {

    Path newPath = (Path)event.context();

    System.out.println("New file: " + newPath);

  }

}

Diese Nutzung der Notification Funktion von java.nio.File benachrichtigt den Thread immer dann wenn eine Datei erstellt oder gelöscht wird. Wenn dies der fall ist wird dementsprechend darauf reagiert und die Synchronisationsmethode aufgerufen.

Der Synchronisationsmethode wird gleichzeitig auch die neue Datei übergeben beim Aufruf.

### Problem

Das Problem tritt genauso wie bei Dropbox dann auf wenn beide Teilnehmer eine Datei gleichzeitig ändern da die Änderungen erst beim speichern in Kraft treten und somit möglicherweise zwei Nodes gleichzeitig ungespeicherte Änderungen vornehmen könnten.

In diesem Fall ist meine Lösung, dass der der zuerst speichert gewinnt.

### Problemstellungen

Wie oft wird repliziert?

Es wird immer dann repliziert wenn eine Änderung stattgefuden hat diese wird durch die oben beschriebene Notification erkannt.

Wie erfolgt der Aufruf des Replikationsmanager bzw. läuft der Replikationsmanager stets im HIntergrund?

Der Replikationsmanager läuft immer im Hintergrund andernfalls gibt es keine Synchronisation. Synchronisiert wird allerdings erst bei Änderungen.

Was passiert im Fehlerfall?

Im Fehlerfall beispielsweise bei gleichzeitiger Veränderung gewinnt derjenige der zuerst speichert.

Bei wird das Urbild des Datenelements dieses wird anschließend auf alle anderen Nodes wieder übertragen.

Welche Probleme koennen auftreten?

Dateien mit gleichen Namen

Dieses Problem muss hier ausgeschlossen werden da der Name als Identifikation gilt. Dieser muss unique sein.

Wenn beim starten Dateien mit gleichem Namen auftreten und diese die gleiche Größe haben ist das nicht weiter problematisch. Andernfalls siehe unten.

Dateien mit gleichen Namen und unterschiedlicher Größe

Wenn Datensätze mit gleichem Namen auftauchen aber eine Unterschiedliche Größe besitzen (geht nur beim startup ansonsten wird eine Fehlermeldung ausgegeben) dann wird diese unbenannt in „Dateinname In Konflikt stehende Datei.xx“ und der Pfad dieser Datei in eine Liste aufgenommen von Dateien die nicht synchronisiert werden.

Datensatz mit gleichem Schlüssel

Spielt bei Files keine Rolle.

## Datenbank

Für die Replikation einer Datenbank wurde der Code von Aufgabe „heterogene Datenbank Synchronisation“ hergenommen und adaptiert, sodass dieser auch mit homogenen DBMS funktioniert. Weiters wurde der Logger nicht mit Log4j implementiert, sondern lediglich mit Files und Streams, da es für diese Übung vollkommen ausreichend ist. Es werden alle lese- und Schreibvorgänge während das Programm läuft in mit Timestamp und DBMS gekennzeichnete Files gelogged. Zu erwähnen ist, dass die Vorlage aus dem letzten Jahr nicht abgenommen werden konnte, da mit einer zweiten Gruppe aus der Klasse (Schulmeister, Maran) zusammengearbeitet wurde ohne dies kenntlich zu machen. Deswegen teilen wir hiermit mit das die Source nicht nur von Klune und Huang stammen sondern auch von Maran und Schulmeister. Die genaue Teilung, wer was geschrieben hat, kann nicht mehr durchgeführt werden, da dies zu lange her ist um bestimmten Code einer Person zuzuweisen. Die Beschreibung des Grundsources ist im Protokoll dieser Übung und kann entweder in Moodle oder im Abgabeordner gefunden werden.

Die Synchronisation ist als Master – Master aufgebaut. Es werden zuerst alle Daten aus der DB gelesen, miteinander verglichen, ein gemeinsamer Daten 2D Array erstellt und dieser wird am Ende auf beide DBMS übertragen. Dadurch hat jedes DBMS direkt nach der Synchronisation die gleichen Daten. Das Unterscheiden, welcher Datensatz bei Master – Master geschieht durch Timestamps, welche bei einem Update des Datensatzes in diesem Befehl manuell erhöht werden muss.

## Problemstellung

### Wie oft wird Repliziert?

Wie oft repliziert wird, hängt vom User ab, denn die Anzahl hängt von den Startargumenten ab. Diese kann in ganzen Schritten angegeben werden. Bei mehreren Replizierungen wird alle 30 Sekunden eine vollständige Replizierung durchgeführt.

### Wie erfolgt der Aufruf des Replikationsmanager bzw. läuft der Replikationsmanager stets im Hintergrund?

### Was passiert im Fehlerfall?

Im Fehlerfall können im Schlimmsten Fall die Daten verloren gehen. Doch dies ist meist eigenverschulden durch falschen Input und somit bei gründlicher Bedienung auszuschließen.

### Welche Probleme können auftreten?

-Datensatz mit gleichem Schlüssel

Derselbe Schlüssel ist kein Problem, da jeder Datensatz über einen Timestamp verfügt, welcher die Aktualität festlegt. Damit kann eruiert werden welcher Datensatz aus beiden DBMS der aktuellere ist und kann auf das andere System übernommen werden. Sollten die Timestamps und die Primarykeys gleich sein, wird der Datensatz des ersten DBMS (config) übernommen.

Durch das Einführen einer Bedienungspolicy kann ein solches Problem behoben werden.

# Problemstellungen

# Resultate

# Niederlagen

# Quellen

[1] „java.nio.file: Zeitgemäßes Arbeiten mit Dateien“ <http://jaxenter.de/artikel/javaniofile-Zeitgemaesses-Arbeiten-mit-Dateien-166848>

[2] „Package java.nio.file“ <http://docs.oracle.com/javase/7/docs/api/java/nio/file/package-summary.html>

[3] <http://stackoverflow.com/questions/14977663/java-sockets-send-a-file-from-the-client-to-the-server>