2012
/13

Aufgabe 15

Synchronisation von Heterogenen Datenbanken

Alexander Klune, Kuanlun Huang
4AHITS
2012/13

Inhalt

Aufgabenstellung

Dokumentieren Sie Ihren Versuch zwei heterogene Datenbanksysteme (MySQL, Postgresql) zu synchronisieren. Verwenden Sie dabei unterschiedliche Schemata (verschiedene Tabellenstruktur) und zeigen Sie auf, welche Schwierigkeiten bei den unterschiedlichen Heterogenitätsgraden auftreten können (wie im Unterricht besprochen) [2Pkt].

Implementieren Sie eigenständig eine geeignete Middleware [8Pkt]. Testen Sie Ihr gewähltes System mit mehr als einer Tabelle [4Pkt] (Synchronisation bei Einfügen, Ändern und Löschen von Einträgen) und dokumentieren Sie die Funktionsweise, sowie auch die Problematiken bzw. nicht abgedeckte Fälle [2Pkt].

Das PDF soll ausführlich beschreiben, welche Annahmen getroffen wurden und wie diese begründet werden. Der Source-Code muss den allgemeinen Richtlinien entsprechen und ebenfalls abgegeben werden.

Gruppengröße: 2
Gesamtpunkte: 16 [Aufteilung in eckigen Klammern ersichtlich]

Punkte

Dokumentation der Synchronisation [2Pkt]

Implementierung der Middleware [8Pkt]
Zeittrigger bzw. Listener für Synchronisation
Konfiguration bez. Mapping der Tabellen und Attribute
Konfliktlösung bei Zeitüberschneidung bzw. Datenproblemen (Log)
LostUpdate-Problem
Master-Slave, Master-Master Replikation

Test mit mehr als einer Tabelle [4Pkt]
Uni- und Bidirektionale Änderungen mehrerer Tabellen
Einfügen, Ändern und Löschen

Dokumenation der Funktionsweise, Problematiken und Problemfälle [2Pkt]
Designdokumentation (Code + DB)
Synchronisationsverhalten
unbehandelte Problemfälle

Protokoll und Sourcecodedokumentation [0..-6Pkt]

Arbeitsaufteilung

Arbeitsauftrag	Klune	Huang
Konzept	X	X
Erstellen von2 Heterogenen DB- Schemata	X	X
Einlesen und Interpretieren des Config Files		X
Aus bestehenden DBs auslesen	X	
Tabellendaten vergleichen und mergen	X	X
DB mit neuen Daten befüllen	X	
Tests	X	X
Protokoll schreiben	X	X
Summe	5	4

Arbeitsaufwand

Aufwandsabschätzung

Arbeitsauftrag	Klune	Huang
Konzept	1 h	1 h
Erstellen von 2 Heterogenen DB- Schemata	0,5 h	0,5 h
Einlesen und Interpretieren des Config Files		2 h
Aus bestehenden DBs auslesen	2 h	
Tabellendaten Synchronisieren MM/SM	5 h	5 h
DB mit neuen Daten befüllen	2 h	
Tests	2 h	2 h
Protokoll schreiben	2 h	2 h
Summe	14,5 h	12,5 h

Tatsächlicher Arbeitsaufwand

Arbeitsauftrag	Klune	Huang
Konzept	1 h	1 h
Erstellen von 2 Heterogenen DB- Schemata	0,5 h	0,5 h
Einlesen und Interpretieren des Config Files		2,5 h
Aus bestehenden DBs auslesen	3 h	
Tabellendaten Synchronisieren MM/SM	3,5 h	3 h
DB mit neuen Daten befüllen	1,5 h	
Tests	1 h	1 h
Protokoll schreiben	2 h	2 h
Summe	12,5 h	10 h
Summe Gesamt	22,5 h	

Programmstart

Um das Programm auszuführen muss ein funktionierendes config File vorliegen sowie 2 erreichbare Datenbankserver.

Der Programmstart sieht so aus:

Java –jar „Pfad zur -jar“ „Pfad zum config File“

Nachdem das Programm gestartet wurde arbeitet es alle Einstellungen die im config File beschrieben sind ab.

Das config File

Das config File ist der wichtigste Bestandteil des Programms. Es enthält Anmeldeinformationen über MySQL sowie PostgreSQL, Informationen über die verschiedenen Tabellen, Variablen und Datentypen in der gewünschten Datenbank. Weiters wird in der config Datei festgelegt wie Synchronisiert wird, ob MasterMaster oder MasterSlave. Dies kann auf Tabellenebene angegeben werden. Ohne dem config File kann keine Synchronisation mit dem Programm stattfinden. Um eine config Datei selber zu schreiben kann die Darstellung und aufschlüsselung im nachfolgenden Unterkapitel „Wie sieht eine config Datei aus?“ hilfreich sein. In diesem Beispiel sind alle Möglichkeiten vertreten.

Wie sieht eine config Datei aus?

Das im Test verwendete config File hat folgenden Inhalt:

```
[mysql]
localhost root Test SyncTestMy
[PostgreSQL]
localhost postgres Test synctestpost
MyTabelle1 PostTabelle1 MyTabelle1
MyTestnummer PostTestnummer number+
MyName PostName string
- PostZFeins string
- PostZFzwei string
- PostZFdrei number
;
MyTabelle2 PostTabelle2 (MyTime,PostTime)

MyName PostName string+
MyAlter PostAlter number
MyAdr PostAdr string
MyTime PostTime number
;
MyTabelle3 PostTabelle3 MyTabelle3
MyNummer PostNummer number+
MyBank PostBank string
MyHomepage PostHomepage string
MyAdresse PostAdresse string
;
MyTabelle4 PostTabelle4 PostTabelle4
MyName PostName string+
MyStart PostStart string
MyEnd PostEnd string
MyGehalt PostGehalt number
;
```

```
MySQL Header
        URL, User, Passwort, Datenbankname
PostgreSQL Header
        URL, User, Passwort, Datenbankname
MySQLTabelle PostgreTabelle        MasterTabelle
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
    KeineSpalteinMySQL PostgreSpalte Datentyp
    KeineSpalteinMySQL PostgreSpalte Datentyp
    KeineSpalteinMySQL PostgreSpalte Datentyp
Beenden der Tabelle
MySQLTab. PostgreTab. (MySQLTimestamp,
PostgreTimestamp) *
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
    MySQLTimestamp PostgreTimestamp Datentyp
Beenden der Tabelle
MySQLTab. PostgreTab. Mastertabelle
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
Beenden der Tabelle
MySQLTab. PostgreTab. Mastertabelle
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
    MySQLSpalte PostgreSpalte Datentyp
Beenden der Tabelle
```

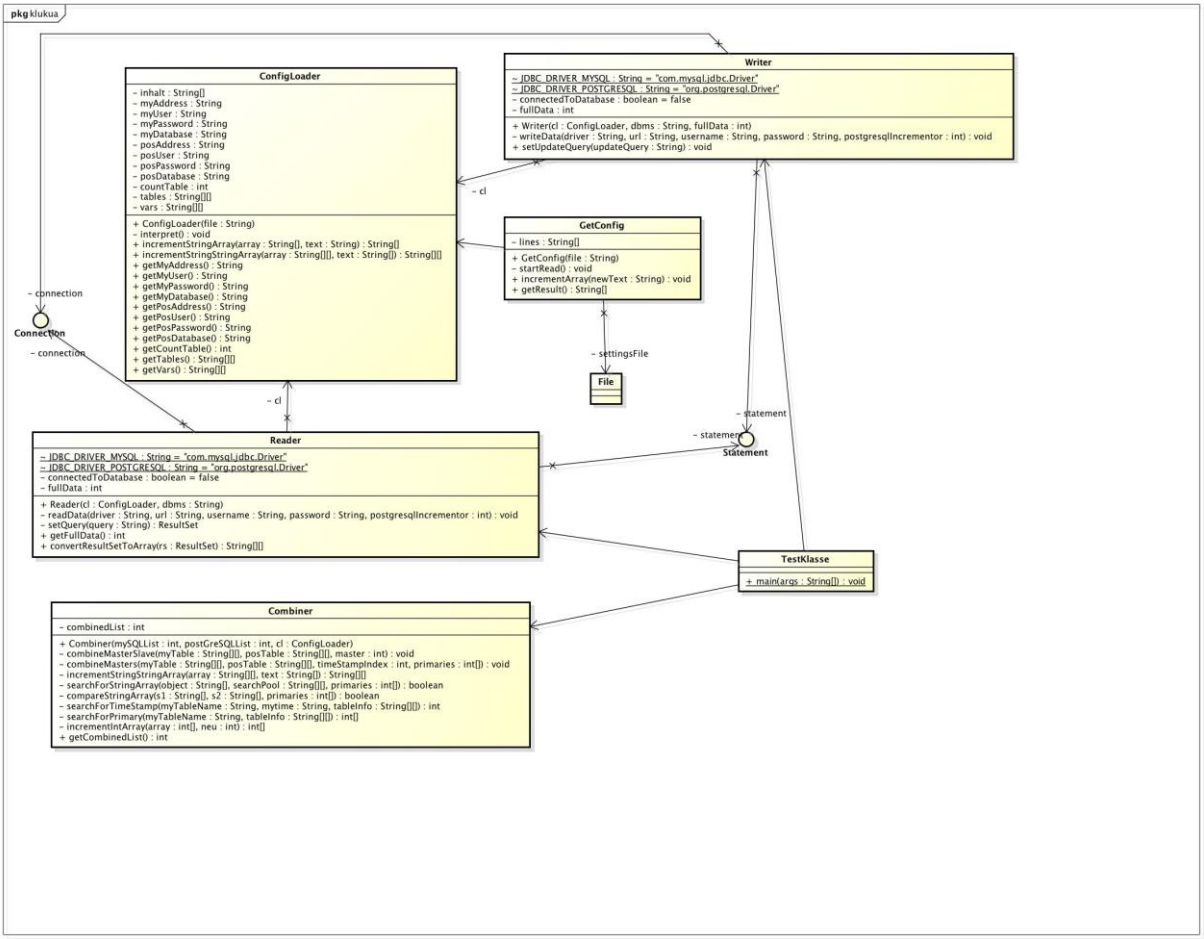
Das ‚+‘ am ende des Datentypes identifiziert den Primarykey

*Master-Master Synchronisation;Wenn keine Klammern vorhanden sind, ist es eine Master-Slave Synchronisation

Design

Dieses Programm sollte von der Konsole mit Argumenten aus gestartet werden. Es soll in einem Programmablauf die Daten der Datenbanken einmal synchronisieren. Für eine erneute synronisation muss das Programm lediglich nur ein weiteres mal aufgerufen werden. Bei der Erstellung dachten wir an eine Gute Aufteilung der einzelnen Funktionen, da wir annahmen eine Datenbanksynchronisation wird viel Code benötigen. Wir haben unser Programm in vier Hauptfunktionen unterteilt: Config, Reader, Combiner, Writer. Config beinhaltet das auslesen der config Datei und das interpretieren der im File stehenden Informationen. In Reader werden alle Tabelleneinträge der im config- File angegebenen Tabellen lokal gespeichert. Der Combiner kümmert sich um die Zusammenführung der einzelnen Daten mit Master/Master und Master/Slave und speichert die synchronisierten Daten lokal ab. Zu Guter letzt wird im Writer vor dem Inserten der Daten alle Einträge aus den Tabellen gelöscht um einen komplett neuen Datensatz hinzufügen zu können.

Klassendiagramm



Test

Es wurde ein Test mit Heterogenen Datenbanken und mehreren Tabellen durchgeführt. Des Weiteren sind in dem Testfall die verschiedenen Synchronisationsmethoden getestet worden, sowie ein Eintrag mit null Wert.

```
C:\Users\Klune>java -jar C:\Users\Klune\Documents\NetBeansProjects\HeterogeneSynchronisation\dist\HeterogeneSy
nchronisation.jar "C:\Users\Klune\Documents\4. Klasse\VSDB\Aufgaben\Aufgabe 15\config"
2 Row/s are affected!
INSERT INTO MyTabelle1 (MyTestnummer,MyName) VALUES(1,'Alexander Klune')
1 Row/s are affected!
INSERT INTO MyTabelle1 (MyTestnummer,MyName) VALUES(2,'Kuanlun Huang')
1 Row/s are affected!
2 Row/s are affected!
INSERT INTO MyTabelle2 (MyName,MyAlter,MyAdr,MyTime) VALUES('Alexander Klune',19,'Zuhause 2',2)
1 Row/s are affected!
INSERT INTO MyTabelle2 (MyName,MyAlter,MyAdr,MyTime) VALUES('Kuanlun Huang',19,'zuhause 1',1)
1 Row/s are affected!
2 Row/s are affected!
INSERT INTO MyTabelle3 (MyNummer,MyBank,MyHomepage,MyAdresse) VALUES(11,'BACA','baca.at','Bankstrasse 1')
1 Row/s are affected!
INSERT INTO MyTabelle3 (MyNummer,MyBank,MyHomepage,MyAdresse) VALUES(12,'Erste Bank','erstebank.at','Bankstras
se 2')
1 Row/s are affected!
1 Row/s are affected!
INSERT INTO MyTabelle4 (MyName,MyStart,MyEnd,MyGehalt) VALUES('Testfall mit NULL','2010-12-12','2022-12-22',nu
ll)
1 Row/s are affected!
2 Row/s are affected!
INSERT INTO PostTabelle1 (PostTestnummer,PostName) VALUES(1,'Alexander Klune')
1 Row/s are affected!
INSERT INTO PostTabelle1 (PostTestnummer,PostName) VALUES(2,'Kuanlun Huang')
1 Row/s are affected!
2 Row/s are affected!
INSERT INTO PostTabelle2 (PostName,PostAlter,PostAdr,PostTime) VALUES('Alexander Klune',19,'Zuhause 2',2)
INSERT INTO PostTabelle2 (PostName,PostAlter,PostAdr,PostTime) VALUES('Kuanlun Huang',19,'zuhause 1',1)
1 Row/s are affected!
2 Row/s are affected!
INSERT INTO PostTabelle3 (PostNummer,PostBank,PostHomepage,PostAdresse) VALUES(11,'BACA','baca.at','Bankstrass
e 1')
1 Row/s are affected!
INSERT INTO PostTabelle3 (PostNummer,PostBank,PostHomepage,PostAdresse) VALUES(12,'Erste Bank','erstebank.at',
'Bankstrasse 2')
1 Row/s are affected!
1 Row/s are affected!
INSERT INTO PostTabelle4 (PostName,PostStart,PostEnd,PostGehalt) VALUES('Testfall mit NULL','2010-12-12','2022
-12-22',null)
1 Row/s are affected!
```

Resultate/Niederlagen

Die Datentypen der Tabelleneinträge haben wir am Anfang gesamt in eine Liste gespeichert, was dazu geführt hat, dass bei jedem Tabellendurchlauf die Datentypen aus der ersten Tabelle verwendet wurden.

Lösung: Wie haben die Datentypen in einer Schleife in einen Array gespeichert. Beim raufladen in die Datenbank konnten wir die Datentypen mit einer Schleife korrekt auslesen und benutzen.

Quellen

http://elearning.tgm.ac.at/mod/assign/view.php?id=12404
http://jdbc.postgresql.org/
http://dev.mysql.com/downloads/connector/j/