VSDB

Replikation

Protokoll

Alexander Rieppel Dominik Backhausen

14. Februar 2014

5AHITT

Inhaltsverzeichnis

1 Aufgabenstellung **3**

2 Designüberlegung **5**
2.1 Allgemein 5
2.2 Konsistenzmodell 5

3 Arbeitsaufteilung **6**

4 Arbeitsdurchführung **7**

5 Testbericht **8**
5.1 Fragen 8
5.2 Fehlerszenarien 8

1 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 können auftreten? (Dateien mit gleichen Namen, Dateien mit gleichen Namen und unterschiedlicher Größe, Datensatz mit gleichem Schlüssel)

Meilensteine (16Pkt):

- 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)

2 Designüberlegung

2.1 Allgemein

Es soll einen oder mehrere Clients geben die sich zuerst mit dem Server verbinden müssen um die aktuellen Daten zu erhalten. Als nächstes können die Clients sich mit dem Server synchronisieren, wobei das Verfahren im Kapitel Konsistenzmodell näher erläutert ist. Wenn ein Client ein File in seinem Dateisystem aktualisiert, kann er dieses File synchronisieren. Es wird danach ein Abgleich der Änderungsdaten gemacht und die entsprechenden Files aktualisiert.

Bei der Datenbank ist die Umsetzung so geplant, dass sämtliche Befehle die an der Datenbank getätigt werden zunächst mitgeloggt werden und dann alle notwendigen Befehle an die anderen Hosts gesendet werden.

2.2 Konsistenzmodell

Als Konsistenzmodell wird das Monotone Schreiben gewählt, da es nach unserer Ansicht für diese Aufgabenstellung sinnvoll ist, erst alle Schreibvorgänge des Servers beim Client auszuführen bevor dieser Client wieder etwas zum Server hochladen kann.

3 Arbeitsaufteilung

Name	Arbeitssegment	Time Estimated	Time Spent
Alexander Rieppel	Datenbankverbindung	1h	3h
Alexander Rieppel	Replikationsmanager	4h	2h
Alexander Rieppel	Trigger für Datenbank	1h	1h
Dominik Backhausen	Verbindung der Clients	1h	1h
Dominik Backhausen	Replikationsmanager	4h	5h
Dominik Backhausen	Datenbankverbindung	1h	1h
Gesamt		12h	13h

4 Arbeitsdurchführung

Die tatsächliche Implementierung der Synchronisation von normalen Files ist nach oben beschriebener Designüberlegung erfolgt. Der Austausch der Files funktioniert problemlos.

Die Datenbank wurde ebenfalls so, wie in der Designüberlegung angeführt, umgesetzt. Um dies zu realisieren gibt es eine eigene Tabelle welche für das mitloggen aller Aktivitäten zuständig ist. Dabei ist zu beachten, dass alle INSERTS, UPDATES, und DELETES von auf der Datenbank vorinstallierten TRIGGERN, in die Logtabelle geschrieben werden. Aller TRIGGER werden vom Programm für die entsprechenden Tabellen generiert und beim Programmstart in die Datenbank automatisch eingetragen. Anschließend werden nach dem loggen die an der Datenbank durchgeführten Aktivitäten ausgelesen und alle benötigten Befehle an den Server und damit an die anderen Hosts gesendet.

5 Testbericht

Zum Start des Programms gibt es folgende Möglichkeiten:

- ds (Starten des Servers mit Default Einstellungen)
- dc (Starten des Clients mit Default Einstellungen)
- s <PORT> <FILEDIR> <LOGFILE> <DB-IP> <DB-NAME> <DB-USER> <DB-PASS> (Starten des Servers mit angegebenen Argumenten)
- c <SERVER-IP> <PORT> <FILEDIR> <LOGFILE> <DB-IP> <DB-NAME> <DB-USER> <DB-PASS> (Starten des Clients mit angegebenen Argumenten)

Die Argumente für die Datenbank sind die gleichen, da es sich um ein einziges Programm handelt, dass beides managed.

5.1 Fragen

- Wie oft wird repliziert?

  Es existieren beliebig viele Clients und ein Server, demnach wird sooft repliziert wie Nodes vorhanden sind.

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

  Der Aufruf erfolgt per Commandline und Argumenten, wobei der Rep-Manager stets im Hintergrund läuft.

- Was passiert im Fehlerfall?

  Im Fehlerfall wird das File bzw. der Datenbankeintrag schlicht nicht gesendet. Da das Programm allerdings über Problemlösungsleitfäden verfügt, ist die Wahrscheinlichkeit, dass ein Fehler auftritt, eher gering.

5.2 Fehlerszenarien

- Dateien mit gleichen Namen

  Wird insofern gehandhabt, dass die Datei simpel als gleiche Datei behandelt und überschrieben wird, wenn eine neuere Datei bei Client oder Server ankommt.

- Dateien mit gleichen Namen und unterschiedlicher Größe

  Siehe Punkt 1

- Datensatz mit gleichem Schlüssel

  Bei Insert ist die Synchronisation fehlgeschlagen, da es, wenn er die aktuelle Datenbank besitzt, erst gar nicht in die Datenbank eingetragen wird. Sollte er eine alte Version der Datenbank haben (was hoffentlich nicht der Fall sein sollte), passiert das gleiche, mit der Ausnahme, dass der Synchprozess erst beim Austausch der Daten mit dem Server fehlschlägt.