VSDB

Datenbanksynchronisation verteilter Datenbanken

Umsetzung mit Couchbase Lite

Alexander Rieppel

27. Mai 2014

5AHITT

Inhaltsverzeichnis

1 Einführung

1.1 Umsetzung im Diplomprojekt

Ziel des Diplomprojektes war es eine Applikation für ein Tablet zu entwickeln, dass die Mitarbeiter der Lebensmittelversuchsanstalt bei der Probennahme im Supermarkt unterstützen soll. Insgesamt umfasst das Diplomprojekt eine Android Applikation und eine Server-Applikation. Bei der Probennahme werden Lebensmittel eingekauft, die am Tablet entsprechend in eine Datenbank eingetragen und später in der LVA für die weiter Bearbeitung bereitgestellt werden. Die Daten werden anschließend in die interne Datenbank in der LVA synchronisiert und für die Verarbeitung der Daten durch das Laborteam bereitgestellt. Auf weitere Vorgänge innerhalb der Firma hat allerdings die Tablet-Applikation keinen Einfluss mehr. Deshalb wird hier lediglich auf den Teil der Speicherung in die Datenbank und vor allem die Synchronisation mit der LVA-Datenbank eingegangen.

Innerhalb der Tablet Applikation wird eine SQLite Datenbank verwendet, da diese nicht sonderlich viele Ressourcen des Tablets benötigt und so eine einfache und angenehme Verwendung durch den Benutzer ermöglicht. Die Datenbank erlaubt zudem nicht allzu viele Datenbanken und im Vergleich zur klassischen MySQL Lösung nur wenige Nutzer. Auf dem firmeneigenen Server läuft eine klassische Mysql Instanz. Der Auftraggeber hatte zwar die Wahl offen gelassen welche Datenbanklösung bevorzugt wird, allerdings gleichzeitig auch bekanntgegeben, dass bereits eine Lizenz für Mysql vorhanden ist. Weitere in Betracht gezogene Datenbanksysteme waren PostgreSQL und ebenfalls auch für das Serverprogramm SQLite.

Die Umsetzung sah so aus, dass Daten vom Tablet zunächst an die Server-Applikation über Java-Streams versandt und anschließend von der Server-Applikation entsprechend in die Datenbank eingepflegt wer-

den. Während Daten von der Datenbank auf das Tablet synchronisiert sind, dürfen diese auch von niemandem bearbeitet werden. So wird Konsistenz innerhalb des Systems gewährleistet.[RBDT]

1.2 Datenbanksynchronisation und Ansätze

Die sichere Synchronisation von Daten in verteilten Systemen ist ein wichtiges Anliegen, da der Prozess in erster Linie Konsistenz der Daten gewährleisten muss. Zu diesem Thema gibt es verschiedene Ansätze auf die hier, bezogen auf das oben beschriebene Diplomprojekt, näher eingegangen wird.

1.2.1 Synchronisationsprobleme

Der Kernpunkt ist, dass ein zentrales Datenbanksystem und ein oder mehrere kleinere Systeme existieren. Während die zentrale Datenbank sämtliche Daten des Systems beinhaltet, verfügen die kleineren Datenbanken nur über einen Bruchteil dieser Daten. Diese Bruchteile können von den einzelnen Geräten natürlich jederzeit geändert werden. Die Aufgabe besteht nun, dass all diese Bruchteile von Daten, wieder zur Hauptdatenbank synchronisiert werden ohne, dass größere Konflikte entstehen.

Zusätzlich zu den bereits geschilderten Punkten, werden noch wichtige Nebenpunkte betrachtet.

Die Verbindung der verteilten Datenbanken zum Hauptsystem muss nicht immer vorhanden sein. Das zentrale System muss wissen wann eine Verbindung besteht.

Die verteilten Datenbanken besitzen nur einen Teil der in der Hauptdatenbank gespeicherten Daten, wobei allerdings keiner der Datenbestände mit einem anderen der verteilten Datenbanken überlappt. Nur die zentrale Datenbank besitzt den selben Datenbestand wie eine der verteilten Datenbanken.

Ein Problem stellt hierbei ein Fall dar, wenn eine der verteilten Datenbanken vorübergehend offline geht und währenddessen Daten in der Hauptdatenbank geändert werden. Eine Methode um dies zu beheben

wäre das simple überschreiben der Daten auf der entsprechenden Datenbank, falls einem der beiden Datenbanken eine höhere Priorität zugesprochen wird. Wenn zum Beispiel die verteilte Datenbank eine höhere Priorität besitzt als die zentrale Datenbank, kann die verteilte Datenbank die Daten der Hauptdatenbank einfach überschreiben.

Die Synchronisation der einzelnen Daten findet stets nur zwischen einer verteilten Datenbank und der zentralen Datenbank statt und keinesfalls unter zwei verteilten Datenbanken. Zusätzlich sind die Systemzeiten der einzelnen verteilten Datenbanken nicht synchronisiert. In einem zusammengefasst kommt man zu folgenden Punkten:

- Die Daten sollten konsistent bleiben, auch wenn die Synchronisation einmal fehlschlägt oder keine Verbindung besteht.
- Konflikthandhabung hängt von den einzelnen Tabellen ab
- Verteilte Datenbanken werden hier nur mit der zentralen Datenbank synchronisiert und nicht untereinander
- Datenbankzeiten werden nicht synchronisiert

[WFKM]

1.2.2 Konfliktbehebung

Besonders wichtig bei der Synchronisation von verteilten Datenbanken ist die Behebung von entstandenen Konflikten. Im Idealfall sollten diese natürlich gar nicht erst auftreten, doch muss auch eine passende Strategie für den Fall der Fälle vorhanden sein. Im Allgemeinen besteht die Behebung von Konflikten aus zwei Teilen: die Erkennung und die eigentliche Behebung. Die Erkennung ist hierbei ebenfalls ein wichtiger Faktor, da Fehler und Konflikte so früh wie möglich erkannt werden müssen um Konsistenz sicherzustellen und Datenverlust zu vermeiden. Unter anderem gibt es bei der Konfliktbehebung zwei Aspekte die ausschlaggebend sind. Zum einen ist es wichtig zu wissen, welche Teilnehmer den Konflikt verursacht haben und in welcher Beziehung diese Teilnehmer zu einander stehen. Zu beachten ist, dass mehr als zwei Teilnehmer in einen Konflikt verwickelt sein können.

Der zweite Aspekt sind die eigentlichen in Konflikt stehenden Daten

welche wiederum eigene Kriterien beinhalten die dabei beachtet werden müssen. Unter anderem wo die Daten gespeichert wurden in der Datenbank, welche Datentypen sie haben und welche Werte, bezogen auf die in Konflikt stehenden Teilnehmer, tatsächlich im Konflikt stehen.

Ein Beispiel für die Notwendigkeit des Ortes der Daten für die Konfliktbehebung ist, wenn Daten lediglich in einer Tabelle gespeichert wurden, welche für die Speicherung von temporären Daten zuständig ist. Hier hängt es nun davon ab wie wichtig diese Daten für andere Teilnehmer sind. Hier könnte als Problemlösungsansatz einfach immer der neueste Wert in die Tabelle gespeichert werden. Sinnvoll wäre es unter Umständen ebenfalls die Tabelle komplett aus dem Synchronisationsprozess zu entfernen und eine Ausnahme zu setzen, falls es nicht notwendig ist diese mit anderen Teilnehmern zu synchronisieren.

Des weiteren existieren allgemein Strategien zur Behebung von Problemen bei der Synchronisation.

Eine davon nennt sich zeit-basierte Strategie, bei der schlicht so vorgegangen wird, dass immer der aktuellste Wert übernommen wird. Der offensichtliche Nachteil dieser einfach zu realisierenden Strategie ist, dass sämtliche Updates die kurz danach unternommen wurden verloren gehen. Dies macht diese Strategie in seiner natürlichen Form nicht wirklich nutzbar. Auch wenn andere Strategien mit dieser erweitert werden, ist die Wahrscheinlichkeit eines Lost Update dennoch gegeben. Ein weiteres großes Problem dieser Strategie ist die Notwendigkeit, alle Uhrzeiten der Datenbanken zu jeder Zeit synchron halten zu müssen. Da allerdings Zeitsynchronisation kein triviales Thema ist, könnte zum Beispiel eine logische Uhr verwendet werden.

Eine andere Methode ist die Teilnehmer-basierte Strategie. Diese basiert auf den vorhandenen Datenbanken und den Beziehungen zwischen ihnen. Diese Strategie geht von der Tatsache aus, dass zumindest einer der Teilnehmer wichtiger ist als die anderen. Die Daten der weniger wichtigen Datenbanken würden demnach einfach überschrieben werden. Ein wichtiger Punkt dieser Strategie ist, dass die einzelnen Prioritäten der Datenbanken vorher festgelegt werden müssen. Dies kann, in Abhängigkeit von der Applikation, statisch mit Hilfe eines einfach zugewiesenen Wertes oder dynamisch mit Hilfe eines Algoritmus passieren,

der beispielsweise auf der Netzwerkauslastung basieren kann. Die einfachste Möglichkeit ist, jedem Teilnehmer eine eindeutige Nummer zuzuweisen und nach dieser zu operieren. Wenn dies nicht so gehandhabt wird können Konflikte zwischen zwei gleichwertigen Datenbanken entstehen. Falls hier die Implementierung dieser Strategie nicht ausreicht, kann diese auch beispielsweise mit einer zusätzlichen Strategie erweitert werden. Ein möglicher Einsatzbereich dieser Strategie wäre ein Szenario wo eine zentrale Datenbank existiert und verschiedene verteilte Datenbanken synchronisieren ihre Daten ausschließlich mit dieser zentralen Datenbank. In diesem Fall wird die zentrale Datenbank wohl die höchste Priorität erhalten.

Den kompliziertesten Ansatz für die Konfliktbehebung stellt die Datenbasierte Strategie dar.

2 Couchbase Lite

Couchbase Lite ist eine leichtgewichtige, dokumenten-orientierte und leicht synchronisierbare Datenbank welche speziell für den Einsatz in mobilen Anwendungen und Geräten geeignet ist.

Leichtgewichtig bedeutet:

- Die Datenbank Engine ist eine in der Applikation gebundene Bibliothek und kein separater Serverprozess.
- Sie besitzt sehr klein gehaltenen Code damit Apps die auf die Schnittstelle zurückgreifen rasch heruntergeladen werden können.
- Garantiert eine kurze Startzeit da mobile Geräte meist geringere CPU-Leistung haben als PCs.
- Mobile Datensätze sind zwar relativ klein, allerdings können manche Dokumente große multimediale Anhänge haben, weswegen ein geringe Speicherauslastung von Nöten ist.
- Bietet auch eine gute Performance, wobei diese zu einem großen Teil von der implementierten Applikation abhängt.

Dokumenten-orientiert bedeutet:

- Speichert Einträge im flexiblen JSON-Format, weshalb keine vordefinierten Schemata benötigt werden.
- Dokumente können eine frei wählbare Größe von Binary-Anhängen besitzen, wie z.B. multimediale Inhalte.
- Das Datenformat der Applikation kann sich über die Zeit weiterentwickeln, ohne dass die Datenbank geändert werden muss.
- MapReduce Indizierung bietet eine schnelle Datensatzabfrage, ohne dass spezielle Query-Languages verwendet werden müssen.

Leicht synchronisierbar bedeutet:

- Zwei Kopien einer Datenbank können problemlos über einen Replikations-Algorithmus synchronisiert werden.
- Die Synchronisation kann On-Demand oder fortlaufend stattfinden.
- Geräte können auch nur Teilmengen einer riesigen Datenbasis eines Remote-Servers synchronisieren.
- Die Sync-Engine erlaubt auch das Synchronisieren über unbeständige und unzuverlässige Netzwerkverbindungen.
- Konflikte können einfach über einen Merge-Algorithmus, gefunden und behoben werden.
- Revisions-Bäume erlauben auch komplexe Replikations-Topologien, wie z.B. Server-to-Server (für mehrere Datenzentren) und Peer-to-Peer, ohne Datenverlust oder Konflikte befürchten zu müssen.

[Coua]

2.1 Synchronisation mit Couchbase Sync Gateway

[Coub]

2.2 Vorteile dieser Umsetzung

Literaturverzeichnis

[Coua] Couchbase: *Couchbase Lite - http://docs.couchbase.com/couchbase-lite/cbl-concepts/*

[Coub] Couchbase: *Couchbase Sync Gateway - http://docs.couchbase.com/sync-gateway/*

[RBDT] Rieppel, A. ; Backhausen, D. ; Dimitrijevic, D. ; Traxler, T.: *Diplomarbeit FI@D*

[WFKM] Wohlmuth, B. ; Frank, M. ; Kaltenböck, P. ; Masopust, T.: *Diplomarbeit Buga Tracking*