Nutzerdefinierte Replikation zur Realisierung neuer mobiler Datenbankanwendungen

Christoph Gollmick

Friedrich-Schiller-Universität Jena, Institut für Informatik
Ernst-Abbe-Platz 2, 07743 Jena
gollmick@informatik.uni-jena.de

Kurzfassung: Replikation ist für die unverbundene Verfügbarkeit von Daten in mobilen Umgebungen unverzichtbar. Das Konzept der nutzerdefinierten Replikation erweitert die Möglichkeiten zum Umgang mit replizierten relationalen Datenbankinhalten um die anwendungsgesteuerte dynamische Auswahl von Replikaten. Die nutzerdefinierte Replikation bietet dazu sowohl dem Administrator als auch dem Anwendungsprogrammierer, an SQL angelehnte, deskriptive Schnittstellen. Über diese Schnittstellen werden die zur Replikation bereitgestellten Daten, die Datenauswahl durch die mobile Anwendung sowie die jeweils damit verbundenen Zusicherungen und Konfliktbehandlungsmöglichkeiten spezifiziert. Der Beitrag stellt die neue Funktionalität vor und erläutert die Verwendung der Schnittstellen am Beispiel des interaktiven Reiseinformationssystems HERMES.

1 Einführung und Motivation

Die Computertechnologie wurde in den letzten Jahren um eine Dimension erweitert, die Mobilität von Geräten und Nutzern. Mobile Geräte sind dabei abhängig von unterwegs verfügbaren Kommunikationsmitteln und einer mobilen Energieversorgung. Beide Voraussetzungen sind heute und auch in naher Zukunft noch mit erheblichen Einschränkungen versehen. Der Stand der Technik bietet zwar adäquate drahtlose Kommunikationsmittel wie WLAN oder UMTS, diese sind aber noch nicht weit verbreitet, teuer und energieintensiv in der Benutzung. Außerdem sind sie prinzipbedingt nicht so leistungsstark und sicher gegen Angriffe von Dritten wie feste Netze. Das Problem kurzer Akkulaufzeiten, das durch (drahtlose) Kommunikation verstärkt wird, ist ebenfalls noch ungelöst. Für den jederzeitigen, sicheren und kostengünstigen Zugriff lokaler Anwendungen auf Datenbankinhalte (mobiler Datenbankzugriff) sind deshalb die Möglichkeit zum *zeitweise unverbundenen Arbeiten* mobiler Clients [Gol01, JHE99] und geeignete *Replikationstechniken* [ÖV99, HHB96] erforderlich.

Heutige kommerzielle mobile Datenbanklösungen [Fan00] erlauben sowohl eine isolierte Arbeit mobiler Clients als auch die Integration in eine kooperative Datenverwaltung mit Workstation- und Großrechner-DBS. Ihre Fähigkeiten sind dabei auf die mobile Umsetzung traditioneller Anwendungen (z. B. Zugriff auf ein SAP R/3-System oder die Datenbank für einen Versicherungsvertreter) zugeschnitten. Bei diesen Anwendungen sind die

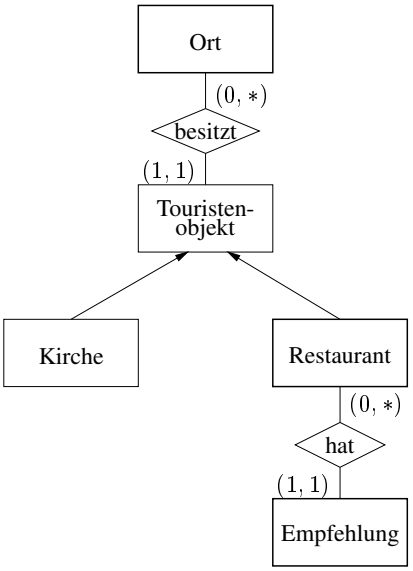
Tätigkeit eines jeden Nutzers und damit die zu replizierenden Daten bekannt und es besteht wenig Konfliktpotential. Die Definition von Replikationsabbildungen, die Datenauswahl für mobile Clients und die Nutzerverwaltung können durch den Administrator auf der Serverdatenbank bzw. einem zwischengeschalteten Synchronisationsserver erfolgen (*zentralistischer Ansatz*). Die Fähigkeiten zur automatischen Behandlung von Änderungskonflikten beschränken sich auf einfache, attributorientierte, Strategien. In der Praxis sind deshalb bei komplexeren Anwendungen eigene anwendungsabhängige Konfliktbehandlungsprozeduren oder sogar ein manueller Eingriff erforderlich.

Die Mobilität von Nutzern bietet über die Bereitstellung klassischer Dienste hinaus aber auch Chancen für die Entwicklung ganz neuer datenbankgestützter mobiler Anwendungen. Vorstellbar sind Anwendungen, die dem mobilen Nutzer zum einen auf seinen Aufenthaltsort und seine aktuelle Tätigkeit zugeschnittene Daten lokal zur Verfügung stellen und zum anderen die Interaktion zwischen Nutzern und dem System ermöglichen. Das heißt, Nutzer sollen unmittelbar Vorort neue Daten und Änderungen an gemeinsam genutzten Daten ins System einbringen können. Beispiele solcher Anwendungen sind Unterstützungssysteme für Katastrophenhilfskräfte oder das, im Rahmen dieses Projektes entwickelte, interaktive mobile Reiseinformationssystem HERMES [Bau03]. Diese neuen Anwendungsszenarien stellen andere Anforderungen an die Datenhaltungskomponente eines Informationssystems als klassische Anwendungen. Dazu gehört ein Dienst, über den eine mobile Anwendung Daten dynamisch zur Laufzeit, abhängig von der Zeit, dem Ort und der aktuellen Tätigkeit ihrer Nutzer für unverbundene Zeiten zur Replikation auswählen kann – *nutzerdefinierte Replikation* (NR). Wie ein solcher Dienst gestaltet werden kann und welche damit zusammenhängenden Probleme gelöst werden müssen, zeigt dieser Beitrag.

Der Beitrag ist wie folgt gegliedert. Abschnitt 2 stellt das mobile Reiseinformationssystem HERMES und den zur Illustration verwendeten Datenbankausschnitt vor. In Abschnitt 3 werden die charakteristischen Eigenschaften und das Arbeitsmodell der nutzerdefinierter Replikation anhand eines Beispiels erläutert. Für die Realisierung der NR ist u. a. eine effiziente Verwaltung dynamisch replizierter Nutzerdaten erforderlich, Abschnitt 4 gibt einen Überblick zur prototypischen Umsetzung. Abschnitt 5 schließt mit einer Zusammenfassung und einem Ausblick auf noch zu lösende Probleme und mögliche Erweiterungen.

2 Das Reiseinformationssystem HERMES

Das datenbankbasierte Reiseinformationssystem HERMES hat die Aufgabe, seine Nutzer bei der Reiseplanung zu unterstützen und ihnen unterwegs ortsabhängige Informationen bereitzustellen. Abrufbar sind beispielsweise Informationen über Orte, Sehenswürdigkeiten, historische Gebäude, Hotels, Restaurants oder auch empfehlenswerte Routen. Im Unterschied zu klassischen Touristenführern werden aber nur die Basisdaten, wie Name, Einwohnerzahl oder Historie von Orten, zentral bereitgestellt. Der weitaus größte Teil wird von den mobilen Nutzern selbst und in der Regel vor Ort ergänzt und aktualisiert. Hierzu zählen beispielsweise Reiseberichte/Bewertungen, besuchte Routen, Öffnungszeiten von Gebäuden, Preise und Erfahrungen mit Hotels oder Speisekarten von Restaurants. HERMES ist also ein mobiles Reiseinformationssystem von Nutzern für Nutzer.



(a) E/R-Modell

Ort	Name	Land	Einwohner
	München	Bayern	1300000
	Erfurt	Thüringen	220000
	Jena	Thüringen	100000

Kirche	Name	Ort	e/k
	Friedenskirche	Jena	e
	Stadtkirche	Jena	e
	Peterskirche	München	k

Restaurant	Nr	Name	Ort
	1	Zur Noll	Jena
	2	Roter Hirsch	Jena
	3	Hohe Warte	Erfurt
	4	Brotzeitstüberl	München

Empfehlung	Von	RNr	Text
	Erwin	1	„gut“
	Egon	1	„na ja“
	Egon	2	„geht so“

(b) Eine relationale Umsetzung

Abbildung 1: Beispiel

Die angestrebte Funktionalität und die Verwendung von HERMES soll an einem einfachen Beispiel (Abbildung 1, *Restaurant.Ort* und *Kirche.Ort* sind jeweils Fremdschlüssel zur Tabelle *Ort*, *Empfehlung.RNr* ist Fremdschlüssel bzgl. *Restaurant* und es gilt `UNIQUE`(*Restaurant.Name*, *Restaurant.Ort*)) skizziert werden. Nach der Ankunft in Jena möchte sich ein Reisender über die hiesigen Kirchen informieren und wählt über die Oberfläche der lokalen HERMES-Anwendung auf seinem PDA die gewünschten Informationen aus, die nach einer Verbindungsaufnahme vom zentralen Server auf sein Gerät übertragen werden. Während des Stadtbummels kann er auf die replizierten Informationen offline zugreifen. Nach dem Stadtbummel möchte er sich zum Abendessen ein gutes Restaurant wählen. Dazu selektiert er die Daten der Restaurants mit den Nutzerempfehlungen. Nach dem ausführlichen Studium der Informationen entscheidet er sich für das Restaurant „Zur Noll“. Auf dem Weg dorthin kommt er am Restaurant „Ratszeise“ vorbei und entscheidet sich spontan für diesen Gastronomiebetrieb. Da die „Ratszeise“ noch nicht im Restaurantkatalog verzeichnet ist, fügt er ihre Daten ein. Am nächsten Morgen synchronisiert er seine Änderungen mit der zentralen Datenbank.

Dieses kleine Beispiel beinhaltet bereits die meisten der zu lösenden Fragestellungen und Probleme, die wir mit dem Konzept der nutzerdefinierten Replikation adressieren wollen. Die Anwendung muß die Möglichkeit haben, nach den Vorgaben des Nutzers, geeignete Datenmengen vom Server zur Laufzeit zu replizieren. Bereits lokal vorhandene und noch aktuelle Daten sollen dann nicht erneut repliziert werden. Für den Fall, daß der lokale

Speicherplatz nicht mehr ausreicht, müssen replizierte Daten wieder gelöscht werden. Für das Hinzufügen von Restaurants muß dem mobilen Nutzer mindestens das Einfügerecht auf der Tabelle *Restaurant* gewährt werden und eine automatische Konfliktbehandlung muß für die Konsistenz der Daten bei der Synchronisation sorgen.

	traditionelle Anwendungen	HERMES
Nutzeranzahl	begrenzt, stabil	potentiell unbegrenzt, variabel
Datenauswahl		
◦ Wo?	zentral	durch Nutzer
◦ aufgabenabhängig	ja	ja
◦ ortsabhängig	selten	häufig
◦ Variabilität	gering	hoch
◦ Änderungskonflikte	selten	häufig

Tabelle 1: Vergleich traditioneller mobiler Anwendungen mit HERMES

Tabelle 1 stellt die charakteristischen Eigenschaften von HERMES denen der traditionellen Anwendungen gegenüber. HERMES steht dabei stellvertretend für eine Klasse von neuen mobilen Datenbankanwendungen, die sich durch die Forderung nach einer dynamischen, orts-, aufgaben- und zeitabhängigen Datenauswahl auszeichnen.

3 Die nutzerdefinierte Replikation

Trotz ihrer Fokussierung auf den zentralistischen Ansatz bieten einige kommerzielle mobile Datenbanklösungen (z. B. Oracle 9i lite) der mobilen Anwendung die Möglichkeit, über die Definition materialisierter Sichten aus einer vorher für die Replikation vorbereiteten Datenmenge auszuwählen. Für die Unterstützung einer großen Zahl von Nutzern mit dynamischer Datenauswahl fehlen jedoch wichtige Elemente, wie eine einfach zu handhabende Konfliktbehandlung und eine effiziente Verwaltung replizierter Daten sowohl auf dem Client als auch auf dem Server. Mit der Konzeption der nutzerdefinierten Replikation für relationale Datenbankinhalte verbinden wir die folgenden Ziele:

- Bereitstellung einer Schnittstelle für mobile Anwendungen, die es erlaubt, zur Laufzeit Daten aufgaben-, orts-, und zeitabhängig in Verbindung mit Zusicherungen (z. B. konfliktfreie Änderbarkeit) zur Replikation anzufordern.
- Wo möglich, Bereitstellung deskriptiver, an SQL angelehnter, Konstrukte sowohl für den Administrator als auch für den Anwendungsprogrammierer. Dieses Ziel soll insbesondere für die Spezifikation der Konfliktbehandlung verfolgt werden.
- Ein weiteres wichtiges Ziel ist die Trennung der Aspekte der Replikatdefinition (Auswahl zu replizierender Daten durch die Anwendung) von den Aspekten der Replikatverwaltung (Anlegen von Metadaten für die Synchronisation, Caching/Prefetching-Verfahren, etc.), die in kommerziellen Systemen oft eng verknüpft sind. Auf die Re-

plikatverwaltung und eine mögliche Realisierung der NR werden wir in Abschnitt 4 kurz eingehen.

In einer mobilen Replikationsumgebung gibt es zwei grundlegende Aktivitäten, die *Replikationsdefinition* und die *Synchronisation*. Beide Aktivitäten sind in das *Arbeitsmodell* eingebettet, das Voraussetzungen und Abläufe vorgibt.

3.1 Replikationsdefinition

Bei der Replikationsdefinition wird eine Auswahl getroffen, welche Daten eines Quellsystems (*Replikationsquelle*) auf dem Zielsystem (*Replikationsziel*) repliziert verfügbar sein sollen (*Replikate*), für wen welche Operationen auf den Replikaten erlaubt sind (*Zusicherungen*) und welche Konfliktvermeidungs-, -erkennungs-, und -auflösungsstrategien (*Konfliktbehandlung*) für die Replikate Verwendung finden. Als Replikationsquelle sind bei der NR nur stationäre Server erlaubt, Replikationsziele sind die mobilen Clients.[1] Die Replikationsdefinition gliedert sich in die zwei Unteraktivitäten *Replikationsschemadefinition* und *Replikatdefinition*:

1. Replikationsschemadefinition

   Das *Replikationsschema* ist der Ausschnitt des Schemas einer relationalen Datenbank auf der Replikationsquelle, der für die Replikation auf mobile Clients sichtbar ist, erweitert um Zusatzinformationen. Zum Replikationsschema gehören Zusicherungen für mögliche Operationen (z. B. lokal änderbar) sowie alle zur Synchronisation notwendigen Erweiterungen, sofern sie sich auf Elemente der Schemadefinition (z. B. Datentypen, Integritätsbedingungen) beziehen.[2] Die *Replikationsschemadefinition* ist bei der NR, wie beim zentralistischen Ansatz, Aufgabe eines Administrators.

2. Replikatdefinition

   Bei der *Replikatdefinition* werden, bezogen auf ein Replikationsschema, die für eine Aufgabe, einen Zeitpunkt und einen Ort benötigten Daten ausgewählt und die bei der Replikationsschemadefinition festgelegten Zusicherungen und Konfliktbehandlungsmethoden instantiiert. Die Definition der Replikate wird bei der NR, meist auf Wunsch des Nutzers, durch die mobile Anwendung durchgeführt. Ein Client kann beliebig viele Replikate anlegen lassen, geforderte Zusicherungen in gewissem Rahmen ändern und nicht mehr benötigte Replikate löschen.

3.2 Synchronisation

Da es über die Replikation mehrere Kopien eines Datums bei verschiedenen mobilen Clients geben kann, auf denen unabhängig voneinander lokal gearbeitet wird, ist eine *Synchronisati-*

[1] In Anbetracht der exponierten Stellung mobiler Clients und der angedachten Nutzerzielgruppe verbieten wir, daß Primärkopien gemeinsam genutzter Daten auf mobilen Clients liegen.

[2] Das Anlegen eines *Key Pool* zur Einfügekonfliktvermeidung für einen Schlüsselkandidaten gehört zum Replikationsschema. Die Bereitstellung einer konkreten Anzahl von Schlüsseln aus dem Pool gehört zur *Replikatdefinition*.

on untereinander und mit der Replikationsquelle erforderlich. Ziel ist dabei die Aufrechterhaltung oder Wiederherstellung eines, für alle Beteiligten, konsistenten Datenbankzustands. Bei der nutzerdefinierten Replikation erfolgt die Synchronisation immer zwischen einem Client und dem Server.[3] Die Synchronisation von Client und Replikationsquelle besteht aus zwei Phasen, der *Reintegration* und der *Rückübertragung*:

1. Reintegration

   Bei der *Reintegration* werden lokale Änderungen an Replikaten beim Server eingebracht und der nachgelagerten Integritätsprüfung sowie einer Konfliktbehandlung (primärschlüsselorientierte Konflikterkennung) unterzogen. Die Synchronisation der NR orientiert sich dabei an der in [GHOS96] vorgestellten transaktionsorientierten *Two-Tier-Replication*. Dabei werden unverbunden lokal ausgeführte Änderungstransaktionen auf der Replikationsquelle mit aktuellen Daten erneut ausgeführt, wo sie ihr endgültiges Commit oder Abort erfahren.

2. Rückübertragung

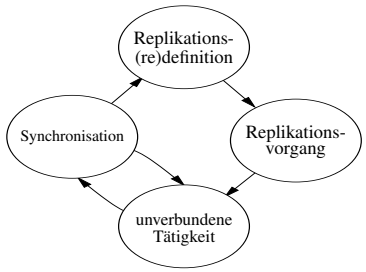
   Bei der *Rückübertragung* vom Server wird ein aktueller transaktionskonsistenter, u. U. konfliktbereinigter, Zustand der zur Replikatdefinition gehörenden Daten an den Client übermittelt.

Es ist zu beachten, daß für den kompletten Reintegrationsprozeß eines mobilen Client die Transaktionseigenschaften *Konsistenz* (für den Zustand der Replikationsquelle) und *Isolation* erfüllt sein müssen.[4] Dies kann beispielsweise durch eine Serialisierung der Reintegrationsprozesse mehrerer Clients erreicht werden.

3.3 Arbeitsmodell

Wir vertreten die Meinung, daß die Akzeptanz von HERMES entscheidend von zwei Punkten abhängt, der Verfügbarkeit der vollen Funktionalität am „Ort der Information“ und geringen (Übertragungs-)Kosten. Die Möglichkeit unverbundener Tätigkeit hat deshalb in unseren Betrachtungen einen hohen Stellenwert. Das Arbeitsmodell der aktuellen Konzeptspezifikation unterscheidet nur zwischen *verbundenem* und *unverbundenem* Zustand eines mobilen Client (siehe Abschnitt 5). Die Voraussetzung für unverbundenes Arbeiten ist ein ausreichend dimensionierter Client mit einem lokalen DBS, das transaktionsorientierte Synchronisation beherrscht (siehe Produktübersicht in [Fan00]).

Abbildung 2: Ablauf der NR



Die Beschreibung des Ablaufs bei der nutzerdefinierten Replikation (Abbildung 2) soll anhand einer Umsetzung des zweiten Teils des Beispielszenarios aus Abschnitt 2 (Restaurantbesuch) erfolgen. Für eine Beschreibung von Syntax und Semantik der hier vereinfacht dargestellten Konstrukte sei auf [Mül03] verwiesen.

[3] Der Abgleich zwischen gleichberechtigten Clients schafft eine Reihe zusätzlicher (Konsistenz-)Probleme (z. B. weiß nicht jeder Client über die Replikatdefinition eines anderen Bescheid), die wir in der aktuellen Konzeptspezifikation vermeiden wollen.

[4] Atomarität (und Dauerhaftigkeit für lokale Transaktionen) kann nicht gewährleistet werden, da die Reintegration einzelne lokale Transaktionen zurückweisen kann.

1. Der Administrator gibt die Tabellen *Restaurant* und *Empfehlung* der Server-Datenbank zur Replikation frei in dem er *konsolidierte Tabellen* (siehe Abschnitt 4), als Elemente des Replikationsschemas, anlegt:

```
CREATE CONSOLIDATED TABLE ConsRestaurant
SOURCE Restaurant
FOR OFFLINE INSERT ADD KEYPOOL (Nr) DEFAULT 1 MAX 5
FOR OFFLINE DELETE
DEFAULT RESOLUTION DISCARD;

CREATE CONSOLIDATED TABLE ConsEmpfehlung
SOURCE Empfehlung
FOR OFFLINE INSERT FOR OFFLINE DELETE
FOR OFFLINE UPDATE (Text)
DEFAULT RESOLUTION DISCARD;
```

   In beide konsolidierten Tabellen kann eingefügt werden, für den Primärschlüssel der Tabelle *ConsRestaurant* wurde vom Administrator die Konfliktvermeidungsstrategie `KEYPOOL` ausgewählt. Die mobile Anwendung kann maximal 5 Schlüssel gleichzeitig reservieren, Default-Wert ist 1 Schlüssel.[5] Als Konfliktauflösungsstrategie wurde festgelegt, die vorläufigen Änderung des *Client* zu verwerfen.

2. Die Anwendung des Reisenden fordert im *verbundenen* Zustand von der Replikationsquelle die Daten der Jenaer Restaurants zur Replikation an:

```
CREATE REPLICATION VIEW RepRestaurantJena
AS SELECT Nr, Name, Ort FROM ConsRestaurant
   WHERE Ort='Jena'
FOR OFFLINE MODIFICATION WITH CHECK OPTION;

CREATE REPLICATION VIEW RepEmpfehlungJena
AS SELECT Von, RNr, Text FROM ConsEmpfehlung
   WHERE RNr IN (SELECT Nr FROM ConsRestaurant
                 WHERE Ort='Jena')
FOR OFFLINE MODIFICATION WITH CHECK OPTION;

SYNCHRONIZE ALL;
```

   Die zu replizierenden Daten werden über den `SELECT`-Teil der `CREATE REPLICATION VIEW`-Anweisung bestimmt. Standardmäßig ist eine `REPLICATION VIEW` (RV) nicht änderbar, werden aber Angaben zu `FOR OFFLINE` gemacht, so wird sie, wenn die Rückabbildung möglich ist, für unverbundene Modifikationen bereitgestellt. Optional können Parameter für die Konfliktbehandlung angegeben werden.

   Nach dem erfolgreichen `SYNCHRONIZE ALL` steht der mobilen Anwendung eine Sicht zur Verfügung, welche die in der RV-Definition selektierten und replizierten

[5] Alternativ könnte der nutzerdefinierte Replikationsdienst die Anzahl benötigter Schlüssel für einen Client anhand der Häufigkeit der Einfügungen zwischen zwei Synchronisationsintervallen in der Vergangenheit bestimmen.

Daten referenziert. Auf eine solche Sicht können lokale Anfragen, Sicht- und Cursor-Definitionen aufsetzen. Die angegebenen Beispielkonstrukte erzeugen auf dem mobilen Client die zwei unverbunden änderbaren RVs *RepRestaurantJena* und *RepEmpfehlungJena*.

Wichtig ist, daß eine RV-Definition vollständig unabhängig von den tatsächlich replizierten Daten ist. Die Replikationsquelle entscheidet bei der Auswertung von `CREATE REPLICATION VIEW`, welche Daten für die geforderten Zusicherungen und unter Berücksichtigung der schon lokal vorhandenen Daten auf den Client repliziert werden müssen. Eine mobile Anwendung kann damit ohne redundante Datenhaltung dynamisch beliebige RVs, auch mit überlappender Definition, anlegen und wieder löschen. Das Löschen einer RV (im verbundenen Zustand) signalisiert dem System, daß die von der Sicht referenzierten Daten nicht mehr benötigt werden. Die betroffenen Replikate und ihre Metadaten werden, sofern sie von keiner anderen RV mehr benötigt werden, daraufhin aufgelöst.

3. Im *unverbundenen* Zustand fügt der Nutzer über seine HERMES-Anwendung einen neuen Datensatz in die RV *RepRestaurantJena* der lokalen Datenbank ein:

   ```
   INSERT INTO RepRestaurantJena
   VALUES (Nr_VALUE(), 'Ratszeise', 'Jena');
   ```

   `Nr_VALUE()` stellt dabei für den Primärschlüssel der Quelltabelle einen Schlüsselwert aus dem lokalen *Key Pool* bereit.[6]

   Bei der RV-Definition haben wir auch eine änderbare Kopie der *ConsEmpfehlung*-Tabelle angefordert. Wir könnten also beispielsweise die Empfehlungen eines anderen Nutzers aus der Datenbank löschen. Das zeigt, daß die Realisierung von Zusicherungen eng an die Verfügbarkeit einer inhaltsbasierten Authentisierung und Rechteverwaltung gekoppelt ist, wie sie von kommerziellen DBMS nicht unterstützt bzw. der jeweiligen Anwendung überlassen wird. Der Entwurf der nutzerdefinierten Replikation sieht ein solches Rechtekonzept vor [Mül03], wir werden hier jedoch nicht darauf eingehen.

4. Mit der anschließenden Synchronisation kann, bei positivem Ausgang der Konfliktauflösung, die Dauerhaftigkeit der lokalen Änderungen erzielt werden. Sollte ein anderer Restaurantbesucher bereits früher einen Eintrag zur „Ratszeise“ reintegriert haben, wird die eigene Änderung verworfen.

4 Realisierung der NR

Die prototypische Realisierung der nutzerdefinierten Replikation wird zur Zeit im Rahmen mehrerer Arbeiten betrieben (Architektur [Mül03], Konfliktbehandlung [Lie03] und Replikatspeicherung [Gol02a]). Über die am Anfang von Abschnitt 3 formulierten allgemeinen Ziele hinaus fordern wir von einer Umsetzung: Skalierbarkeit in der Anzahl der mobilen Clients, Reduktion von Verbindungszeiten und Flexibilität bzgl. der Geräte/Produkte für

[6] Für eine weitere Einfügung müßte der *Key Pool* im verbundenen Zustand zunächst wieder aufgefüllt werden.

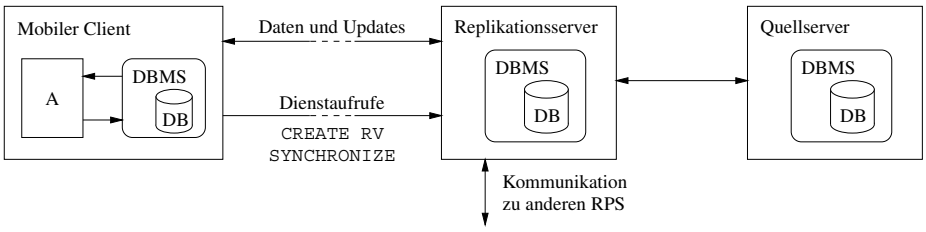


Abbildung 3: RPS-Architektur zur Realisierung der NR

Replikationsquellen und -ziele. Diese Anforderungen gelten zwar auch für die zentral administrierte Replikation, sind aber bei der NR besonders problematisch, da Anzahl und Verhalten der mobilen Nutzer nicht im voraus detailliert bekannt sind.

Die von uns favorisierte 3-stufige Architektur [Gol02b] ist in Abbildung 3 dargestellt. Die nutzerdefinierte Replikation der Server-Daten wird dabei von einem oder mehreren zwischengeschalteten Replikationsservern (RPS – *Replication Proxy Server*) vermittelt,[7] die später auch geographisch verteilt sein sollen. Im Unterschied zu kommerziell erhältlichen Synchronisationsprodukten, hält ein RPS über stationäre asynchrone Replikation eine Kopie (konsolidierte Tabellen) der Quellserver-Daten. Damit wird eine Entkopplung der Arbeit mobiler Nutzer vom Quellserver erreicht. Dies erhöht Verfügbarkeit und Sicherheit und gibt uns die Möglichkeit, die replizierten Daten auf dem RPS für die erfaßten Zugriffs- und Mobilitätsprofile der Clients optimal zu (re-)organisieren.

Die zwei wichtigsten Aufgaben eines RPS sind die automatische Anlage von Daten- und Schemaelementen für die Änderungserfassung und die Konfliktbehandlung (z. B. die Auslesefunktion `Nr_VALUE`) sowie die Bestimmung der Daten und Schemaelemente die, unter Berücksichtigung der bereits lokal vorhandenen, tatsächlich übertragen werden müssen. Der zweite Punkt wirft zwei Fragen auf, wie kann ich Daten semantisch gruppieren und wie kann bei gegebener Anfrage eine minimale Menge zu replizierender Granulate bestimmt werden. Diese Fragen haben wir in [Gol02a] untersucht und ein Fragmentkonzept für relationale Daten entwickelt.

5 Zusammenfassung und Ausblick

In Abgrenzung zu traditionellen Anwendungen haben wir eine Klasse neuer mobiler Datenbankanwendungen untersucht, die eine dynamische, orts-, aufgaben- und zeitabhängige Replikation von Daten erfordern. Zur Unterstützung dieser Anwendungsszenarien wurde das Konzept der nutzerdefinierten Replikation für relationale Datenbankinhalte entwickelt, dessen Funktionalität und Schnittstellen am Beispiel der Anwendung HERMES erläutert wurden.

Die prototypische Umsetzung der nutzerdefinierten Replikation ist im Rahmen mehrerer Arbeiten eingeleitet. Dennoch müssen eine Reihe von Teilproblemen, besonders im Bereich der automatischen Konfliktbehandlung, noch konzeptuell gelöst werden. So sind,

[7] Die Replikationsquelle für die mobilen Clients ist der RPS, nicht der eigentliche Quellserver.

für den Fall einer erfolglosen automatischen Konfliktauflösung, der Anwendung geeignete Informationen für eine Einbeziehung des Nutzers bereitzustellen (z. B. Handynummer des Konfliktpartners). Als Arbeitsmodell haben wir zunächst nur den wichtigen Fall der vollständig unverbundenen Arbeit betrachtet. In der Praxis finden sich jedoch auch Kommunikationsmittel, die für eine Datenübertragung ungeeignet sind, aber die Übermittlung von Status-Informationen erlauben (z. B. SMS zur Übermittlung von Sperrfreigaben). Die damit verbundenen Möglichkeiten sind zu analysieren und, in einem zweiten Schritt, geeignet in das Konzept zu integrieren.

Literaturverzeichnis

[Bau03] K. Baumgarten. Konzept und Realisierung des interaktiven Reiseinformationssystems HERMES. Studienarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2003. In Vorbereitung.

[Fan00] T. Fanghänel. Vergleich und Bewertung kommerzieller mobiler Datenbanksysteme. Studienarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, September 2000.

[GHOS96] J. Gray, P. Helland, P. O'Neil und D. Shasha. The Dangers of Replication and a Solution. In *Proc. of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada*, Seiten 173–182. ACM, 1996.

[Gol01] C. Gollmick. Unterstützung mobiler Clients durch erweiterte Client/Server-Datenbanksysteme. In *Tagungsband zum 13. GI-Workshop „Grundlagen von Datenbanken", Gommern, Deutschland*, Preprint Nr. 10, Seiten 43–47. Fakultät für Informatik, Universität Magdeburg, Juni 2001.

[Gol02a] C. Gollmick. Konzept und Anforderungen der nutzerdefinierten Replikation zur Realisierung neuer mobiler Datenbankanwendungen. Jenaer Schriften zur Mathematik und Informatik Math/Inf/15/02, Institut für Informatik, Friedrich-Schiller-Universität Jena, September 2002.

[Gol02b] C. Gollmick. Using Replication Proxy Servers for Scalable Mobile Database Access. In *Proc. of the EDBT 2002 PhD Workshop, Prag, Tschechische Rep.*, Seiten 115–118. MATFYZPRESS, März 2002.

[HHB96] A. Helal, A. Heddaya und B. Bhargava. *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, August 1996.

[JHE99] J. Jing, A. Helal und A. Elmagarmid. Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, 31(2):117–157, 1999.

[Lie03] M. Liebisch. Synchronisationskonflikte beim mobilen Datenbankzugriff: Vermeidung, Erkennung, Behandlung. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2003. In Vorbereitung.

[Mül03] T. Müller. Architektur und Prototyp eines Replication Proxy Server für die nutzerdefinierte Replikation von Datenbankinhalten. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2003. In Vorbereitung.

[ÖV99] T. Özsu und P. Valduriez. *Principles of Distributed Database Systems*. Prentice-Hall, Inc., 2. Auflage, 1999.