View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Universität Rostock, Lehrstuhl Datenbank- und Informationssysteme: Dbis Repository

„Dynamische Replikationsverfahren für den mobilen Datenbankeneinsatz“

Diplomarbeit
Universität Rostock
Fachbereich Informatik

vorgelegt von : Torsten Reincke
geboren am : 19.09.1972

Betreuer : Prof. Dr. Andreas Heuer
Prof. Dr. Heidrun Schumann
Astrid Lubinski

Abgabedatum : 22.12.1999

INHALTSVERZEICHNIS

1 Einführung und Gliederung

1.1 Einführung

Diese Arbeit entstand im Rahmen des MoVi-Projektes und beschäftigt sich mit den durch die Mobilität hervorgerufenen Problemen beim Einsatz von Replikation in mobilen Anwendungsszenarien.
Im Rahmen des MoVi-Projektes (Mobile Visualisierung) wird angestrebt, eine offene und adaptive Visualisierungsarchitektur für ein global verteiltes Informationssystem zu schaffen, durch die der mobile Zugang zu allen denkbaren Daten und Diensten ermöglicht wird. Dazu erfolgte eine Unterteilung in 4 Gebiete :

1. Visualisierung
2. Interaktion und Anfrageerstellung
3. Datenaustausch
4. Informationsbereitstellung

In Bezug auf den letzten Punkt wird am Lehrstuhl Datenbanken und Informationssysteme des Fachbereichs Informatik der Einfluß sämtlicher Faktoren, die Einfluß auf die Mobilität haben, auf die Informationsbereitstellung in mobilen verteilten Systemen untersucht. Diese Einflußfaktoren werden unter dem Oberbegriff mobile Kontexte zusammengefaßt. Der Einsatz der Replikation in mobilen Systemen hat in erster Linie zum Ziel, die Verfügbarkeit von Informationen zu erhöhen. Weitere durch mobile Kontexte geforderte Ziele können die Wahrung der Konsistenz sowie die Minimierung der Kommunikationskosten sein. Im Gegensatz zu normalen verteilten Systemen sind aber in mobiler Umgebung aufgrund der durch die Mobilität hervorgerufenen Orts- und somit Kontextwechsel andere Verfahrenskonzepte für die Verwaltung und Aktualisierung von Replikaten gefragt. Neben begrenzten Netz- und Rechnerressourcen, die

eine Updatepropagierung erschweren oder unmöglich machen, können sich die Anforderungen an ein Replikationsverfahren aufgrund der Dynamik mobiler Kontexte häufig ändern. Davon sind im Besonderen betroffen :

- Replikationsziele :

  Diese sind je nach Kontext unterschiedlich gewichtet, d.h. ein oder mehrere der obig genannten Ziele haben eine höhere Priorität. Bei einem Kontextwechsel verschiebt sich diese Wichtung.
- Replikationsverfahren :

  Um auf diese Wechsel zu reagieren, erfolgt ein Modifikation der einzelnen Verfahrensparameter. Die sich dabei ergebenden Grenzen können durch einen Wechsel des Verfahrens überwunden werden.

Das Ziel dieser Arbeit ist somit, die Modifikation der Ziele bzw. der Verfahren speziell auf mobile Kontexte zuzuschneiden. Im Einzelnen werden folgende Aufgaben bewältigt :

1. Es werden die konkreten Auswirkungen von Kontextwechseln auf Replikationsziele untersucht.
2. Anhand eines konkreten Verfahrens wird eine Parametrisierung für diese Verfahren vorgeschlagen, durch die eine Wichtungs-verschiebung der Replikationsziele aufgefangen wird.
3. Für das konkrete Verfahren wird eine Testimplementation angestrebt, innerhalb dieser durch Parameterwechsel eine Reaktion auf die Änderung des Kontextes und somit des Replikationsziels erfolgt.

Diese drei Punkte werden zu einem Großteil für ein spezielles Anwendungsszenario umgesetzt. Dafür wird die in [Web98] erstellte Literaturdatenbank genutzt. Zum Abschluß erfolgt eine allgemeine Einordnung des Verfahrens und die Benennung weiterer Probleme für spätere Arbeiten.

1.2 Gliederung

Im folgenden Kapitel 2 wird ein Überblick über Replikationsverfahren in stationären verteilten Systemen gegeben. In Kapitel 3 werden mobile Kontexte definiert, ihre wichtigsten Eigenschaften genannt und verschiedene Klassifikationsmöglichkeiten für Kontexte vorgegeben. Das im weiteren Verlauf der Arbeit verwendete Anwendungsszenario der Literaturdatenbanken wird in Kapitel 4 vorgestellt. Die Untersuchung der Auswirkungen mobiler Kontexte auf Replikationsziele sowie eine Bewertung der bisher genannten Verfahren im Bezug auf deren Verwirklichung erfolgt in Kapitel 5. In Kapitel 6 werden dann einige existierende, speziell für mobile Umgebungen erstellte Replikationsverfahren vorgestellt und eines für die weitere Bearbeitung ausgewählt. Eine Parametrisierung für diese Verfahren, speziell ausgerichtet auf das Literaturdatenbankenszenario wird in Kapitel 7 vorgenommen. Kapitel 8 beschreibt dann die Implementationsumgebung und den –verlauf. Schließlich erfolgt in Kapitel 9 eine allgemeine Einordnung des in Kapitel 7 vorgestellten Verfahrens und eine Problembeschreibung für weitere Arbeiten, bevor in Kapitel 10 eine Zusammenfassung den Abschluß bildet.

2 Replikation in verteilten Systemen

2.1 Allgemeines

In den letzten Jahren hat der Einsatz verteilter Systeme im Gegensatz zu zentral verwalteten Systemen stark zugenommen. Ein Grund dafür ist, daß aufgrund eines Rechnerausfalls das Arbeiten im Gesamtsystem in eingeschränktem Maße weiterhin möglich ist, während dies bei einem zentralen System zum Totalausfall führen würde. In verteilten Datenbanksystemen gibt es dabei zwei Möglichkeiten mit den vorhandenen Datenbeständen zu arbeiten, die Arbeit mit und ohne Replikate.

Definition 2.1 Replikat
Ein Replikat ist eine von mehreren Kopien eines Fragmentes.

Durch die Arbeit mit Replikaten wird eine Erhöhung der Verfügbarkeit des Gesamtsystems und ein effizienterer Zugriff auf Datenbestände erreicht. Dadurch erhöht sich gleichzeitig die Fehlertoleranz des Systems. Nachteile der Replikation sind hingegen ein höherer Kommunikations- und Verwaltungsaufwand sowie das mögliche Auftreten von Inkonsistenzen durch Rechner- oder Netzausfälle.
Es wird von replizierten Systemen gefordert, daß sie sich wie nicht replizierte Systeme verhalten sollen. Ein Maßstab dafür ist die sogenannte 1-Kopie-Serialisierbarkeit.

Definition 2.2 1-Kopie-Serialisierbarkeit
Ein Schedule S einer replizierten Datenbank heißt 1-Kopie-serialisierbar, wenn es mindestens eine serielle Ausführung der Transaktion dieses Schedules auf einer nicht replizierten Datenbank

gibt, welche die gleiche Ausgabe sowie den gleichen Datenbankzustand wie S auf der replizierten Datenbank erzeugt (nach [Sar97])

Somit existiert bei der Datenreplikation nach [BeDa95] ein Zielkonflikt zwischen maximaler Verfügbarkeit und der Korrektheit des Gesamtsystems, der in Abbildung 1.1 noch einmal kurz dargestellt wird.

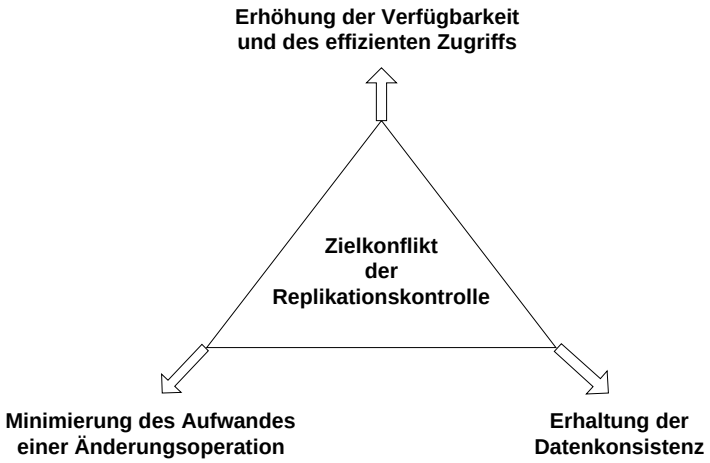

Abbildung 2.1 : Zielkonflikt der Replikationskontrolle [BeDa95]

Zur Lösung dieses Zielkonflikts existieren eine Vielzahl sogenannter Replikationsverfahren, die nach den verschiedensten Kriterien unterschieden werden können. Ein Hauptkriterium dabei ist, nach welcher Art und Weise und aufgrund welcher Informationen diese Verfahren die Sicherung der Konsistenz gewährleisten. Dabei wird in erster Linie zwischen pessimistischen und optimistischen Verfahren unterschieden. Optimistische Verfahren gehen von der Annahme aus, daß Inkonsistenzen nur selten auftreten und daß diese erkannt und aufgelöst werden können. Dabei machen sie sich häufig semantisches Wissen über die Transaktionen zunutze, wie z.B. die Kommutativität einzelner Operationen. Pessimistische Verfahren hingegen gehen immer vom worst case aus, d.h. falls es zu Inkonsistenzen kommen kann, treten diese auch auf. Deshalb wird die Verfügbarkeit der Daten

zugunsten ihrer Konsistenz eingeschränkt. Im folgenden sollen nun einige dieser Verfahren näher vorgestellt werden.

2.2 Pessimistische Replikationsverfahren.

Im Bereich der pessimistischen Replikationsverfahren gibt es drei Klassen, nach denen unterschieden wird. Dies sind die sogenannten Primary-Verfahren, die Voting-Verfahren und die ROWA-Verfahren, welche nun im Einzelnen näher vorgestellt werden sollen.

2.2.1 Primary-Verfahren

Dieses Verfahren ist eine zentralistisch angelegte Methode der Replikationskontrolle. Schreibanforderungen werden zunächst immer nur an eine sogenannte Primärkopie gestellt, die dann eine kopienübergreifende Transaktionssynchronisation regelt. Verändert eine Transaktion ein Datenobjekt, so wird diese Änderung zunächst nur bei der Primärkopie vorgenommen und im weiteren Verlauf asynchron an alle weiteren Datenkopien weitergegeben. Das Lesen von Daten ist allerdings ohne den Zugriff auf die Primärkopie möglich. Aufgrund der asynchronen Änderungen ist es dabei aber möglich, das Leseanforderungen auf inkonsistente Daten zugreifen. Der Vorteil dieses Verfahrens liegt in der Einfachheit des Sperralgorithmus. Allerdings hat dieses Verfahren einen entscheidenden Nachteil. Ein verändernder Datenzugriff ist stark von der Verfügbarkeit der Primärkopie abhängig. Ist diese Primärkopie aufgrund eines Rechner- oder Netzausfalls nicht erreichbar, sind solange keine Änderungen möglich, wie die Primärkopie nicht verfügbar ist oder durch eine andere Kopie ersetzt wurde.

2.2.2 Voting-Verfahren

Ein verbesserter Ansatz im Gegensatz zu den Primary-Verfahren sind die Voting-Verfahren. Hier erfolgt die Synchronisation der Zugriffe durch Abstimmung. Jede Kopie erhält dazu eine bestimmte Anzahl an Stimmen (meistens 1). Der Zugriff einer Transaktion auf ein logisches Objekt wird in diesem Fall nur dann erlaubt, wenn eine entscheidungsfähiges sogenanntes Quorum an Kopien zustimmt.

Definition 2.3 Qorum

Ein Quorum ist eine entscheidungsfähige Anzahl an Stimmen, die zu einem bestimmten Datenzugriff (Lesen/Schreiben) auf die Daten eines Fragmentes angefordert werden.

Dabei wird zwischen einem Lesequorum Q_R und einem Schreibquorum Q_W unterschieden. Dateninkonsistenzen werden dann durch folgende Überschneidungsregeln vermieden :

- Schreib/Schreib-Überschneidungsregel:
  $2 * Q_W > \sum$ über alle Stimmen
- Schreib/Lese-Überschneidungsregel :
  $Q_W + Q_R > \sum$ über alle Stimmen

Es gibt zwei Hauptkriterien, nach denen man die Voting-Verfahren einteilen kann. Eine Möglichkeit ist die Struktur des Quorums. Beim unstrukturierten Ansatz sind alle Stimmen von beliebigen Kopien zur Abstimmung zugelassen. Hingegen werden die Kopien bei strukturierten Verfahren in einer logischen Struktur (Baum oder Gitter) angeordnet, so daß man zur Erlangung eines Quorums nur noch entlang dieser Struktur auf Stimmenfang gehen muß und somit weniger Stimmen als bei unstrukturierten Verfahren benötigt. Ein weiteres Einteilungskriterium ist die Größe des Quorums. Durch mögliche auftretende Partitionen ist es manchmal nötig, daß sich ein Quorum dynamisch an die gerade verfügbare Stimmenanzahl anpasst, um eine

erforderliche Stimmenmehrheit zu erreichen. Weitere in der Literatur genannte Möglichkeiten sind die Abstimmung mit sogenannten „Ghosts“ (Geistern), „Witnesses“ (Zeugen) oder „Bystanders“ (Zuschauern). Dies sind Rechner, die selber nicht Besitzer eines Replikates sind, in speziellen Fällen aber in die Abstimmung mit einbezogen werden. Eine nähere Beschreibung erfolgt in [Sar97].

2.2.3 ROWA

Ein weiteres pessimistisches Verfahren ist das sogenannte ROWA-Verfahren (Read One Write All). Bei dieser Methode müssen alle Kopien synchron aktualisiert werden (Write All). Somit können Lesezugriffe immer auf einer, wenn vorhanden, lokalen Kopie ausgeführt werden. Ein Nachteil hierbei ist, daß bei Knotenausfällen die Schreibverfügbarkeit nicht mehr gegeben ist. Daher gibt es eine Reihe von Ansätzen, die z.B. beim Schreiben nur die erreichbaren Kopien aktualisieren (Write all Available) oder bei Nichtverfügbarkeit einzelner Kopien auf ein anderes Replikationsverfahren umschalten (Missing Update). Beim Write all Available-Verfahren wird die Anzahl der verfügbaren Kopien in sogenannten replizierten Verzeichnissen (Directories) gespeichert. Vor jedem Zugriff wird das lokale Verzeichnis konsultiert, um die verfügbaren Kopien zu erhalten. Zum Lesen wird dann auf eine dieser Kopien zugegriffen. Beim Schreiben werden nur die verfügbaren Kopien aktualisiert. Der Nachteil dieses Verfahrens liegt in der sehr aufwendigen Verwaltung und Aktualisierung der replizierten Verzeichnisse. Beim Missing Update-Verfahren wird nur im fehlerfreien Normalmodus die ROWA-Methode verwendet. Bei auftretenden Fehlern schaltet das Verfahren in den Fehlermodus, in dem ein fehlertolerantes Voting Verfahren zum Einsatz kommt, um diese aufgetretenen Fehler möglichst zu beheben. Auch bei ROWA gibt es noch eine Reihe von Erweiterungsmöglichkeiten, die hier aber nicht genauer erläutert werden sollen.

Soweit ein Überblick über die Möglichkeiten der pessimistischen Replikationskontrolle. Im folgenden Abschnitt sollen nun die optimistischen Replikationsverfahren unter die Lupe genommen werden.

2.3 Optimistische Replikationsverfahren

Die zwei wichtigsten Verfahren in dieser Kategorie sind die Log-Transformation und das Data-Patch-Verfahren, welche sich beide semantischer Informationen von Transaktionen bedienen, um mögliche Inkonsistenzen zwischen einzelnen Fragmenten aufzulösen.

2.3.1 Log-Transformation

Die Log-Transformation erlaubt das uneingeschränkte Arbeiten mit Partitionen. Diese entstehen durch netzbedingte bzw. ressourcenbedingte Abkopplungen von einem oder mehreren Rechner aus einem vorhandenen Rechnernetz. Datenänderungen werden asynchron an Replikate weitergeleitet und auftretende Inkonsistenzen durch Rücksetzen und erneute Ausführung einer Transaktion beseitigt. Dafür werden bei der Log-Transformation die zur erneuten Ausführung notwendigen Logbucheinträge durch die folgenden drei Regeln minimiert :

1. Kommutative benachbarte Transaktionen können ohne Konsistenzverlust vertauscht werden.
2. Eine Transaktion kann gelöscht werden, falls ihre Änderungen komplett durch die nachfolgende Transaktion überschrieben werden.
3. Benachbarte Transaktionen können gelöscht werden, falls sie sich gegenseitig in ihrer Wirkung neutralisieren.

Die Nachteile hierbei liegen in der umfangreichen Buchführung von Änderungen und im verfahrensbedingten, ineffizienten Rücksetzen von Transaktionen.

2.3.2 Data patch

Beim Data-Patch-Verfahren wird das Rücksetzen von Transaktionen bei der Reintegration von Partitionen durch die Verwendung anwendungsbezogener Regeln vermieden, die bereits beim Datenbank-entwurf für jede Relation festgelegt worden sind. Diese Regeln sind dabei in zwei Klassen eingeteilt. Die erste Klasse, die sogenannten Objekt-Einfügungsregeln, steuert das Einfügen von Datenbankobjekten in nur einer Partition. Sie besteht aus vier Regeln :

- **Keep-Regel**
  Bewirkt das Einfügen eines Datenbankobjektes in jeder Partition
- **Remove-Regel**
  Löscht ein Datenbankobjekt, wenn es nur in einer Partition eingefügt wurde
- **Program-Regel**
  Aktiviert anwendungsspezifische Konfliktlösungsprogramme
- **Notify-Regel**
  Informiert bei nicht lösbarem Konflikt den Datenbank-administrator

Die zweite Klasse, die sogenannten Objekt-Integrationsregeln, soll die Eindeutigkeit von Datenbankobjekten mit gleicher Identifikation wahren. Sie besteht aus den folgenden fünf Regeln :

- **Latest-Regel**
  das zuletzt eingefügte Tupel mit gleicher Identifikation wird übernommen
- **Primary-Regel**
  das Tupel auf dem Rechnerknoten R ist das gültige und wird übernommen

- **Arithmetic-Regel**
  bildet unter Verwendung mathematischer Formeln aus mehreren Datenbankobjekten mit gleicher Identifikation ein neues Datenbankobjekt
- **Program-Regel**
  Aktiviert anwendungsspezifische Konfliktlösungsprogramme
- **Notify-Regel**
  Informiert bei nicht lösbarem Konflikt den Datenbank-administrator

2.4 Schwache Konsistenzmodelle

Bei den oben vorgestellten pessimistischen und optimistischen Replikationsverfahren wird die Konsistenz der Replikate auf Kosten einer geringen Verfügbarkeit erzielt. Für die Erhöhung der Verfügbarkeit ist es daher notwendig, schwächere Konsistenzmodelle zuzulassen. Im folgenden sollen zwei Modelle, die in diese Richtung gehen, vorgestellt werden.

2.4.1 ε-Serialisierbarkeit

Dieses Verfahren stellt eine Erweiterung des bereits in Definition 2.2 dargestellten Serialisierbarkeitsbegriffes dar. Durch das ε wird die tolerierbare Abweichung von Replikaten zu den letzten Änderungen festgelegt und unter Beachtung diese Toleranzbereiches die überlappte Ausführung von Transaktionen mit Lesezugriffen auf veraltete Replikate und damit eine höhere Verfügbarkeit dieser Datenbankobjekte erlaubt.

2.4.2 Quasi-Copy-Verfahren

Das Quasi-Copy-Verfahren legt die tolerierbare Inkonsistenz von Datenbankobjekten über die Anzahl lokal erlaubter Änderungen oder über die Zeitspanne zwischen Aktualisierungen fest.
Bei beiden Verfahren wird die zentrale Verwaltung von Schreibzugriffen und die Aktualisierung von Replikaten über einen dedizierten Rechner verlangt.

2.5 Zusammenfassung

Das vergangene Kapitel gibt einen groben Überblick über die verschiedenen Arten klassischer Replikationsverfahren in verteilten Systemen im Bezug auf den in Abbildung 2.1 dargestellten Zielkonflikt der Replikationskontrolle in verteilten Systemen. Dieser Konflikt wird sich im Laufe dieser Arbeit bezogen auf das mobile Umfeld noch etwas ändern. Dann wird auch noch einmal auf die Tauglichkeit dieser Verfahren für den Einsatz in der mobiler Umgebung eingegangen. In den folgenden beiden Kapiteln sollen zunächst aber die verschiedenen Einflüsse und Merkmale des mobilen Umfeldes, sprich die mobilen Kontexte, dargelegt, klassifiziert und ihre Auswirkungen auf die Replikation in verteilten mobilen Systemen herausgestellt werden.

3 Mobile Kontexte

3.1 Einführung

Im Gegensatz zu normalen VDBS gibt es bei der mobilen Datenverarbeitung eine Reihe unterschiedlichster Faktoren, die Einfluss auf eine Vielzahl von Bearbeitungsvorgängen für mobile VDBS haben. Dies wird auch in den in [Wat96] definierten Eigenschaften für mobile Umgebungen deutlich :

1. Mobilität bringt einen häufigen Umgebungswechsel für mobile Elemente hervor
2. Der mobile Nutzer besitzt ein erhöhtes situationsabhängiges Informationsbedürfnis
3. Mobile Geräte besitzen gegenüber stationären Geräten eingeschränkte Ressourcen und daraus resultierend häufig eingeschränkte Funktionalität
4. Der mobile Nutzer verwendet häufig unterschiedliche Netzstrukturen, die stark abweichende Eigenschaften besitzen und deshalb zur häufigen Verbindungsunterbrechung und zum schnellen Medienwechsel führen können
5. Mobile Elemente sind erhöhten Sicherheitsrisiken ausgesetzt

Aus diesen Eigenschaften resultieren eine Menge von Einflußfaktoren der mobilen Umgebung, die man unter dem Begriff des mobilen Kontextes zusammenfaßt. In den nächsten Abschnitten soll zum besseren Verständnis und für weitere Ausführungen der Begriff des Kontextes nun noch einmal genauer definiert , typische Eigenschaften genannt und eine spezielle Klassifikation für den mobilen Kontext herausgestellt werden. Dabei orientiere ich mich an den Ausführungen in [Wat96], der sich bereits eingehender mit diesem Thema beschäftigt hat.

3.2 Definition

In diesem Abschnitt soll eine Definition für den Begriff Kontext speziell für den Bereich der Datenbanken gegeben werden. Dazu ist es zunächst einmal notwendig, einige grundlegende Definitionen voranzustellen.

Definition 3.1 Datenbank

Eine Menge von persistenten auf einem Datenträger abgelegten Datenobjekten heißt Datenbank.

Definition 3.2 Datenbankmodell

Ein Datenbankmodell ist ein System von Konzepten zur Beschreibung von Datenbanken. Es legt somit Syntax und Semantik von Datenbankbeschreibungen für ein Datenbanksystem fest, den sogenannten Datenbankschemata. [HeSa95]

Definition 3.3 DBS

Unter dem Begriff Datenbank-Managment-System, kurz DBMS verstehen wir die Gesamtheit der Software-Module, die die Verwaltung einer Datenbank übernehmen. Ein Datenbanksystem, kurz DBS, ist die Kombination eines DBMS mit einer Datenbank. [HeSa95]

Somit ergibt sich für einen Datenbank-Kontext folgende Definition :

Definition 3.4 DBS-Kontext

Ein Kontext heißt DBS-Kontext, kurz K^{DBS}, wenn er einen Einfluß auf ein gegebenes DBS-Modell hat. [Wat96]

3.3 Eigenschaften

Um auf verschiedene Kontexte reagieren zu können, benötigt man Informationen über sie. Diesen Informationen kann man verschiedene Eigenschaften zuweisen, die in diesem Abschnitt näher erläutert werden sollen.

Aktualität.

Dies ist ein sehr wichtiges Kriterium. Wenn nämlich veraltete Informationen benutzt werden, kann daraus ein falscher Kontext abgeleitet werden und es erfolgt meistens eine falsche Anpassungsreaktion des Systems an den jeweiligen Kontext.

Genauigkeit.

Genauigkeit ist ebenfalls ein sehr wichtiges Kriterium. Ungenaue Angaben bei der Kontextbeschreibung könne ebenso wie nicht aktuelle Informationen zu einer falschen oder unsinnigen Anpassungsreaktion durch das System führen.

Herkunft.

Aus der Herkunft der Kontextinformationen kann man andere wichtige Informationen gewinnen, wie Genauigkeit, Aktualität und Zweck der Information. Dazu ist es notwendig, daß das System weiß, wie es von der Herkunft auf andere Eigenschaften schließen kann.

Überprüfbarkeit.

Aus Gründen der Sicherheit und aufgrund unerwarteter Reaktionen ist es nötig, die verwendeten Kontextinformationen überprüfbar zu machen.

3.4 Klassifikation

Im nun folgenden Abschnitt werden einige Klassifikationen für Kontexte vorgestellt, die diese nach den verschiedensten Kriterien

unterteilen, bevor dann im nächsten Abschnitt eine Kontextmodell für mobile Systeme genauer definiert wird.

1. **Klassifikation nach dem Zeitpunkt der Einbeziehung**

   Hier werden Kontexte nach dem Zeitpunkt, zu dem sie vom System zur Anpassung verwendet werden, unterteilt.

   - vor Systemnutzung
   - während Systemnutzung :
     - kontinuierlich
     - vordefinierter Zeitpunkt
     - vor bzw. nach vordefinierten Aktionen
     - bei Nutzeranforderung
     - besondere Situationen
   - zwischen Sessions

2. **Klassifikation nach der Häufigkeit der Änderung**

   Ändert sich der Kontext, in dem eine Datenbank benutzt wird, nicht, so ist dieser statisch. Bei einer Änderung hingegen handelt es sich um einen dynamischen Kontext.

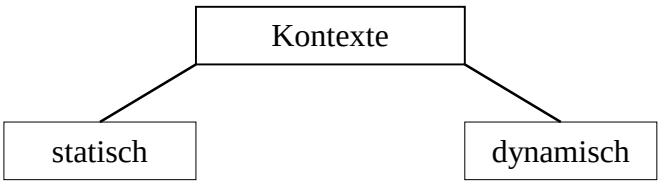

Abbildung 3.1 Klassifikation nach Häufigkeit der Kontextänderung

Die ersten beiden genannten Klassifikationen kann man zueinander in Beziehung setzen. Die sich daraus ergebenden Zusammenhänge verdeutlicht Tabelle 3.1.

3. **Klassifikation nach Beschreibbarkeit**

   - „reicher Kontext“ : kann nicht vollständig beschrieben werden
   - „armer Kontext“ : kann vollständig beschrieben werden

Zeitpunkt der Einbeziehung	Änderung des Kontextes
vor der Systemnutzung	statisch
während Systemnutzung :	
kontinuierlich	stark dynamisch
vordefinierter Zeitpunkt	dynamisch
vor/nach definierten Aktionen	statisch, dynamisch
bei Nutzeranforderung	statisch, dynamisch
besondere Situationen	statisch, dynamisch
zwischen Sessions	wenig dynamisch, statisch

Tabelle 3.1 Zusammenhang – Zeitpunkt der Einbeziehung und Änderungshäufigkeit

4. Klassifikation nach Art und Weise der Beeinflussung

Hier werden Kontexte nach der Art des Einflusses auf die Semantik von Informationen klassifiziert. Dabei unterscheidet man im Wesentlichen zwischen 4 Kategorien :

- Nutzerspezifika
  Vorlieben, Wünsche und Fähigkeiten des Nutzers
- Umgebung
  lokale, historische, kulturelle und temporale Gegebenheiten, in denen sich der mobile Nutzer befindet
- Anwendung
  Priorität von Informationen, Speicher- und Verarbeitungsort, Gültigkeitsdauer von Informationen sowie weitere Kriterien zur verlustlosen Informationskomprimierung
- Ressourcen
  genutzte mobile Geräte und genutzte Netzwerkverbindungen

3.5 Kontextklassifikation für mobile DB-Umgebung

In diesem Abschnitt wird nun eine Klassifikation vorgestellt, die Kontexte, welche die Arbeitsweise eines DBS in mobiler Datenbank-umgebung beeinflussen, klassifiziert. Im MoVi- Projekt erfolgt die Betrachtung des Mobilitätsbegriffes im Bezug auf den Grad der zeitlich abhängigen Ortsveränderung der 3 Kategorien Mensch, Gerät und Information. Diese drei Elemente stellen daher die wesentlichen Kontext-Kategorien dar und sind deshalb Grundlage für die folgende Klassifikation, die in Abbildung 3.2 dargestellt wird.

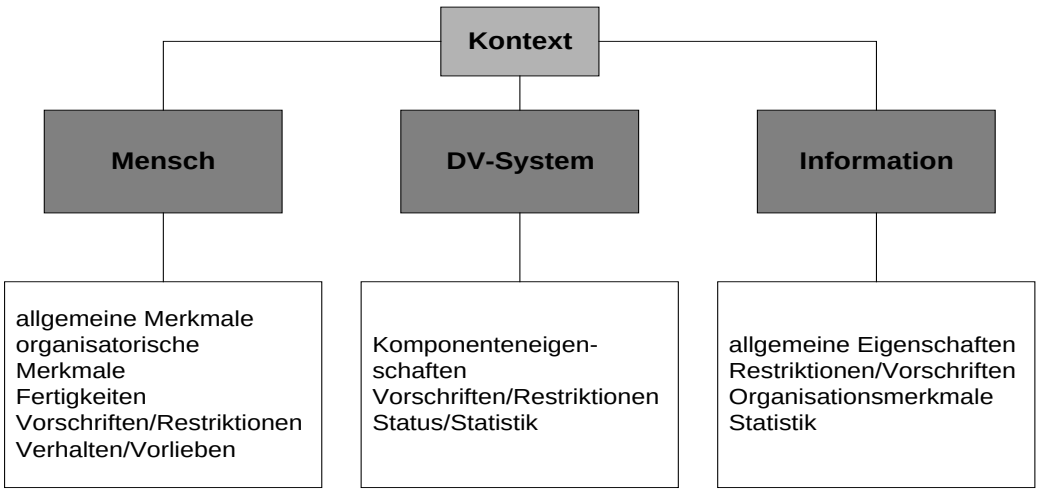

Abbildung 3.2 Kontextklassifikation für das mobile Umfeld

Als nächstes werden die drei einzelnen Kategorien noch einmal genauer betrachtet und ihre wesentlichen Inhalte beschrieben.

Mensch

- allgemeine Merkmale

  Diese Merkmale beinhalten Eigenschaften, wie Name, Vorname, Alter usw., aber auch den derzeitigen Aufenthaltsort und die sich daraus ergebende soziale Situation.

- organisatorische Merkmale

  Dies sind Eigenschaften, die sich aus der Stellung des Menschen in einer bestimmten Organisation ergeben, wie Stellung in der Organisation, zu erfüllende Aufgaben, Ort der Aufgabenerfüllung und notwendige Werkzeuge und Maß an notwendigen Informationen zur Aufgabenerfüllung.
- Fertigkeiten

  Diese Kategorie beinhaltet bestimmte Fähigkeiten des Anwenders, wie Computerkenntnisse, Systemkenntnisse, Anwendungskenntnisse, Erfahrungen usw.. Leider sind diese Größen nur schwer zu quantifizieren und somit schwierig als Kontexte zu nutzen.
- Vorschriften/Restriktionen

  Jeder Nutzer besitzt bestimmte Rechte und Befugnisse, welche sich in der mobilen Umgebung sowohl aus der organisatorischen Stellung ergeben, aber auch durch technische Gegebenheiten der anwendbaren Geräte und Kommunikationsstrukturen, die orts- und zeitbedingt sein können, hervorgerufen werden.
- Verhalten/Vorlieben

  Dies sind z.B. die Art und Weise, wie bestimmte Aufgaben vom Nutzer erledigt werden, der Zeitpunkt und der Ort der Erledigung, ein bestimmtes Informationsbedürfnis und die Nutzung bestimmter Werkzeuge. Diese Eigenschaften werden häufig durch Vorschriften und Restriktionen eingeschränkt.

Information

- allgemeine Eigenschaften

  Informationen besitzen eine bestimmte Aktualität, Qualität, Quantität, benutztes Medium usw..

- Restriktionen/Vorschriften

  Es gibt häufig Situationen, bei denen Informationen nur an bestimmten Orten zur Verfügung stehen oder nur bestimmten Personen zugänglich sein dürfen.
- Organisationsmerkmale

  Diese Kategorie umfaßt Fakten, die sich aus der Verwendung einer Information ergeben, wie Ort und Zeit, zu der sie zur Verfügung stehen muß oder solche organisatorischen Merkmale, wie die Einteilung nach Stamm- und Bewegungsdaten.
- Statistik (Zugriffshäufigkeit, Zugriffswahrscheinlichkeit usw.)

DV-System

- Komponenteneigenschaften

  Diese umfassen Informationen, die durch Konfiguration des Systems und die konstruktiven Merkmale vorgegeben sind, wie Geräte-eigenschaften und Ressourcen, Netzeigenschaften usw..
- Vorschriften/Restriktionen

  Sie können aus organisatorischen Vorschriften, die die Kosten und die Sicherheit betreffen, aber auch aus Geräteeigenschaften entstehen.
- Status/Statistik

  In dieser Kategorie werden Fakten zusammengefaßt, die sich aus dem Betrieb bestimmter Komponenten ergeben. Beispiele für den Status sind die aktuelle Netzbelastung, Übertragungs-geschwindigkeit, Speicherbelegung usw., ein Beispiel für die Statistik ist, welche Kommunikationskanäle durch welchen Nutzer benutzt wurden.

Bezugnehmend auf die hier dargestellte Klassifikation wird nun im weiteren Verlauf der Arbeit geklärt, welchen Einfluß mobile Kontexte auf Replikationsverfahren und im Speziellen auf Replikationsziele

haben. Im nächsten Kapitel wird aber erst einmal das für die Testimplementation vorgesehene Anwendungsszenario der Literaturdatenbanken vorgestellt.

4 Das Anwendungsszenario : Literaturdatenbanken

Das hier beschriebene Anwendungsszenario ist der Ausgangspunkt für weitere Diskussionen innerhalb dieser Arbeit und wird in der Testimplementation zu dieser Arbeit als Anwendungsbeispiel dienen.

4.1 Einführung

In [Web98] wurde ein Datenbanksystem erstellt, mit dem über das World Wide Web auf eine Literaturdatenbank zugegriffen werden kann. Diese Literaturdatenbank verwaltet dabei Einträge von drei sehr gebräuchlichen, bereits bestehenden Literaturdatenbanken, nämlich :

- $BIBTeX$-Dateien
- Refdbms
- NCSTRL(Networked Computer Science Technical Reports Library)

NCSTRL ist dabei nur für die Verwaltung technischer Bericht gedacht, während in $BIBTeX$-Dateien und Refdbms die einzelnen Einträge bestimmten Klassen wie Artikel oder Buch zugeordnet werden, wobei die Zuordnung dem jeweiligen Anwender überlassen bleibt. Einen Überblick über die verschiedenen möglichen Eintragstypen gibt Tabelle 4.1 Das Ergebnis der Arbeit ist ein Datenbanksystem, das folgende Möglichkeiten für den Anwender bietet :

- Import von Einträgen in den einzelnen Formaten
- Änderung und Reklassifizierung der Einträge
- Anfragen an die Datenbank, im Einzelnen nach :
    - Themen
    - Autoren
    - Einträgen von Erfassern und

- einzelnen Attributen

- Export von Einträgen in die einzelnen Formate

Der zuletzt genannte Punkt des Exportes kann nach nutzerdefinierten Anfragen erfolgen und wurde von mir zur Überführung der Datenbank in Oracle genutzt.

$BIBT_EX$	**refbbms**	**NCSTRL**
Article	Article	
Book	Book	
Booklet		
Conference		
Inbook	InBook	
Incollection		
Inproceedings	InProceedings	
Manual	Manual	
Masterthesis		
Misc	Miscellanous	
Phdthesis	PhDthesis	
Proceedings	Proccedings	
Techreport	TechReport	alle Einträge
Unpublished	UnPublished	
weitere Eintragstypen, die nicht standardmäßig sind		

Tabelle 4.1 Vergleich der Eintragstypen [Web98]

Das erstellte Datenbanksystem basiert auf O_2 und für die Präsentation im World Wide Web wurde das Modul O_2Web genutzt.

4.2 Überführung in ein Relationales Modell

In dieser Arbeit geht es darum, ein spezielles Replikationsverfahren auf ein bestimmtes mobiles Anwendungsszenario zuzuschneiden, und eine Testimplementation dazu zu entwickeln. Als Beispielanwendung dient in diesem Fall die oben beschriebene Literaturdatenbank. Derzeit gibt es aber noch keine Anbindung mobiler Geräte an ein objektorientiertes DBS wie O_2. Deshalb mußte diese Datenbank erst einmal in ein relationales Datenbanksystem überführt werden, in unserem Fall in

Oracle 8.0. Da es sich aber nur um eine Implementation zu Testzwecken handelt, ist es ausreichend, nur einen Teil der Datenbank in Oracle zu überführen. Dies geschah mit der schon oben erwähnten Möglichkeit des Exports nutzerdefinierter Anfrageergebnisse in das $BIBTeX$-Format. Aufgrund auftretender Schwierigkeiten und in Absprache mit meinem Betreuer wurden die drei Eintragstypen Inproceedings, Masterthesis und Techreport überführt. Dabei gibt es für jeden Eintragtypen Felder, die einer zwingenden Angabe bedürfen und sogenannte optionale Felder, die weitere Angaben zu dem jeweiligen Eintragstypen enthalten. Im folgenden sollen die drei genannten Eintragstypen zusammen mit den zwingenden und optionalen Feldern vorgestellt werden :

inproccedings Artikel in einem Konferenzband
- zwingend : author, title, school, year
- optional : editor, volume oder number, series, pages, address, month, organization, publisher, note

masterthesis Diplomarbeit
- zwingend : author, title, school, year
- optional : type, address, month, note

techreport Bericht, der von einer Hochschule oder Institution veröffentlicht wurde, normalerweise numerierte Ausgabe in einer Reihe
- zwingend : author, title, institution, year
- optional : type, number, address, month, note

Bei der Überführung in Oracle habe ich mich auf die zwingenden Eintragsfelder beschränkt. Das Feld *author* enthält Angaben über den oder die Autoren. Namen können dabei in verschiedenen Formen angegeben werden. Sie können aus bis zu vier Teilen bestehen (Vorname, von, Nachname, Jr). Ein Eintrag kann mehrere Autoren haben. Ich habe mich bei meiner Überführung auf drei Autoren beschränkt. Hat ein Eintrag mehr als einen Autor, wird ein Zusatzfeld

and_others mit dem boolschen Wert true belegt. Einige der Einträge enthalten einen Verweis auf andere. Dies wird mit dem Feld *crossref* realisiert. Enthält ein Eintrag dieses Feld, übernimmt er nicht spezifizierte Felder aus dem Eintrag, auf den er verweist. Da ich nicht alle Daten in Oracle überführe, kann es passieren, daß auf einen Eintrag verwiesen wird, der nicht überführt wurde. Daher gebe ich den für die verschiedenen Einträge festgelegten Schlüssel unter dem Feld *crossref* an. Dieser ist wie folgt aufgebaut :

Bei BIBTEX-Dateien und refdbms-Einträgen existiert für jeden Eintrag ein Schlüsselwort, welches innerhalb dieser Literaturdatenbanken eindeutig ist und der Identifizierung dient. Es kann aber passieren, daß sich Einträge nur über den Erfasser unterscheiden. Der Erfasser oder Submitter ist hier diejenige Person, die den Eintrag in die Literaturdatenbank vorgenommen hat. Ein Schlüssel für einen Eintrag setzt sich somit aus dem Schlüsselwort des Eintrages und dem Login des jeweiligen Erfassers zusammen.

Die in [Web98] entwickelte Entry-Klasse für jeden einzelnen Eintrag setze ich hier nicht in eine eigene Relation um, womit sich für die Überführung der Literaturdatenbanken 4 Relationen mit ihren einzelnen Attributen ergeben:

- **Personen** Namenseinträge der verschiedenen Personen (hier nur die Autoren)
  Schlüssel : Personen_ID
  weitere Attribute : Vorname, Von_Teil, Nachname, Jr_Teil
- **Inproceedings** Einträge für diesen Eintragstyp
  Schlüssel : Keyword und Submitter_Log
  weitere Attribute : author, co_author1, co_author2, and_others, title, booktitle, year, crossref

- **Masterthesis** Einträge für diesen Eintragstyp
  Schlüssel : Keyword und Submitter_Log
  weitere Attribute : author, co_author1, co_author2, and_others, title, institution, year,
- **Techreport** Einträge für diesen Eintragstyp
  Schlüssel : Keyword und Submitter_Log
  weitere Attribute : author, co_author1, co_author2, and_others, title, institution, year,

Zur Vereinfachung habe ich die Attribute School bei Masterthesis und Institution bei Techreport gleichgesetzt.

4.3 Eigenschaften von Literaturdaten

In diesem Abschnitt will ich noch ein paar Anmerkungen zu den Eigenschaften von Literaturdaten machen Ist ein Literatureintrag erst einmal erfolgt, so ist davon auszugehen, daß er im weiteren Verlauf seines Bestehens keinen großen Veränderungen mehr unterzogen wird. Wenn überhaupt, so wird er höchstens gelöscht. Man kann somit also sagen, daß die Werte innerhalb dieser Literaturdatenbank als sehr konstant und immer aktuell anzusehen sind. Eine Änderung des Datenbestandes erfolgt nur durch einen Neueintrag. Diese Eigenschaft wird bei der späteren Modifikation des Replikationsverfahrens noch wichtige Rolle spielen.

5 Einfluß von Kontexten auf die Replikation

Mobile Kontexte sind, bezieht man sich auf die in Kapitel 3 vorgestellten Klassifikationen, dynamisch und „reich“. Sie beschreiben, wie bereits erwähnt die verschiedensten Einflüsse mobiler Umgebungen auf die Vorgänge der Datenverarbeitung. Laut [HeLu99] ist die Replikation von Daten dabei eines der Kernprobleme, die es zu bewältigen gibt. Im nun folgenden Kapitel werden die Einflüsse der verschiedenen Kontexte auf die Datenreplikation, speziell auf die sich daraus ergebenden Ziele geschildert. Dabei orientiere ich mich an der in Abschnitt 3.5 vorgestellten Kontextklassifikation für mobile Datenbankumgebungen. Desweiteren werden, falls notwendig, veränderte Ziele für die mobile Datenreplikation genannt, die in Kapitel 2 vorgestellten Verfahren hinsichtlich dieser Ziele kurz bewertet und Schlußfolgerungen aus dieser Bewertung gezogen. Tabelle 5.1 faßt die nachfolgenden Ausführungen dann noch einmal zusammen.

5.1 Der Mensch

Ein Mensch besitzt, bezogen auf die mobile Umgebung, allgemeine und organisatorische Merkmale sowie bestimmte Fertigkeiten, unterliegt Vorschriften und Restriktionen und hat ein bestimmtes Verhalten bzw. Vorlieben. Für den Grad der Mobilität spielt der Aufenthaltsort des Menschen eine große Rolle. Ein Mensch kann auch mobil sein, ohne ein mobiles Gerät bzw. eine mobile Verbindungseinrichtung zu nutzen. Er ist dann aber teilweise von den vor Ort vorhandenen Möglichkeiten der Rechnernutzung bzw. der vorhandenen Kommunikationsstruktur abhängig. Erfolgt z.B. eine Datenabfrage über einen Festnetzrechner und eine stabile Kommunikationsverbindung, so ist es sicherlich nicht wichtig, die Verfügbarkeit der Daten zu erhöhen oder die

Kommunikationskosten niedrig zu halten, sondern die Wichtung der Replikationsziele verschiebt sich fast gänzlich in die Richtung der Konsistenzerhaltung. An einem anderen Ort nutzt der Mensch nun aber ein mobiles Gerät und eine mobile Verbindungseinrichtung. Um dann ein ordnungsgemäßes Arbeiten auf dem Gerät während plötzlicher oder gewollter Verbindungsunterbrechungen zu gewährleisten, erlangen Replikationsziele, wie die Verfügbarkeit der Daten und die Minimierung der Kommunikationskosten eine höhere Wichtung als die Konsistenz und die Tolerierung lokaler Inkonsistenzen auf dem mobilen Gerät wird fast zwingend erforderlich.

Auch die organisatorische Stellung des Menschen kann Einfluß auf die Replikation haben. So kann es möglich sein, daß einem Menschen aufgrund seiner Position in einer bestimmten Organisation der Zugriff auf bestimmte Daten erlaubt bzw. nicht gestattet ist, was sich dann auf die Verfügbarkeit der Daten auswirkt. Ein weiteres charakteristisches Merkmal mit Einfluss auf die Wichtung der Replikationsziele ist das Informationsbedürfnis des Menschen. Abhängig von der zu erfüllenden Aufgabe benötigt der Nutzer bspw. Daten mit einem hohen Grad an Aktualität. Daraus ergibt sich als Hauptziel der Replikation die Wahrung der Datenkonsistenz. Ist der Aktualitätsgrad der Daten hingegen nicht sehr gefragt, kann man zugunsten einer besseren Verfügbarkeit die Konsistenz etwas vernachlässigen.

Der Ort der Aufgabenerfüllung kann, ähnlich zum Aufenthaltsort, aufgrund der möglichen und unterschiedlichen verwendeten Geräte und Verbindungen Einfluss auf die Wichtung der jeweiligen Replikationsziele haben.

Durch das Wissen über bestimmte Fertigkeiten und Kenntnisse eines Nutzers kann man Einfluß auf die Verwirklichung bestimmter Replikationsziele nehmen. Dies kann z.B. geschehen, wenn man ihm die Entscheidung darüber überläßt, wann oder ab welcher Größe eine veränderte Datenmenge zur Datensynchronisation bzw. zum -abgleich weitergeleitet wird. Dadurch könnte man direkten Einfluss auf die

Gestaltung der Kommunikationskosten nehmen. Diese Kenntnisse und Fertigkeiten sind für einen einzelnen Nutzer allerdings schwer zu quantifizieren.

5.2 Information

Wichtige Eigenschaften für Informationen im mobilen Umfeld sind ihre Aktualität und Quantität, welchen Vorschriften und Restriktionen sie unterliegen sowie organisatorische Merkmale und statistische Aussagen über die Daten. Aus dem Aktualitätsgrad und der Quantität kann man gegebenenfalls auf das geforderte Replikationsziel schließen. Haben Daten z.B. eine hohe Änderungsfrequenz, ist es sicherlich sehr wichtig, großen Wert auf die Konsistenz zu legen. Werden Daten allerdings sehr selten bis gar nicht geändert, kann man die Konsistenz zugunsten der Verfügbarkeit einschränken. Auch die Beschaffenheit der Daten kann sich auf das angestrebte Replikationsziel auswirken. Bilddaten z.B. können eine erhebliche Größe besitzen und so beschaffen sein, daß sie nicht überall verfügbar gemacht werden können.
Wie bereits in Abschnitt 5.1. erwähnt, können Vorschriften und Restriktionen dazu führen, daß Daten nur an bestimmten Orten bzw. bestimmten Personen zugänglich sein dürfen, was sich direkt auf die Verfügbarkeit auswirkt.
Informationen über den Ort und die Zeit, zu denen bestimmte Daten zur Verfügung stehen müssen, wirken sich ebenfalls auf das angestrebte Replikationsziel aus. Werden die Daten von einem stationären Rechner über eine sichere und stabile Verbindung angefordert, ist die Sicherung der Konsistenz als Ziel mit der höchsten Wichtung zu betrachten. Erfolgt die Abfrage allerdings von einem mobilen Gerät über eine unsichere Funknetzverbindung, verschiebt sich diese Wichtung in Richtung der Verfügbarkeit und der Kostenminimierung.

Aufgrund statistischer Informationen, wie Zugriffshäufigkeit und Zugriffswahrscheinlichkeit, kann man über die Zuweisung von Replikaten an Rechner, bei denen diese Werte als ausreichend angesehen werden, die Verfügbarkeit von Daten erhöhen und somit gleichzeitig, falls eine Funkverbindung benutzt wird, Kosten für eventuelle teure Anfragen auf diese Informationen sparen.

5.3 DV-System

Ein DV-System wird charakterisiert über die Eigenschaften seiner Komponenten, wie Geräteeigenschaften, Ressourcen und Netzeigenschaften, über organisatorischen Vorschriften, welche die Kosten und die Sicherheit betreffen sowie über Status- und Statistikinformationen, die sich aus dem Betrieb ergeben. Dabei gilt : Je besser und ressourcenreicher ein Gerät und je stabiler und sicherer die Verbindung, desto größeres Gewicht erlangt die Wahrung der Konsistenz.
Sind aus einem bestimmten Grund z.B. die Kommunikationskosten beschränkt, ist es wichtig, die lokale Verfügbarkeit von Daten zu erhöhen, um unnötige Kommunikation zu vermeiden. Ist desweiteren aufgrund von Ressourcenmangel, z.B. Energieressourcen, ein Aufrechterhalten der Verbindung nicht mehr möglich, ist es notwendig, die benötigten Daten lokal verfügbar zu machen, auch, wenn dadurch die Konsistenz der Daten nicht gewahrt werden kann.
Über Statusinformationen, wie Netzbelastung und Übertragungsgeschwindigkeit können Entscheidungen darüber getroffen werden, ob eine Datenübertragung vom mobilen Gerät ins Festnetz z.B. noch verzögert oder eventuell, im Falle einer abzusehenden Verbindungsunterbrechung, vorgezogen werden sollte.
Zusammenfassend kann man also sagen, daß mobile Kontexte einen entscheidenden Einfluss auf die Art und Weise der Replikation haben. Ständig mögliche Wechsel dieser Kontexte haben eine fortwährende

Verschiebung der Replikationsziele zu Folge. Dabei kann es aber auch zu Widersprüchen zwischen den Ansprüchen der einzelnen Kontexte an das Replikationsziel kommen. So kann z.B. ein Nutzer zur Erfüllung seiner Aufgaben einen hohen Konsistenzgrad seiner Daten fordern, da er für seine Arbeit über ein mobiles Gerät und eine teure Funkverbindung nutzt, ist dies aufgrund von geforderten Kosten-einsparungen für das System oder durch mögliche Verbindungs-unterbrechungen nicht möglich. In Tabelle 5.1 werden die verschiedenen Kontextkategorien, ihre Eigenschaften und auf welche Replikationsziele sie sich auswirken noch einmal überblicksartig dargestellt. Stehen unter einer Spalte mehrere Ziele, so kann es aufgrund von Kontextwechseln zur Verschiebung des Zieles kommen. Im nächsten Abschnitt werden dann die mobilen Replikationsziele im Vergleich zu denen in Abbildung 2.1. noch einmal genauer präzisiert.

Kategorie	**Eigenschaften**	**Replikationsziele**
Mensch	Aufenthaltsort	Verfügbarkeit Konsistenz Kosten
	organisatorische Stellung	Verfügbarkeit
	Vorschriften/Restriktionen	Verfügbarkeit Konsistenz Kosten
	Informationsbedürfnis	Verfügbarkeit Konsistenz
Information	Aktualität	Verfügbarkeit Konsistenz
	Quantität	Verfügbarkeit
	Restriktionen/Vorschriften	Verfügbarkeit
	Organisationsmerkmale	Verfügbarkeit Konsistenz Kosten
	Zugriffsstatistik	Verfügbarkeit

DV-System	Komponenteneigenschaften	Verfügbarkeit Konsistenz Kosten
	Vorschriften/Restriktionen	Verfügbarkeit Konsistenz Kosten
	Status/Statistik	Verfügbarkeit Kosten

Tabelle 5.1 Kontexte und Replikationsziele

5.4 Replikationsziele in mobilen Umgebungen

Replikation in mobilen Umgebungen hat den Vorteil, daß aufgrund des Vorhandenseins von lokalen Replikaten auf einem mobilen Rechner, die Kommunikationskosten für Anfragen gering gehalten werden und die Effizienz des lokalen Arbeitens auch im Laufe von Verbindungsunterbrechungen hochgehalten wird. Aufgrund der Nutzung von instabilen und teuren Funkverbindung zur Kommunikation sind die Kosten für das Propagieren von Datenänderungen im Gegensatz zu den Anfragen sehr hoch und durch die Möglichkeit zur Durchführung lokaler Transaktionen auf verbindungslosen Replikaten besteht die Gefahr auftretender Inkonsistenzen auf den Daten. Somit ergibt sich, im Gegensatz zu den Zielen der Replikation in normalen verteilten Umgebungen (Vergleiche Abschnitt 2.1), für die Ziele in mobilen Umgebungen eine kleine Abänderung. Die Minimierung des Aufwandes einer Änderungsoperation wird in mobilen Umgebungen auf die Minimierung der Kommunikationskosten beschränkt. Dementsprechend werden die drei Ziele für die Replikation in mobilen Umgebungen hier noch einmal kurz genannt (nach [Ber98]) :

1. Erhaltung der Konsistenz der Replikate auch bei stark schwankenden und relativ häufig ausfallenden/ausgeschalteten Verbindungseinrichtungen.

2. Erhöhung der Verfügbarkeit der Replikate insoweit, daß auch bei schlechten, langsamen oder ausfallenden Verbindungseinrichtungen ein Arbeiten sowohl auf dem mobilen Rechner als auch auf Rechnern im Festnetz gewährleistet wird.
3. Verringerung der Belastung der Verbindungseinrichtungen in Bezug auf Dauer, Häufigkeit und Bandbreite, und somit auf Minimierung der Kommunikationskosten.

Im folgenden Abschnitt werden die unter Kapitel 2 genannten Replikationsverfahren im Bezug auf die Verwirklichung der eben genannten drei Ziele untersucht und Lösungen für die dadurch auftretenden Probleme genannt.

5.5 Bewertung klassischer Replikationsverfahren bezüglich mobiler Replikationsziele

5.5.1 Bewertung

Laut [Ber98] existiert kein Replikationsverfahren, das alle drei Ziele für die Replikation in mobilen Umgebungen gleichzeitig verwirklicht. Pessimistische Verfahren müssen Abstriche bei der Verfügbarkeit, optimistische bei der Konsistenz machen. Durch eine stärkere Auslastung der Funknetze hinsichtlich der Dauer und Häufigkeit der Verbindung wird zwar ein Einsatz speziell pessimistischer Verfahren möglich, die Fehleranfälligkeit des Systems erhöht sich aber dadurch.

Aus diesen Ausführungen kann man somit schlußfolgern, daß weder pessimistische noch optimistische Verfahren in ihrer reinen Form zum Einsatz in mobilen Szenarien geeignet sind. Pessimistische Verfahren können nur in Verbindung mit externem Wissen über die Arbeitsweise des Nutzers zum Einsatz kommen, so daß dieser in seiner Arbeit nur geringfügig eingeschränkt wird. Optimistische Verfahren benötigen zum Einsatz ausreichende semantische Informationen über die auszuführenden Transaktionen, um Konflikte bei der Reintegration

möglichst gering zu halten. Demnach scheint kein Verfahren zu existieren, das ohne externes Wissen die Erfüllung der drei Ziele für mobile Replikation gewährleisten kann.

5.5.2 Lösungsansatz

Ein Lösungsansatz für die im vorherigen Abschnitt genannten Probleme stellt laut [Ber98] ein hybrides Verfahren dar, das mit ausreichend externem Wissen ausgestattet ist und sehr stark an die zu bearbeitende Aufgabe angepaßt ist. Das sogenannte externe Wissen besteht dabei aus zwei Komponenten :

1. Wissen aus dem mobilen Szenario

   Dieses Wissen zeichnet sich dadurch aus, daß es relativ konstant und gering veränderlich ist, da die Annahmen, unter denen es gewonnen wurde, für den allgemeinen Problemkreis der mobilen Arbeitsweise gelten, aber durch die Festlegung eines Szenarios genug Hintergrundinformationen liefern. Über eine Schnittstelle zur Erfassung externem Wissen werden somit Daten vom

   - Betriebssystem
   - vom Anwendungsprogramm und
   - vom Anwender gewonnen

2. Problemspezifisches Wissen

   Dies kann nur aus der konkreten Einsatzsituation des mobilen verteilten DBS gewonnen werden. Der Einsatz dieses Wissens legt ein Replikationsverfahren dann auf ein einziges Einsatzgebiet fest. Ändert sich dieses, so muß sich auch das Replikationsverfahren ändern.

Ohne dieses Wissen sollte auf den Einsatz von Replikationsverfahren in mobilen Umgebungen verzichtet werden, da ansonsten die Erfüllung der geforderten Replikationsziele nicht voll gewährleistet werden kann.

5.6 Zusammenfassung

Im vergangenen Kapitel wurden die Auswirkungen mobiler Kontexte auf die Ziele für den Einsatz von Replikationsverfahren dargestellt. Dabei stellte sich heraus, das aufgrund der starken Dynamik der Kontexte häufige Wechsel in der Zielstellung innerhalb einer einzigen Anwendung keine Seltenheit sind und somit ein Verfahren möglichst alle 3 Ziele verwirklichen sollte. Nachdem die Ziele der mobilen Replikation noch einmal kurz genannt wurden, erfolgte diesbezüglich eine Bewertung der in Kapitel 2 vorgestellten Verfahren für normale verteilte Umgebungen. Dabei stellt sich heraus, daß keines dieser Verfahren allein für den Einsatz in mobilen Umgebungen geeignet scheint. Als Lösung wurde dann die Anreicherung eines hybriden Verfahrens mit externem Wissen über das mobile Umfeld und die konkrete Anwendung präsentiert, ohne daß ein Einsatz von Replikation in mobilen Umgebungen sinnlos erscheint. Unter diesem Gesichtspunkt werden in dem nun folgenden Kapitel einige neuartige, speziell für den Einsatz in mobilen Umgebungen vorgesehene Verfahren aus der Literatur vorgestellt.

6 Replikationsmodelle für das mobile Umfeld

In dem nun folgenden Kapitel werden einige erweiterte Vorschläge für die Replikationsverwaltung in mobilen Umgebungen vorgestellt. Dabei handelt es sich u.a. um eine Erweiterung der Primary-Methode [FaZa95, FaZaSri95], ein Verfahren, das sich speziell mit der Minimierung der Kommunikationskosten und der Einsparung mobiler Rechnerressourcen beschäftigt [Schrö97] sowie ein Modell mit räumlich abgegrenzten unterschiedlichen Konsistenzgraden zwischen sogenannnten Clustern. Im Anschluß an die jeweilige Vorstellung der einzelnen Verfahren erfolgt immer eine Bewertung bezüglich der Erfüllung der Replikationsziele in mobilen Umgebungen, so daß an Ende dieses Kapitels die Auswahl eines zur Umsetzung im Literatur-datenbankenszenario geeigneten Verfahrens vorgenommen werden kann.

6.1 Virtual-Primary-Copy

6.1.1 Vorstellung

Bei diesem Verfahren handelt es sich um eine Erweiterung des Primary-Copy-Verfahrens unter Berücksichtigung der Einflüsse der Mobilität auf die Replikation. Dabei wird von einer Netzstruktur ausgegangen, wie sie in Abbildung 6.1 zu sehen ist. Hierbei gibt es Funkzellen, die eine Sende- und Empfangseinheit besitzen. Diese sogenannten HBN´s stellen die Verbindung zu den mobilen Endgeräten (MH´s) her und sind im Festnetz mit weiteren stationären Rechnern (z.B. Servern) verbunden.

Die VPC-Methode geht davon aus, daß die Primärkopie eines Replikates auf einem Mobilen Gerät (MH) liegt. Wie wir bereits wissen, kann diese Primärkopie aufgrund von Ressourcenmangel oder

Netzwerkschwankungen die Verbindung zu seiner HBN abbrechen. In diesem Fall wäre ein Arbeiten auf dem Fragment des betroffenen Replikates nur noch lokal auf dem Primary möglich. Dies wird umgangen, indem die mit dem MH verbundene HBN eine Kopie der Primärkopie hält. Wird die Primärkopie vom Netz getrennt, übernimmt somit die ″virtuelle″ Kopie auf der HBN die Aufgaben des Primary. Bei erneutem Verbindungsaufbau erfolgt dann ein Abgleich der beiden Replikate.

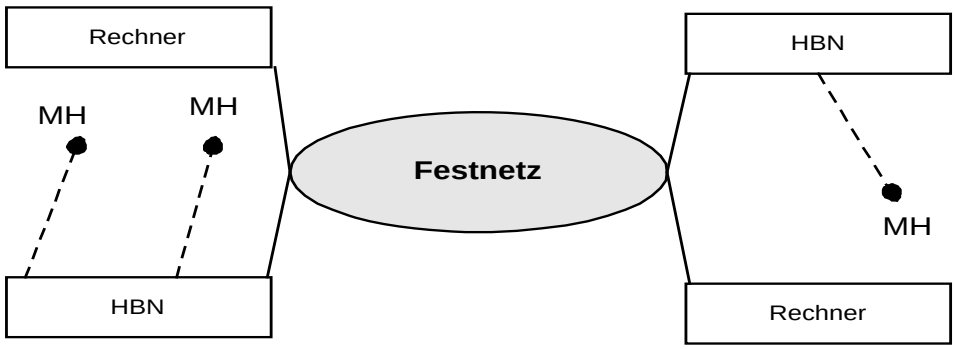

Abbildung 6.1 Netzstruktur bei mobilen Anwendungen

6.1.2 Bewertung

Das Primary-Verfahren gehört zu der Klasse der pessimistischen Replikationsverfahren. Dies sind Verfahren, die keinerlei Inkonsistenzen zwischen den verschiedenen Replikaten einer Kopie zulassen. Somit kann man sagen, daß die Wahrung der Konsistenz durch dieses Verfahren erfüllt wird. Ein MH, der nicht der Primary des Replikates ist, das er besitzt, kann bei Verbindungsunterbrechungen nicht lokal auf seinem Replikat arbeiten, da keine Zustimmung vom Primary erfolgen kann. Somit ist die Verfügbarkeit von Daten bei diesem Verfahren stark eingeschränkt. Dies wäre dann nur durch das Zulassen lokaler Inkonsistenzen auf einem MH möglich. Bei bestehenden Verbindungen muß ein MH, der nicht Primary ist, Verbindung zum Primary aufnehmen, um eine Transaktion genehmigen zu lassen. Dies bedeutet allgemein ein hohes Kommunikationsaufkommen und widerspricht der

Forderung nach Minimierung der Kommunikation. Deshalb ist dieses Verfahren in seiner reinen Form nicht für die Anwendung in mobilen Verarbeitungsumgebungen geeignet.

6.2 Caching in mobilen Systemen

6.2.1 Vorstellung

In [Schrö97] wird ein dynamisches Replikationsverfahren für mobile Umgebungen vorgestellt, daß folgende 4 Punkte beinhaltet :

- Minimierung von Kommunikationskosten und Energieverbrauch
- dynamische Anpassung an variierende Übertragungsbandbreiten
- Benutzerfreundlichkeit durch Verbergen der mobilen Infrastruktur
- permanente Konsistenz der Datenbankobjekte

Eine große Rolle bei diesem Verfahren spielt der Einsatz von Caches bei der Speicherung der Daten auf dem mobilen Gerät. Durch diesen Einsatz wird vor Ort ein schnellerer Zugriff auf die Daten ermöglicht und im Falle von Verbindungsunterbrechungen die Verfügbarkeit der Daten erhöht. Ebenfalls zum Einsatz kommt eine sogenannte Kontingentierung von Daten durch den Server. Dabei erhält jedes mobile Gerät vom Server ein bestimmtes Kontingent von angeforderten Daten, über das es lokal bis zur Rückgabe verfügen kann. Die Vergabe und Umverteilung der Kontingente erfolgt durch den Server. Um dabei die Verfügbarkeit replizierter Datenbankobjekte zu erhöhen, werden temporale Inkonsistenzen erlaubt. Zur Umsetzung des Verfahrens wird zunächst ein Zustandsmodell für mobile Geräte vorgestellt, das folgende Zustände beinhaltet :

- Verbunden
- Verbindungslos
- Schwach-Verbunden
- Minimal-Verbunden
- Schwach/Minimal-Verbunden

- Warten

Das im Vordergrund stehende Ziel in diesen Zuständen ist die effiziente Gestaltung des zu übertragenden Datenvolumens sowie die Kostenminimierung bei der Datenübertragung und die Minimierung des Energieverbrauches beim mobilen Gerät. Die Minimierung des Datenvolumens erfolgt dabei über das Führen von Logbüchern und die damit verbundene Anwendung der Log-Transformation. Ein wichtiges Mittel bei der Kostenminimierung ist die Datenübertragung in Intervallen. Diese Intervalle können vom Nutzer bestimmt werden oder werden in Abhängigkeit von der Änderungsmenge der Daten und dem Optimierungseffekt der Log-Transformation berechnet. Durch diese Intervalle wird ein bestimmtes Maß an Inkonsistenzen toleriert. Der Server hat bei diesem Verfahren die folgenden drei Aufgaben zu bewältigen :

- Vergabe von Kontingenten
- Synchronisation von Manipulationen sonstiger Datenbankobjekte
- Aktualisierung von Replikaten

Die bei der Synchronisation der Datenänderungen auf dem Server auftretenden Konflikte werden durch die Anwendung des Data-Patch-Verfahrens in Verbindung mit der BOCC-Methode behoben.

Die BOCC-Methode (Backward-oriented optimistic concurrency control) unterteilt eine Transaktion in eine Lese-, Validierungs- und Schreibphase. Während der Lesephase kann die Transaktion lesend auf die Datenbank zugreifen. Änderungen werden lokal gespeichert. Nach der Beendigung der Datenmanipulationen tritt die Transaktion in eine Validierungsphase ein. Mögliche Inkonsistenzen werden dabei erkannt und eine der Serialisierbarkeit genügende Tranaktionsausführung durch das Rücksetzen und die erneute Ausführung von Transaktionen erzeugt. Das Erkennen von Inkonsistenzen wird bei dieser Methode durch den Vergleich mit bereits abgeschlossenen Transaktionen erreicht.

Durch den Einsatz der ε-Serialisierbarkeit als Serialisierbarkeitskriterium wird bei dem hier angewendeten Verfahren ein bestimmtes Inkonsistenzmaß toleriert. Einen Überblick über auftretende Konflikte bei der Synchronisation und ihre Beseitigung gibt Tabelle 6.1 Die Aktualisieung der Replikate erfolgt aus Gründen der Kommunikationsreduzierung über das Versenden von Invalidierungsnachrichten durch den Server. Diese infomieren den MH über die Aktualität seiner Replikate. Invalidierte Replikate werden nicht aus den Caches entfernt. Spezielle Invalidierungsnachrichten klären den MH über gelöschte Datenbankobjekte auf dem Server auf. Dieser greift dann zum Ändern dieser Objekte nicht mehr auf den Server zu. Eine Information des Servers über Auslagerungen von Daten aus dem Cache erfolgt als Reaktion auf versuchte Invalidierungen der ausgelagerten Replikate. Bei Verbindungsunterbrechungen werden diese Nachrichten in dafür vorgesehenen Queues gelagert und bei erneutem Verbindungsaufbau ausgetauscht.

Operation	**Konflikte**	**Konflikterkennung**	**Konfliktbehebung**
Ändern von Datenbankobjekten	Lost-Update-Problem	BOCC-Verfahren ohne Berücksichtigung reiner Lesezugriffe	Merge-Prozedur für Änderungen an unterschiedlichen Merkmalen Sequentielle Ausführung von Änderungen an quantifizierten Datenbankobjekten Transaktionsabbruch
	Dirty-Read-Problem		Doppelte Datenhaltung
Einfügen von Datenabnkobjekten	Unterschiedliche Datenbankobjekte mit gleichem Schlüssel Gleiche Datenbankobjekte mit unterschiedlichen Schlüsseln		Identifikation mittels schlüssel über alle eindeutigen Merkmale
Löschen von Datenbankobjekten	Analog zu Änderungen	BOCC-Verfahren	Transaktionabbruch

Tabelle 6.1 Mögliche Konflikte und deren Behandlung [Schrö97]

6.2.2 Bewertung

Dieses Verfahren enthält meiner Meinung nach einige gute Ansätze. Besonderes auffällig ist die starke Berücksichtigung der Kommunikationskosten. Die Beschränkung dieser ist, wie wir bereits wissen, eines der Ziele für mobile Replikationsverfahren, wurde aber in vielen bisher entwickelten Methoden für mobile Umgebungen nicht so sehr berücksichtigt wie hier. Zur Umsetzung verbindet dieses Verfahren klassische optimistische Replikationsansätze mit Neuentwicklungen für das mobile Umfeld, durch die auch die zwei weiteren Ziele für mobile Replikation erfüllt werden (Kontingentierung, ε-Serialisierbarkeit, Übertragungsverzögerung, Invalidierungsnachrichten). Eine Schwäche des Verfahrens ist, daß eine Reaktion auf unbedingte Wahrung der Konsistenz nicht 100%-ig realisierbar ist. Durch das Aufschieben der Übertragung von Datenänderungen vom MH zum Server sind nicht immer die aktuellsten Daten verfügbar und somit volle Konsistenz nicht zu gewährleisten. Dies wäre nur mit einer Erhöhung des Kommunikationsaufkommens möglich. Ein weiterer Kritikpunkt ist meiner Meinung nach die Verwendung der Log-Transformation zur Reduzierung der Änderungsmenge auf dem MH. Das Führen von Logbüchern ist gerade im Falle von Verbindungsunterbrechungen einer sehr speicheraufwendiger Prozess. Speicherressourcen sind aber heutzutage noch nicht in ausreichendem Maße auf mobilen Geräten vorhanden (vielleicht ausgenommen Laptops). Dies gilt auch für die in unserem Fall verwendeten Geräte. Als Fazit kann man somit sagen, daß dieses Verfahren für den Einsatz in mobilen Umgebungen gut geeignet wäre, wenn man noch ein paar notwendige Veränderungen hinsichtlich der Konsistenzsicherung vornehmen würde.

6.3 Clusterbildung

6.3.1 Vorstellung

In [PiBha95,Pit96] wird ein Replikationsverfahren zur Unterstützung von Funkverbindungen in mobilen Systemen vorgestellt. Dabei wird ein gewisses Maß an Inkonsistenzen zugelassen und toleriert. Zur Definition dieser Inkonsistenz werden sogenannte Cluster gebildet, deren Definition ganz unterschiedlich sein kann, z.B. alle Rechner mit einer starken und stabilen Verbindung. Zwischen diesen Clustern kann es dann unterschiedliche Konsistenzgrade geben, die mit dem Buchstaben **d** bezeichnet werden. Im folgenden sei die Definition für diese Konsistenz **d**-ten Grades angegeben :

Definition 6.1 Konsistenz d-ten Grades

Ein Clusterzustand ist konsistent, wenn innerhalb dieses Clusters alle Integritätsbedingungen gewahrt sind. Ein Datenbankzustand ist d-konsistent, wenn alle Clusterzustände konsistent sind und alle Integritätsbedingungen zwischen den Clustern haben Konsistenzgrad d, d.h. differieren um d.

Durch diese begrenzten Inkonsistenzen wird es einem mobilen Gerät ermöglicht, während einer Verbindungsunterbrechung lokal auf seinen Daten arbeiten zu können. Der Inkonsistenzgrad kann bspw. auf die vorhandenen Netzwerkverbindungen abgestimmt werden. Bestehen zumeist stabile Verbindungen zwischen den einzelnen Rechnern, kann **d** klein gehalten werden. Ist die Verbindung hingegen instabil und durch häufige Unterbrechungen gekennzeichnet, sollte man das **d** höher ansetzen. Die unterschiedlichen Werte der Datenkopien sollten dann gelegentlich in Einklang gebracht werden. Dies sollte entweder zu Zeiten einer stabilen Verbindung gemacht werden oder wenn es unbedingt notwendig ist, die Grenzen von **d** nicht zu überschreiten, z.B. aus Gründen der Konsitenzsicherung. Um die lokale bzw. clusterinterne

Datenverarbeitung zu maximieren und die Netzwerknutzung zu reduzieren, erfolgt eine Trennung zwischen schwachen und strikten Transaktionen. Schwache Transaktionen manipulieren nur lokal bzw. innerhalb eines Clusters verfügbare Daten. Strikte Transaktionen hingegen gelten global und sind dauerhaft. Um schwache Transaktionen global zu propagieren, erfolgt nach dem lokalen commit prinzipiell eine Umwandlung in eine strikte Transaktion. Um diese Trennung besser implementieren zu können, erfolgt auch eine Unterscheidung zwischen Core-Kopien von Daten, die dauerhaft und aktuell sind, und Scheinkopien, bspw. in lokalen Caches, deren Werte veraltet sein können. Somit operieren schwache Transaktionen nur auf Scheinkopien. Ein update dieser Kopien erfolgt nur durch eine strikte Transaktion. Zur Begrenzung der Inkonsistenzen werden verschiedene Möglichkeiten vorgegeben, die nun im einzelnen kurz vorgestellt werden.

1. **d** = bis zu **d** Transaktionen können auf inkonsistenten Daten operieren ⇒ die Anzahl von schwachen Schreibtransaktionen innerhalb eines Clusters wird durch **d** begrenzt; danach ist ein globales commit der Änderungen erforderlich.
2. **d** ist die max. Anzahl von abweichenden Versionen eines einzelnen Datenitems ⇒ ist ein Datenitem auf Scheinkopien innerhalb eines Clusters **d** mal verändert worden, muß ein update durch eine strikte Schreibtransaktion erfolgen; dazu ist es aber notwendig, daß eine stabile Verbindung zwischen einzelnen Clusters besteht, da sonst ein globales update nicht möglich ist. Diese Methode ist somit bei instabilen Verbindungen nicht möglich.
3. Jede Kopie kann einen Wert nur innerhalb der Reichweite von **d** Werten annehmen ⇒ dies bezieht sich ausschließlich auf Datenitems mit arithmetischen Werten; es werden somit nur schwache Schreib-transaktionen zugelassen, die ein Datenitem innerhalb der durch **d** vorgegebenen Grenze modifizieren.

4. Bis zu **d** Datenitems dürfen Abweichungen haben ⇒ diese Begrenzung trifft auf Datenitems innerhalb eines Clusters zu; schwache Schreibtransaktionen dürfen somit nur auf einer Menge von **d** Datenitems operieren, bevor ein globales commit erfolgt.
5. Bis zu **d** Kopien eines Datenitems dürfen Abweichungen haben ⇒ die Anzahl von Datenitems, die innerhalb eines Clusters verändert werden dürfen, wird so begrenzt, daß die Gesamtzahl von geänderten Datenitems über alle Cluster **d** nicht überschreitet; dies kann durch eine Beschränkung der schwachen Schreibtransaktionen in jedem Cluster erreicht werden.

Das Konsistenzmaß wird also zum größten Teil über die Begrenzung schwacher Schreibtransaktionen innerhalb bzw. über alle Cluster festgelegt.

6.3.2 Bewertung

Das hier vorgestellte Verfahren erlaubt eine dynamische Anpassung an Gegebenheiten und Einflüsse mobiler Umgebungen. Durch die Tolerierung begrenzter Inkonsistenzen ist ein lokales Arbeiten auf den vorhandenen Daten auch dann möglich, wenn ein mobiles Gerät aufgrund von Verbindungsunterbrechungen nicht erreichbar ist. Das Ziel der Verfügbarkeit ist somit gewährleistet. Der Kommunikations-aufwand wird infolge der periodischen Datenübertragungen z.B. alle **d** Änderungen bzw. bei starken Verbindungen, klein gehalten. Infolge der Möglichkeit zur individuellen Gestaltung des vorgestellten Konsistenz-maßes kann man dynamisch auf die Forderung nach Wahrung der Konsistenz reagieren, indem man **d** möglichst klein hält. Durch die Begrenzung von **d** über alle Cluster erreicht man eine deutliche Einschränkung möglicher Konflikte beim Abgleich der Daten. Allerdings würde die Realisierung und Kontrolle dieser Begrenzung ein erhöhtes Kommunikationsaufkommen erfordern. Zudem müßte dann **d**

von einem Rechner gesteuert werden und ein Auswahlkriterium für diesen Rechner festgelegt werden.

6.4 Weitere Verfahren

An dieser Stelle seien noch ein paar weitere Verfahren bzw. Weiterentwicklungen für mobile Umgebungen kurz vorgestellt und bewertet.

In [FaZa95] wird neben der bereits beschriebenen VPC-Methode eine Möglichkeit für den Einsatz von Voting-Verfahren vorgestellt. Ein Nachteil dabei ist, das durch die Einbeziehung eines oder mehrerer MH´s in das Voting, die Kosten stark ansteigen. Ein Ausweg daraus wäre es, stationären Replikaten ein höheres Voting zu geben als mobilen Replikaten. Im Falle von Verbindungsunterbrechungen würde dann wieder die HBN das Voting für den MH übernehmen. Da es sich hier, wie in Abschnitt 6.1 auch um eine Erweiterung eines pessimistischen Verfahrens, die Inkonsistenzen nicht zulassen und somit nicht alle Replikationsziele erfüllen, handelt, empfiehlt sich diese Methode nicht für den Einsatz in mobilen Anwendungsszenarien.

[Pit98a,Pit98b] stellen ein Server-basiertes Verfahren vor, in dem mobile Clients ausschließlich eine nur-Lese Funktion haben. Dabei überträgt der Server periodisch die komplette vorhandene Datenmenge an eine bestimmte Anzahl mobiler Clients. Der Inhalt der Übertragung ist in jedem Zyklus konsistent. Die Clients lesen dann die für sich notwendigen Daten aus dem Zyklus heraus. Dafür kann das Lesen mehrerer Zyklen notwendig sein. Das Korrektheitskriterium bei diesem Verfahren besteht darin, daß jede nur-Lese Transaktion konsistente Werte liest. Um dies zu gewährleisten, werden in jedem Zyklus zusätzliche Informationen übertragen, die der Client liest und auswertet. Dies kann z.B. ein Invalidationreport sein, der sämtliche Datenitems enthält, die während eines Zyklus‘ geändert wurden. Ist ein

Datenitem x aus diesem Report in der Lesemenge einer Transaktion des Clients enthalten, wird diese Transaktion abgebrochen. Ein gewisser Inkonsistenzgrad kann dabei durch den Einsatz der ε-Serialisierbarkeit gestattet werden. Im Falle von Verbindungsunterbrechungen muß das Schema des Invalidationreports erweitert werden. Dieser enthält dann Versionsnummern der geänderten Datenitems. Diese werden solange akzeptiert, wie die Nummer eine Items kleiner oder gleich ist als die Version, die eine Transaktion zuletzt gelesen hat. Durch den Einsatz von Caches und der Erhöhung der Datengranularität kann man die Effizienz dieses Verfahrens noch weiter erhöhen. Aufgrund des hohen Kommunikationsanteils und der Read-only Funktion der mobilen Geräte ist dieses Verfahren innerhalb der meisten mobilen Szenarien nicht zur Anwendung geeignet, da ein lokales Schreiben auf den Daten i.d.R. notwendig bzw. wünschenswert ist.

6.5 Zusammenfassung

Im vergangenen Kapitel wurden einige Verfahren für den Einsatz in mobilen Anwendungsszenarien vorgestellt. Nach ausführlicher Vorstellung jedes einzelnen Verfahrens erfolgte eine Bewertung hinsichtlich der Erfüllung der Replikationsziele für mobile Umgebungen. Zur weiteren Verwendung für das Anwendungsszenario der Literaturdatenbanken kamen die in Abschnitt 6.2 und 6.3 vorgestellten Verfahren in die engere Auswahl. Das in [Schrö97] vorgestellte Verfahren enthält zwar einige gute Ansätze, aber aufgrund seiner einfachen Strukturierung und den dynamischen Veränderungs-möglichkeiten zur Erfüllung unterschiedlicher Replikationsziele entscheide ich mich zur weiteren Verwendung des in [PiBha95,Pit96] vorgestellten Verfahrens. Zudem ist eine geforderte Parametrisierung des Verfahrens durch seine Struktur bereits vorgegeben. Die Beantwortung offener Fragen und eine genauere Zuschneidung des

Verfahrens auf das Anwendungsszenario der Literaturdatenbanken sowie die Erstellung eines genauen Schemas für die Replikation erfolgt nun im nächsten Kapitel.

7 Replikationsverfahren für das Anwendungsszenario „Literaturdatenbanken“

7.1 Einführung

Wie schon im vorigen Abschnitt erwähnt, fiel die Entscheidung für die Verwendung im Rahmen des Literaturdatenbankenszenarios auf das in [PiBha95,Pit96] vorgestellte Verfahren. Hinsichtlich dieser Verwendung müssen aber noch einige Fragen, unter anderem auch zu den Parametern aus dem obig vorgestellten Verfahren, beantwortet werden, wie :

- Welche Größe hat ein Cluster?
- Wie wird **d** definiert?
- Wie wird **d** verwaltet und festgelegt?
- Welche zusätzlichen Metadaten sind sowohl auf dem mobilen Gerät als auch auf dem Server notwendig.

Diese und weitere Fragen werden in den nächsten Abschnitten beantwortet. Dazu erfolgt zunächst eine Abgrenzung des Verfahrens im Bezug auf das Szenario. Dann wird ein genaueres Modell für das Replikationsverfahren vorgestellt und im letzten Abschnitt soll beispielhaft geklärt werden, wie das Verfahren arbeitet.

7.2 Festlegung der Parameter

In diesem Abschnitt soll geklärt werden, inwiefern die in dem in Abschnitt 6.3 vorgestellten Verfahren eingesetzten Parameter auch im Szenario der Literaturdatenbanken anwendbar sind. Zur Festlegung eines bestimmten Inkonsistenzgrades wurde die Möglichkeit der

Clusterbildung vorgestellt, zwischen denen unterschiedliche Konsistenzgrade existieren durften. Zur Festlegung dieses Konsistenzgrades wurde ein sogenanntes **d** vorgegeben, für das unterschiedliche Definitionen möglich sind. Diese sollen, bezogen auf die in Abschnitt 4.3 genannten Eigenschaften von Literaturdatenbanken, im Folgenden bewertet und eine Auswahl möglicher einsetzbarer Definitionen getroffen werden.

7.2.1 Der Konsistenzparameter d

Für die Bewertung werden die einzelnen Kurzdefinitionen und deren Interpretation für die Umsetzung noch einmal kurz genannt. Daraufhin erfolgt eine Bewertung im Bezug auf die Eigenschaften von Literaturdaten und somit eine Aussage, ob und in welcher Form ein Anwendung der einzelnen Definitionen erfolgen kann.

1. Bis zu **d** Transaktionen können auf inkonsistenten Daten operieren ⇒ dazu wird die Anzahl der schwachen Schreibtransaktionen in jedem Cluster begrenzt ⇒ Dadurch ist es möglich, während einer Verbindungsunterbrechung bzw. ohne unnötig mit dem Umfeld zu kommunizieren lokal auf einem mobilen Gerät bzw. innerhalb eines Clusters auf vorhandenen Daten zu arbeiten, was für den Erhalt der Verfügbarkeit sehr wichtig ist. Daher kann diese Definition ohne weitere Kritik übernommen werden.
2. **d** ist die max. Anzahl von abweichenden Versionen pro Datenitem ⇒ innerhalb eines Clusters kann ein einzelnes Datenitem bis zu **d**-mal durch eine lokale Schreiboperation verändert werden, bevor die Änderung global gemacht wird. ⇒ Wie schon in Abschnitt 4.3 erwähnt, sind Literaturdaten nach ihrer Erstellung relativ konstante Daten, d.h. sie unterliegen keinen großen Veränderungen mehr. Es ist also unwahrscheinlich, daß ein erstellter Literatureintrag noch bis zu **d** mal verändert wird. Somit halte ich den Einsatz dieser Definition für **d** im Rahmen des Anwendungsszenarios der

Literaturdatenbanken zwar für anwendbar, aber nicht sinnvoll zur Konsistenzerhöhung einsetzbar.

3. jede Kopie kann einen Wert nur innerhalb einer Reichweite von **d** Werten annehmen ⇒ nur schwache Schreibtransaktionen mit Werten innerhalb des erlaubten Wertebereiches werden akzeptiert ⇒ Da es sich bei dieser Definition um arithmetische Werte handelt und Literaturdaten nicht dieser Eigenschaft entsprechen, ist auch diese Definition für **d** im Rahmen der Literaturdatenbanken nicht sinnvoll einsetzbar.
4. bis zu **d** Datenitems dürfen Abweichungen haben ⇒ Es können innerhalb eines Clustern bis zu **d** verschiedene Datenitems durch schwache Schreibtransaktionen manipuliert werden ⇒ Aufgrund der Eigenschaften von Literaturdaten kann man davon ausgehen, daß eine Schreibtransaktion einen Wert löscht oder einen Neueintrag vornimmt. Somit greift eine Schreibtransaktion max. zweimal auf ein einzelnes Datenitem zu. Somit ist diese Definition, bezogen auf Literaturdaten sehr ähnlich zu der ersten Definition für **d**. Eventuell wäre hier vielleicht eine Beschränkung von Neueinträgen pro Cluster als sinnvoll anzusehen.
5. bis zu **d** Datenkopien pro Datenitem dürfen Abweichungen haben ⇒ ein Datenitem darf in bis zu **d** Clustern verändert werden ⇒ Laut dem in Abschnitt 6.3 vorgestellten Verfahren, können Cluster sich über den Verbund mehrerer Rechner erstrecken. Eine clusterübergreifende Kontrolle von **d** würde somit eine fortwährende Kommunikation zwischen mobilen und festen Rechnern bedeuten. Dies ist zum Einen sehr kostenintensiv und kann zum Anderen aufgrund der in mobilen Netzen fortwährend bestehenden Gefahr von Verbindungsunterbrechungen nicht dauerhaft gewährleistet werden. Wenn es allerdings eine zentrale Verwaltungsstelle für so ein **d** geben würde, wäre eine Definition für **d**, wie in der folgenden

Form, möglich : **d** ist die Anzahl der schwachen Schreibtransaktionen über alle Cluster.

Man kann somit feststellen, daß für den Einsatz in speziellen Replikationsverfahren für die Literaturdatenbanken nur Punkt 1 vollständig und Punkt 5 mit Einschränkungen als sinnvoll anzusehen sind.

7.2.2 Clustergröße

Die Frage nach der Clustergröße wird in [PiBha95,Pit96] nicht genau beantwortet. Es werden verschiedene Möglichkeiten vorgegeben, wie Rechner mit stabilen Verbindungen, eine Einteilung nach Nutzern oder nach semantischem Wissen. Die Möglichkeit, einen Cluster durch den Verbund mehrerer Rechner zu definieren, möchte ich von vornherein ausschließen. Eine Kontrolle von **d** über mehrere Rechner wäre nämlich, wie weiter oben bereits erwähnt, sehr kostenintensiv und ließe sich nur schwer verwalten, d.h. diese Möglichkeit bestünde nur über einen ausgezeichneten Rechner innerhalb eines Clusters, der die Kontrolle von **d** übernehmen würde. Es ist aber schwer festzulegen, welcher Rechner dies sein sollte. Eine weitere Möglichkeit bestünde meiner Meinung nach darin, die Clustergrenzen um ein einzelnes Fragment herum zu ziehen. Somit würde jedes Replikat dieses Fragmentes zu einem Cluster gehören. Auch diese Methode hat aus den oben genannten Gründen den Nachteil, daß sie sehr kostenintensiv ist. Eine Kontrolle von **d** erscheint mir dann über den Rechner als sinnvoll, der Besitzer des Originalfragmentes ist. Die einfachste und meiner Meinung nach beste Methode wäre aber, die Clustergrenze um ein einzelnes mobiles Gerät, besser noch um eine konkrete Anwendung herum zu ziehen. Die Kontrolle von **d** wäre dann nämlich für den obigen Punkt 1 der möglichen Definitionen lokal und ohne Kommunikationsaufwand zu vollziehen und würde nur für die Daten der konkreten Anwendung erfolgen. Der Wert für **d** könnte dann von dem Rechner festgelegt und übermittelt werden, von dem das mobile

Gerät die Daten für die jeweilige Anwendung empfängt. Im Falle einer Netzstruktur wie in Abbildung 6.1 wäre das die jeweilige Sende- und Empfangseinheit bzw. die HBN.
Soweit sind die ersten Fragen zur Definition von **d** und zu Clustergröße für den Einsatz der in [PiBha95,Pit96] genannten Parameter beantwortet. Im nächsten Abschnitt soll nun geklärt werden, wie und in welcher Form **d** festgelegt und kontrolliert wird und welche zusätzlichen Informationen zur Verwaltung benötigt werden, so daß sich daraus ein Schema für ein Replikationsverfahren ergibt, das im Literaturdatenbankszenario eingesetzt werden kann.

7.3 Das Replikationsmodell

7.3.1 Netzstruktur

Das Szenario der Literaturdatenbanken findet in dieser Arbeit in einer Netzstruktur seine Umsetzung, die nicht typisch für mobile Szenarien ist. Ein typisches mobiles Szenario ist in Abbildung 6.1 dargestellt. Im Rahmen dieser Arbeit geht man davon aus, daß ein oder mehrere MH´s über ein Funknetz mit einem Server verbunden sind, der sämtliche Datenbestände auf sich vereint. Dieses Struktur wird in Abbildung 7.1 dargestellt.

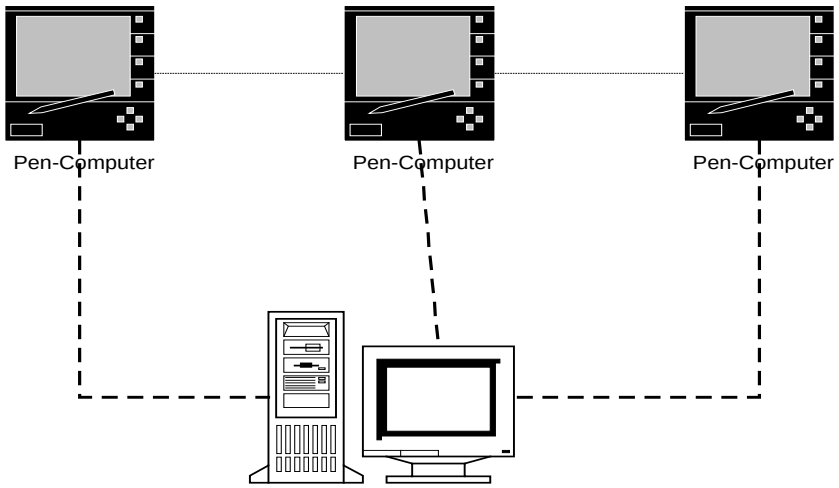

Abbildung 7.1 Netzstruktur für Literaturdatenbankenszenario

Somit ergibt sich beim Replikationsverfahren eine Aufgabenverteilung, bei der nur zwischen Server und MH unterschieden werden muß. Diese Aufgabenverteilung wird im nächsten Abschnitt konkretisiert.

7.3.2 Aufgabenverteilung

Server

Für den Server ergibt sich aus der Definition für **d** heraus die Aufgabe, dieses für die einzelnen MH´s festzulegen und zu propagieren. Je nach Art des angestrebten Replikationszieles kann bei der Festlegung zwischen Definition 1 und Definition 5 aus Abschnitt 7.1.1 unterschieden werden, d.h. ist das Ziel die Erhöhung der Verfügbarkeit auf einem MH, z.B. aufgrund einer sehr instabilen Verbindung, sollte **d** möglichst hoch gewählt werden, um so ein maximales lokales Arbeiten zu ermöglichen. Ist jedoch eine stärkere Konsistenz der Daten erforderlich oder bestehen stabile Verbindungen zu den MH´s, so kann **d** niedrig gewählt werden.

Beim Empfang der veränderten Daten vom MH ergibt sich zudem die Aufgabe, diese mit den eigenen Datenbeständen abzugleichen und die Änderungen, falls notwendig, an andere MH´s zu propagieren. Dies kann, wie bei dem in Abschnitt 6.2 vorgestellten Verfahren mit Invalidationreports geschehen. Sollte die Verbindung zu einem MH unterbrochen sein, so müssen diese Meldungen in einer Queue gesammelt werden und nach Wiederaufbau der Verbindung übertragen werden. Die Synchronisation der eingehenden Datenänderungen kann mit bisherigen bekannten Verfahren zur Concurrency Control erfolgen. Zusammenfassend ergeben sich also folgende 3 Aufgaben :

- Festlegung der Art und Größe von **d**
- Synchronisation der eingehenden Datenänderungen
- Propagierung der Datenänderungen an die Replikate

Zum ersten Punkt sei hier noch anzumerken, daß eine Verbindung zwischen den beiden Definitionen für **d** aus Gründen der Konflikt-

vermeidung ratsam erscheint. Man stelle sich dafür vor, daß **d** einen relativ hohen Wert hat und die Anzahl der mit dem Server kommunizierenden MH´s steigt. Für den Fall, daß dann alle MH´s ihre geänderten Datenbestände in relativ kurzen Abständen an den Server senden, ergibt sich ein zu hohes und eventuell nicht auflösbares Konfliktpotential bei der Datensynchronisation auf dem Server. Somit sollte ab einer gewissen Anzahl von angemeldeten MH´s oder gleich von Anfang an eine Gleichverteilung von **d** auf jedem einzelnen angemeldeten MH angestrebt werden, um das Konfliktpotential auf dem Server möglichst gering zu halten.

MH

Dieser hat eigentlich nur die Aufgabe, **d** gemäß der Vorgabe vom Server zu kontrollieren und ankommende Invalidationreports vom Server auszuwerten. Bei Erreichen der Grenze von **d** müßten die veränderten Daten gesammelt und an den Server gesendet werden. Ist dies aufgrund von Verbindungsunterbrechungen nicht möglich, so sollte dies bei Wiederaufbau der Verbindung geschehen.

7.3.3 Metadaten

In diesem Abschnitt wird nun noch erklärt, welche zusätzlichen Metadaten zur Umsetzung der einzelnen Aufgaben auf dem Server und dem MH nötig sind.

Server

Für den Server ist aus dem Allokationsschema ersichtlich, welcher MH in Verbindung mit ihm steht und Daten von ihm besitzt. Für jeden dieser MH´s hält der Server einen Wert für das dem jeweiligen MH zugewiesene **d** und für den zuletzt bekannten aktuellen Wert von **d**. Dieser kann, sofern eine Verbindung zu dem MH besteht, jederzeit von diesem abgerufen werden.

MH

Für den MH ergibt sich gemäß obiger Ausführungen die Notwendigkeit, ebenfalls seinen aktuellen und den vom Server zugewiesenen Wert für **d**

konstant zu speichern. Zudem sollte der MH eine Kopie des Allokationsschemas des Servers besitzen. Für den Fall des Ausfalls des Servers könnten dann Datenänderungen direkt, falls möglich an die betroffenen MH´s geschickt werden, um ein weiteres lokales Arbeiten auf dem MH zu ermöglichen bzw. benötigte Daten von einem anderen, diese Daten besitzenden MH angefordert werden.
Im nächsten Abschnitt wird nun konkret erklärt, wie die einzelnen Aufgaben bewältigt werden.

7.4 Einsatz des Replikationsverfahrens

In diesem Abschnitt wird nun genau erklärt, wie die Kontrolle und die Festlegung von **d** funktioniert. Dabei soll auch eine Verbindung zwischen den für das Literaturdatenbankenszenario sinnvollen Definitionen für **d** hergestellt werden, die da wären :

- **d** ist die maximale Anzahl von lokalen Schreiboperationen in einem Cluster
- **d** ist die maximale Anzahl von lokalen Schreiboperationen über alle Cluster

Ein Cluster wäre hier gleichzusetzen mit einer einzelnen Anwendung für Literaturdatenbanken auf einem einzelnen MH.
Man gehe nun davon aus, daß laut Abbildung 7.1 mehrere MH´s in Verbindung mit dem Server treten, um verschiedene Literaturdaten abzufragen bzw. Neueinträge vorzunehmen. Ein Ablauf würde dann wie folgt aussehen :
Ein MH meldet sich beim Server per Anfrage für die Nutzung der Literaturdatenbank an. Er bekommt, falls angefordert, Literaturdaten und in jedem Fall einen Wert für **d** vom Server übermittelt. Dieser speichert nun für den MH den zugewiesenen und den aktuellen Wert für **d** (zu Anfang =0). Für jede einzelne lokale Schreibtransaktion innerhalb der Anwendung wird nun der aktuelle Wert für **d** hochgezählt und mit

dem vorgegebenen **d** verglichen. Sind beide Werte gleich bzw. der aktuelle Wert größer, werden die neuen bzw. veränderten Datenbestände an den Server transferiert und der aktuelle Wert für **d** wieder auf Null gesetzt. Sind die abgeänderten Daten auf dem Server angekommen, werden diese mit den einzelnen betroffenen Fragmenten auf dem Server abgeglichen und, falls notwendig, über Invalidation-reports an die einzelnen Kopien der Fragmente auf anderen MH´s propagiert.

Wie bereits erwähnt, könnte diese Vorgehensweise mit steigender Anzahl der angemeldeten MH´s zu einem nicht mehr auflösbaren Konflikt auf dem Server bei der Datensynchronisation führen. Eine Lösung wäre es, ab einer bestimmten max. Zahl von angemeldeten MH´s die lokalen **d**´s so zu verringern, das ein festgesetztes globales **d** nicht überschritten wird. Ob dies für jeden neu angemeldeten MH oder erst ab einer bestimmten Anzahl von MH´s geschieht, sollte von der jeweiligen Anwendung abhängig gemacht werden. Die Umsetzung könnte dann aber folgendermaßen aussehen :

Meldet sich ein neuer MH i für die Anwendung der Literaturdatenbank an, berechnet sich dessen Wert für **d** aus folgender Formel :

$$d_i = \frac{d_{\max} - \sum_{j=1}^{i-1} d_{j_{aktuell}}}{i}$$

Es werden also sämtliche aktuellen **d**´s der bisherigen MH´s addiert und vom maximalen Wert für **d** über alle Cluster subtrahiert. Diese Zahl wird durch die Anzahl der angemeldeten MH´s, also einschließlich dem MH i, dividiert und dem neu angemeldeten MH als neuer Wert für sein lokales **d** zugewiesen (gebrochene Werte werden gerundet). Sollte aufgrund von Verbindungsunterbrechungen der aktuelle Wert für **d** eines MH´s nicht abrufbar sein, wird der Wert genommen, der bei der

letzten Abrufung von **d** bestand (im schlimmsten Fall 0). Bei jedem Abruf dieses Wertes wird dieser auf dem Server gespeichert. Bei den bereits angemeldeten MH´s wird das jeweilige $\mathbf{d}_{aktuell}$ mit dem Wert der Division addiert und als neues maximales **d** eingesetzt. Der neue Wert für **d** auf einem einzelnen MH würde sich also aus folgender Formel berechnen :

$$d_{j\{\,j=1,\ldots,i-1\}} = d_{j_{aktuell}\ \{\,j=1,\ldots,i-1\}} + d_i$$

Somit wird bei steigender Anzahl von MH´s der lokale Inkonsistenzgrad herabgesetzt und eine Erhöhung des globalen Konfliktpotentials vermieden. Erreicht ein MH dann den lokalen max. Wert für **d**, wird die geänderte Datenmenge wieder an den Server geschickt. Ergibt die Division von **d** und den aktuellen **d**-Werten der MH´s einen Wert für **d**, der kleiner als i ist (=Anzahl der angemeldeten MH´s), sollte der Server eine Übertragung der Änderungsmenge von jedem MH für eine Abgleichung veranlassen, **d** sollte gleich unter allen angemeldeten MH´s aufgeteilt und die aktuellen Werte für **d** wieder auf null gesetzt werden.

7.5 Zusammenfassung

Mit dem hier vorgestellten Verfahren wird ein Ansatz zur Replikationskontrolle präsentiert, der durch die Definition eines Inkonsistenzmaßes **d** für lokal definierte Cluster auf einem mobilen Gerät und dessen Veränderung auf die unterschiedlichen Ziele der Replikation reagieren kann. Durch die Möglichkeit lokaler Inkonsistenzen wird zum einen die Verfügbarkeit erhöht und somit ein maximales lokales Arbeiten ermöglicht sowie der Kommunikations-aufwand gering gehalten, da nicht jede Datenänderung sofort propagiert werden muß. Zum anderen kann durch eine Verringerung von **d** bzw.

eine Gleichverteilung über mehrere MH´s der Konsistenzgrad des Systems erhöht werden. Für das Beispiel der Literaturdaten ist die Definition des Konsitenzparameters **d** aufgrund der Dateneigenschaften beschränkt. Es ergibt sich nur eine Begrenzung der lokalen Schreiboperationen bis zur Updatepropagierung bzw. eine gleichverteilte Begrenzung der Schreiboperationen global gesehen. Durch die Festlegung der Clustergröße auf einen MH, speziell auf eine einzelne Anwendung, wird zudem eine einfache und nicht sehr kostenaufwendige Kontrolle von d ermöglicht. Zusammenfassend kann man sagen, daß die Auswahl und die Erstellung eines Replikationsverfahrens sehr stark vom jeweiligen Anwendungskontext abhängig ist. Als Beispiel sei nur zu nennen, daß aufgrund der Eigenschaften der Literaturdaten nicht alle in Abschnitt 6.3 genannten Definitionen für d sinnvoll innerhalb des hier betrachteten Szenarios einsetzbar waren.

Nun ist dieses Verfahren auf ein bestimmtes Anwendungsszenario zugeschnitten worden, und das für eine, bei mobilen Umgebungen eher atypische Netzstruktur (Vergleiche Abbildung 6.1 und Abbildung 7.1). Wie ein solches Verfahren in andere mobile Szenarien eingebaut werden kann bzw. welche Erweiterungsmöglichkeiten sich ergeben, wird in Kapitel 9 verdeutlicht. Im nächsten Kapitel erfolgt eine kurze Darstellung der Implementationsumgebung und des Ablaufes bei der Implementation.

8 Implementation des Verfahrens

Das in dieser Arbeit vorgestellte Verfahren wird im Rahmen einer Testimplementation umgesetzt, um die Reaktionen des Verfahrens auf einen Orts- und somit Kontextwechsel zu demonstrieren. Die Implementation erfolgt mit C++. In diesem Kapitel erfolgt daher eine Beschreibung der Implementationsumgebung sowie des geplanten Simulationsablaufes.

8.1 Implementationsumgebung

Zur Umsetzung wird das Verfahren auf einem für mobile Geräte entwickelten Parser für die Transformation von DML-Ausdrücken (SQL-DML) in eine interne Form der Repräsentation aufgesetzt. Der Parser ist für den Einsatz auf einem mobilen Gerät konzipiert. Die interne Form der Repräsentation besteht aus einer Klasse, deren Instanzen durch mehrere Parameter gekennzeichnet sind und die auszuführenden Aktionen sowie die dafür benötigten Daten umfassen. Die sind :

- Aktionscode (Select, Insert Update, Delete)
- Liste für Attribut-Werte-Paare für Insert und Update
- Anfragebaum zur Selektion der zu bearbeitenden Daten

Zudem gehören zur internen Aktionsrepräsentation Tabellen für die Zuordnung der im Anfragebaum definierten logischen Relationen zu den physischen Relationen im globalen konzeptuellen Schema und weitere, nur innerhalb der Verarbeitungsebene verwendete Daten.
Wenn nun ein SQL-Ausdruck verarbeitet werden soll, wird dieser zunächst zerlegt und eine Grammatikprüfung vorgenommen. Dabei werden die internen Tabellen und der Anfragebaum aufgebaut. Im Falle einer Insert- oder Updateoperation werden die Attribut-Werte-Paare

erkannt und zwischengespeichert. Aus den gewonnenen Daten wird dann eine sogenannte Aktionsinstanz aufgebaut. Diese bildet den Ausgangspunkt für die verteilte Anfrage- und Updateverarbeitung. Diese werden in Verarbeitungszweige unterteilt, wobei jeder Zweig aus einer Liste von Methoden besteht, die nacheinander aufgerufen werden und denen als Parameter ein Zeiger auf die zu manipulierende Aktionsinstanz übergeben wird. Uns interessiert dabei besonders die Update-Bearbeitung, da die Anfrageverarbeitung nur Leseoperationen realisiert, bei dem hier angewendeten Replikationsverfahren aber nur Schreiboperationen Beachtung finden. Wird also ein SQL-Kommando zur Datenmanipulation ausgelöst, werden Routinen zur Prüfung von Integritätsbedingungen, zur Fragmentation, Allokation, Optimierung von Schreibzugriffen und zur globalen Updateverarbeitung aufgerufen. Der Anfragebaum der zu bearbeitenden Daten wird dann in mehrere lokale (Updates auf mehreren Rechnerknoten) und einen globalen Bereich (für die Auswertung der Ergebnisse der lokal durchgeführten Aktionen) logisch unterteilt. Diese Unterteilung wird anhand der im Anfragebaum gefundenen Fragmentinstanzen vorgenommen. Ist nun eine Fragmentinstanz der einzige Subknoten eines Fragmentes, wird direkt eine lokale Updateroutine aufgerufen, d.h. existiert nur ein einziges Replikat eines zu verändernden Fragmentes, wird die Updateoperation sofort ausgeführt. Ist dies nicht der Fall, erfolgt der Einsatz eines Replikationsverfahrens, welches eine Propagierung der Aktionen zu den einzelnen Replikaten gemäß der eingesetzten Strategie vornimmt. An dieser Stelle erfolgt der Einsatz des in Kapitel 7 vorgestellten Verfahrens.

8.2 Simulationsablauf

Die Simulation erfolgt mit einem PDA-Gerät, einem Psion, mit dem über Funk auf die auf dem Server liegende Literaturdatenbank

zugegriffen werden kann. Ziel ist es, einen Orts- bzw. Kontextwechsel beim Zugriff auf die Literaturdatenbank zu simulieren, durch den ein Wechsel bei dem geforderten Replikationsziel erfolgt. Dieser Wechsel soll der Einfachheit halber über eine Tastatureingabe gesteuert werden. Das Replikationsziel soll dann entweder die Erhöhung der Verfügbarkeit bzw. die Wahrung der Konsistenz sein. Durch die dann individuelle Gestaltung von **d** soll auf das jeweilige Ziel eingegangen werden, d.h., für das Ziel der Verfügbarkeit ein möglichst hohes **d** und für das Ziel der Konsistenzwahrung eine Gleichverteilung bzw. Herabsetzung von **d** auf den beiden mobilen Geräten. Die Festlegung der **d**-Werte erfolgt dabei durch den Server. Die Werte von **d** sollen dann als Reaktion auf ein verändertes Replikationsziel auf dem Bildschirm ausgegeben werden.

Ein interner Programmablauf auf dem mobilen Gerät soll dann wie folgt aussehen :

Der Parser übergibt dem Replikationsverfahren, wie weiter oben beschrieben, einen Zeiger auf die veränderte Fragmentinstanz zur weiteren Propagierung an weitere Replikate. Durch das Verfahren würden dann die in Metadaten gespeicherten Werte für **d** und $\mathbf{d}_{\text{aktuell}}$ ausgelesen werden(Siehe Abschnitt 7.3). Der Wert für **d** ist dabei gemäß des angestrebten Replikationszieles bei der Datenvergabe vom Server vorgegeben worden. Im nächsten Schritt würde $\mathbf{d}_{\text{aktuell}}$ um 1 erhöht und mit **d** verglichen werden. Sind beide Werte gleich oder $\mathbf{d}_{\text{aktuell}} > \mathbf{d}$, so sollte eine Propagierung aller bisher veränderten Fragmentinstanzen an den Server erfolgen. Ist dies nicht der Fall, so sollte die Propagierung der Veränderung zurückgehalten werden, d.h. bezogen auf den Ablauf einer Datenmanipulierungsoperation, daß ein globales Update von allen lokalen Datenänderungen erst nach einer Anzahl von **d** Manipulationsoperationen erfolgen würde, wobei der Wert für **d** je nach Art und Weise des Replikationszieles definiert ist.

9 Einordnung und Problembeschreibung

Wie Bereits in Kapitel 7 erwähnt, ist das im Rahmen dieser Arbeit beschriebene Verfahren durch die Verwendung innerhalb des Literaturdatenbankenszenarios stark in seinen Möglichkeiten eingeschränkt. Ein Grund dafür ist, daß die Eigenschaften der Literaturdaten nicht jede Definition für den Konsistenzparameter **d** möglich machen. Zudem wurde eine eher atypische Netzstruktur für das Szenario gewählt, die nicht einem typischen mobilen Szenario entspricht. Dieses wird in Abbildung 6.1 dargestellt. Dabei bilden ein oder mehrere MH´s und eine Sende- und Empfangseinheit, auch HBN genannt, sogenannte Funkzellen, in denen Kommunikation zwischen den einzelnen Rechnern über eine Funkverbindung betrieben werden kann. Eine HBN ist zudem über das Festnetz mit weiteren stationären Geräten verbunden, die entweder Datenserver oder weitere HBN´s sind. Während also in unserem Beispiel nur mit mobilen Replikaten gearbeitet wird, ergibt sich aufgrund der veränderten Netzstruktur nun eine neue Situation. Durch das Vorhandensein von stationären und mobilen Replikaten können sich neue Anforderungen an die Replikatsverwaltung ergeben, auf die in diesem Kapitel näher eingegangen werden soll. Dazu erfolgt im nächsten Abschnitt eine Beschreibung zur Möglichkeit des Einsatzes des in Kapitel 7 beschriebenen Verfahrens in einem solchen Szenario.

9.1 Verfahrenswechsel in mobilen Szenarien

Aufgrund der Existenz stationärer und mobiler Replikate innerhalb einer einzigen Anwendung ergeben für Replikate ein und desselben Datums veränderte Replikationsziele. Für stationäre Replikate können aufgrund der bestehenden stabilen Verbindungseinrichtungen und der Ressourcen des replikatbesitzenden Rechners Replikationsziele, wie Verfügbarkeit und Minimierung der Kommunikationskosten, zugunsten

der Konsistenzwahrung vernachlässigt werden. Die Verwaltung dieser Replikate könnte, wie in normalen verteilten Systemen, über herkömmliche, in Kapitel 2 beschriebene Replikationsverfahren erfolgen. Datenänderungen können unmittelbar an die weiteren vorhandenen Replikate propagiert werden. Dies könnte für ein mobiles Replikat bedeuten, daß die zuständige HBN die Datenänderungen erhält und an den jeweiligen MH über Invalidationreports propagiert.

Für die Verwaltung mobiler Replikate könnte dann das in Kapitel 7 beschriebene Verfahren eingesetzt werden. Aufgrund der veränderten Netzstruktur wären dann aber einige Abänderungen bezüglich der Aufgabenverteilung bei der Replikatsverwaltung notwendig. Dazu wäre unter anderem zu klären, ob die Definition für die Clustergröße, wie in Abschnitt 7.2 beschrieben, auch zu dieser Netzstruktur paßt. Eine Kontrolle und Festlegung von **d** müßte dann über die jeweils zuständigen HBN´s erfolgen, da eine zentrale Vergabe durch eine Server aufgrund der verteilten Datenhaltung hier nicht zu realisieren wäre. Eine weitere Frage, die sich dann ergibt, ist die nach der globalen Kontrolle von **d**. Dafür müßte ein ausgezeichneter Rechner existieren, der den HBN´s maximale Werte für **d** zuweist. Eine Festlegung diesbezüglich gestaltet sich momentan sehr schwierig. Somit sollte nach anderen Möglichkeiten für die Definition der Clustergröße gesucht werden, die eine dezentrale Kontrolle von **d** erlauben. Meiner Meinung nach bieten sich dafür zwei Möglichkeiten an.

1. Ein Cluster entspricht einem einzelnen Fragment.

   D.h., daß alle Replikate ein und desselben Fragmentes zu einem Cluster gehören. Eine HBN könnte dann, abhängig von der Anzahl der Datenkopien eines einzelnen Fragmentes und vom geforderten Replikationsziel eine Zuweisung für **d** an einen einzelnen MH vornehmen. Ein MH müßte somit für jedes seiner Fragmente einen Werte für **d** führen.

2. Ein Cluster entspricht einer Funkzelle mit einer HBN und einem oder mehreren MH´s.

   Der Konsistenzparameter **d** sollte dann nur noch innerhalb des jeweiligen Clusters festgelegt und kontrolliert werden. Die Funktionen des Servers werden dann von der jeweiligen HBN übernommen, d.h. die Kontrolle und Festlegung von **d** und die globale Propagierung von Datenänderungen. Dabei sollte sich die der Maximalwert für **d** innerhalb einer einzelnen Funkzelle an der globalen Gesamtanzahl der sich innerhalb des Clusters befindenden Replikate orientieren. D.h., ist die Gesamtanzahl der Replikate, die sich in einem Cluster befinden, über das komplette Verteilungsschema groß, so sollte **d** für einen einzelnen Cluster niedriger gewählt werden als wenn die Gesamtanzahl der Replikate klein wäre. Ich schlage für die Verteilung und Kontrolle von **d** innerhalb der Funkzelle dann eine ähnliche Vorgehensweise wie in Kapitel 7 beschrieben vor. Dabei sollte die Höhe von **d** auf den einzelnen MH´s von dem jeweiligen Replikationsziel bzw. von der Beschaffenheit der Daten abhängig gemacht werden.

In beiden Fällen wäre es notwendig, das die HBN ein fortwährend aktuelle Kopie des Allokationsschemas besitzt, um zum einen die Werte für **d** festlegen zu können und zum anderen eine Propagierung der Datenänderungen auf den MH´s vornehmen zu können.

Ein weiteres Problem beim Einsatz des Replikationsverfahrens ergibt sich bei der Definition des Konsistenzparameters **d**. Inwieweit die unter Abschnitt 6.3 genannten Definitionen für **d** innerhalb eines Szenarios einsetzbar sind, hängt, wie auch schon in Kapitel 7 anhand der Literaturdaten erwiesen, von der jeweiligen Anwendung und der Beschaffenheit der Anwendungsdaten ab. So wäre z.B. für numerische und änderungsintensive Daten ein Einsatz von Definition 2 und 3 aus Abschnitt 6.3 durchaus denkbar.

Auch die Höhe des Wertes für ein **d** hängt, neben dem geforderten Replikationsziel, von der Beschaffenheit der Daten ab. Sind die Daten

einer Anwendung sehr speicheraufwendig, so sollte **d** niedriger gewählt werden, da die zu übertragende Datenmenge sonst zu groß wird, und somit die Kommunikationskosten steigen. Hingegen bei speichersparenden Daten, wie z.B. numerischen Daten, die zudem häufigen Änderungen unterliegen, sollte **d** größer gewählt werden, da ansonsten die Übertragungsfrequenzen sehr klein sind und sich das ebenfalls auf die Kommunikationskosten auswirkt. Bezogen auf das jeweilige Replikationsziel kann man zunächst feststellen, daß das wichtigste Ziel für mobile Replikation zumeist die Erhöhung der Verfügbarkeit der Daten ist, da ein lokales Arbeiten im Laufe von Verbindungsunterbrechungen ansonsten stark eingeschränkt oder nicht möglich wäre. Somit sollte der Wert für **d** auf einem MH im Allgemeinen hoch angesetzt werden. Es ist aber möglich, daß aufgrund eines sich ergebenden Kontextes für ein mobiles Replikat eine stärkere Einhaltung der Konsistenz gefordert wird. In diesem Fall sollte man entweder **d** niedrig halten bzw. von der Gleichverteilung für **d** abkehren. Eine individuelle Zuweisung von **d** wäre denkbar, so daß der replikatsbesitzende MH mit stärkerer Konsistenzanforderung einen niedrigeren Wert für **d** erhält als andere MH´s.
Soweit sei hier eine Beschreibung für eine Möglichkeit der Anwendung des in Kapitel 7 beschriebenen Verfahrens in klassischen mobilen Szenarien gegeben. Aufgrund der teilweise veränderten Umgebung ergeben sich für die Anwendung Probleme bzw. neue Anforderungen, die im nächsten Abschnitt noch einmal genauer spezifiziert werden, so daß im Anschluß daran eine Aufstellung für zukünftige Aufgaben erfolgen kann.

9.2 Problemanalyse

Aufgrund des veränderten Szenarios ergaben sich in der obigen Beschreibung Probleme beim Einsatz des in Kapitel beschriebenen

Replikationsverfahrens zur Verwaltung der mobilen Replikate. Besonders zwei Fragen konnten nicht genau beantwortet werden :

- Wie wird die Clustergröße definiert? und
- Wie und wie hoch lege ich **d** für einen einzelnen Cluster bzw. MH fest.

Dafür wurden erste Vorschläge gemacht, die für den weiteren Einsatz genauer durchdacht werden müßten. Dies soll aber nicht Aufgabe dieser Arbeit sein. Es kann aber jetzt schon gesagt werden, daß die Beantwortung beider Fragen stark vom jeweiligen Anwendungskontext abhängig gemacht werden sollte.

Bezüglich des zweiten Punktes der Festlegung von **d** will ich hier noch einige Anmerkungen machen. Es ist zwar möglich, den Konsistenzparameter **d** auf ein konkretes Replikationsziel, das sich aus einem gegebenen mobilen Kontext ergibt, zuzuschneiden, wie die Werte von **d** dann aber genau zu gestalten sind, ist momentan noch nicht spezifizierbar. Dies sollte vielleicht durch ausführliche Tests, die im Rahmen dieser Arbeit nicht mehr möglich waren, präzisiert werden. Eine weitere Möglichkeit, ein Verfahren konkret an einen bestimmten Kontext anzupassen, wäre die Kontrolle über einen Kontextmanager, der laut [Ber99] Bestandteil einer mobilen DBS-Architektur sein sollte. In Verbindung mit diesem Manager könnte dann eine angemessene Festlegung und Spezifizierung für **d** abhängig von der jeweiligen Anwendung erfolgen.

Soweit seien nun die noch bestehenden Probleme bzw. Aufgaben für den Einsatz eines Replikationsverfahrens in mobilen Umgebungen genannt. Im Nächsten Kapitel erfolgt nun eine Zusammenfassung der vorliegenden Arbeit.

10 Zusammenfassung

Im dieser Arbeit wurden die Einflüsse mobiler Kontexte auf die Replikationsverwaltung in mobilen Umgebungen untersucht. Dafür wurde zunächst ein Überblick über klassische Replikationsverfahren in verteilten Systemen gegeben sowie der Begriff mobiler Kontext näher erläutert. Nach einer Beschreibung des im Rahmen dieser Arbeit verwendeten Anwendungsszenarios der Literaturdatenbanken erfolgte in Kapitel 5 ein konkrete Beschreibung, wie sich mobile Kontexte auf die Auswahl eines Replikationszieles auswirken bzw. welche Ziele verwirklicht werden müssen. Dies sind im Einzelnen :

- Erhöhung der Verfügbarkeit
- Wahrung der Konsistenz
- Minimierung der Kommunikationskosten

Ebenfalls in Kapitel 5 erfolgte eine Bewertung der bereits vorgestellten klassischen Replikationsverfahren hinsichtlich der Erfüllung der drei oben genannten Ziele. Dabei wurde festgestellt, daß kein herkömmliches Verfahren die Erfüllung aller drei Ziele für mobile Replikation gewährleisten kann. Als Lösungsvorschlag wurde der Einsatz hybrider Verfahren unter Einbindung von z.B. anwendungs-spezifischem Wissen genannt. Daraufhin erfolgte in Kapitel 6 eine Vorstellung bereits existierender Replikationsverfahren für mobile Umgebungen, die ebenfalls bezüglich der Erfüllung der drei Replikationsziele für mobile Umgebungen bewertet wurden. Im Anschluß daran wurde eines dieser Verfahren hinsichtlich der Weiterverwendung innerhalb des in dieser Arbeit betrachteten Anwendungsszenarios ausgewählt. Aufgrund seiner einfachen Strukturierung und den dynamischen Veränderungsmöglichkeiten zur Erfüllung unterschiedlicher Replikationsziele durch die Variierung des Konsistenzparameters **d** entschied ich mich zur Verwendung des in

[PiBha95,Pit96] vorgestellten Verfahrens. In Kapitel 7 erfolgte dann zunächst eine Abgrenzung dieses Verfahrens im Bezug auf das verwendete Anwendungsszenario. Dabei wurde u.a. festgestellt, daß nicht alle Definitionen für **d** auf die Literaturdaten anwendbar sind. Bezüglich der Anwendung wurden weitere Fragen für den Einsatz des Verfahrens beantwortet, wie die Größe eines einzelnen Clusters (ein MH), die Aufgabenverteilung innerhalb des Verfahrens sowie die Reaktion des Verfahrens auf unterschiedliche geforderte Replikationsziele. Als Ergebnis wurde ein Server-basiertes Verfahren entwickelt, das durch seine individuellen Gestaltungsmöglichkeiten für den Konsistenzparameter **d** und die Festlegung der Clustergröße auf einen MH die Anforderungen der unterschiedlichen Replikationsziele gewährleisten kann. Im Anschluß daran erfolgte in Kapitel 8 eine Implementationsbeschreibung für dieses Verfahren, bevor in Kapitel 9 eine Einordnung des Verfahrens in typische mobile Replikations-szenarien vorgenommen wurde. Da das in Kapitel 7 vorgestellte Verfahren in einer eher atypischen Netzstruktur seine Anwendung fand, ergaben sich bei der Einordnung in Kapitel 9 einige neue Fragen bzgl. der Clustergröße und der Definition von **d**. Diese Fragen konnten nicht vollständig beantwortet werden, so daß ebenfalls in Kapitel 9 eine Problembeschreibung für zukünftige Arbeiten erfolgte. Der interessanteste Punkt dabei ist für mich die direkte Einbindung eines Kontextmanagers in das Replikationsverfahren, da dadurch die direkte Einflußnahme des jeweiligen mobilen Kontextes auf die Reaktion des Replikationsverfahrens bezüglich des geforderten Replikationszieles meiner Meinung nach stark verbessert werden kann.

Literaturverzeichnis

[BaIm92] Badinrath, B.R. , Imielinski, T. : „Replication and Mobility“ In : Proceedings of the Second Workshop on the Management of Replicated Data, 1992

[BeDa95] Beuter, T. , Dadam, P. : „Prinzipien der Replikationskontrolle in verteilten Systemen“ Ulmer Informatik-Berichte, 1995

[Ber98] Berger, Benedikt : „Auswahl und Weiterentwicklung eines Replikationsverfahrens für den mobilen Datenbankzugriff im Projekt MoVi“, Studienarbeit, Universität Rostock, 1998

[Ber99] Berger, Benedikt : „Entwicklung einer DBS - Architektur für mobile Verarbeitungsumgebungen“, Diplomarbeit, Universität Rostock, 1999

[BGM94] Barbará, D. , Garcia-Molina, H. : „Replicated data management in mobile environments : Anything new under the sun?“ In : IFIP Conference on Applications in parallel and Distributed Computing, 1994

[FaZa95] Faiz, M. , Zaslavsky, A. : „Database Replica Management Startegies in Multidatabase Systems with Mobile Hosts“ In: 6th International Hong Kong Computer Society Database Workshop : Database Reengeneering and Interoperability, 1995

[FaZaSri95] Faiz, M. , Zaslavsky, A. , Srinivasan, B. : „Revising Replication Strategies for mobile computing Environments“ In : ECOOP´95 Workshop on Mobility and Replication, 1995

[HeLu99] Heuer, A. , Lubinski, A. : „Configured Replication for Mobile Applications“, Rostocker Informatik-berichte, Nr. 24, 1999

[HeSa95] Heuer, A. , Saake, G. : „Datenbanken Konzepte und Sprachen“, International Thomson Publishing Company, 1995

[HeSa99] Heuer, A. , Saake, G. : „Datenbanken Implementie-rungstechniken“, MIPT-Verlag, 1999

[ImKo96] Imielinski, T. , Korth, H.F. : „Mobile Computing“, Kluwer Academic Publishers, 1996

[Kot95] Kottmann, D. A. : „Serializing Operations into the Past and Future : A Paradigm for Disconnected Operations on Replicated Objects“ In : ECOOP´95 Workshop on Mobility and Replication, 1995

[LuSa94] Lu, Q. , Satyanaranyanan, M. : „Isolation–Only Tranactions for mobile Computing“ In : ACM Operating Systems Review, 1994

[LuSa95] Lu, Q. , Satyanarayanan, M. : „Improving Data Consistency in Mobile Computing Using Isolation – Only Transactions“ In : Proceedings of the 5th Workshop on Hot Topics in Operating Systems, 1995

[PiBha95] Pitoura, E. , Bhargava, B. : „Maintaining Consistency of Data in Mobile Distributed Environments“ In : Proceedings of the 15th International Conference on Distributing Computing Systems, 1995

[Pit96] Pitoura, E. : „A Replication Schema to Support Weak Connectivity in Mobile Information Systems“ In : 7th International Conference on Database and Expert Systems Application (DEXA), 1996

[Pit98a] Pitoura, E. : „Scalable Invalidation – Based Processing of Queries in Broadcast Push Delivery“, In : 17th Workshop on Mobile Data Access,1998

[Pit98b] Pitoura, E. : „Supporting Read - Only Transactions in Wireless Broadcasting“ In : Proceedings of the DEXA 98 International Workshop on Mobility in Databases and Distributed Systems, 1998

[Sar97] Sarodnik, Axel : „Evaluierung und Bewertung von Replikationstechniken in verteilten Datenbank-systemen“, Diplomarbeit, Universität Rostock, 1997

[Schrö97] Schröder, Torsten : „Caching in mobilen Datenbank-systemen“, Diplomarbeit, Universität Oldenburg, 1997

[TaDu91] Tait, C.D. , Duchamp, D. : „Service Interface and Replica Managment Algorithm for mobile File System Clients“, In : Proceeding of the parallel and distributed information systems conference, 1991

[WaPa95] Wang, Q.R. , Pâris, J.-F. : „Managing Replicated Data Using Referees“ In : ECOOP´95 Workshop on Mobility and Replication, 1995

[Wat96] Waterstraat, Jörn : „Definition und Nutzung von Kontexten in mobiler, verteilter Datenbank-Umgebung“, Diplomarbeit, Universität Rostock, 1996

[Web98] Weber, Gunnar : „Integration von Literatur-datenbanken über das WWW“, Diplomarbeit, Universität Rostock, 1998

[Zuk97] Zukunft, O. : „Integration mobiler und aktiver Datenbankmechanismen als Basis für die orts-gebundene Vorgangsbearbeitung“, Dissertation, Universität Oldenburg, 1997

Abbildungsverzeichnis

Tabellenverzeichnis

Thesen

1. Mobile Kontexte können, je nach Ausprägung, eine Erfüllung aller möglichen Replikationsziele für mobile Umgebung erfordern, was sich in einer Gewichtsverschiebung zum jeweiligen Ziel äußert. Dabei kann es zwischen den Anforderungen unterschiedlicher Kontexte bezüglich des geforderten Replikationsziels zu Konflikten kommen.

2. Für mobile Systeme wird das Ziel der Minimierung des Änderungsaufwandes einer Operation auf die Minimierung der Kommunikationskosten beschränkt.

3. Klassische Replikationsverfahren sind in ihrer reinen Form nicht für den Einsatz in mobilen Replikationsszenarien geeignet. Eine Lösung wäre der Einsatz hybrider Verfahren unter Anreicherung des Verfahrens mit z.B. anwendungsspezifischem Wissen.

4. Die VPC-Methode ist nicht für den Einsatz in mobilen Szenarien geeignet, in denen eine lokale erhöhte Verfügbarkeit von Daten aufgrund des Anwendungskontextes unbedingt von Nöten ist.

5. Aufgrund seiner Möglichkeiten zur individuellen Gestaltung bezüglich des Anwendungskontextes ist das in Abschnitt 6.3 beschriebene Verfahren sehr gut für den Einsatz in mobilen Systemen geignet.

6. Durch die Veränderung des Konsistenzparameters **d** kann eine Wichtungsverschiebung zwischen verschiedenen Replikationszielen aufgefangen werden.

7. Der Einsatz bzw. die Festlegung einzelner Verfahrensparameter, wie z.B. der Konsistenzparameter d oder die Größe eines Clusters, sind sehr stark vom jeweiligen Anwendungskontext bzw. der vorhandenen Netzstruktur abhängig.