Auswahl und Weiterentwicklung eines Replikationsverfahrens für den mobilen Datenbankzugriff im Projekt MoVi

Studienarbeit

Universität Rostock, Fachbereich Informatik

vorgelegt von Berger, Benedikt
geboren am 24.05.1974 in Rostock

Betreuer: Prof. Dr. Andreas Heuer
Dipl. Ing. Astrid Lubinski

Zusammenfassung

Diese Studienarbeit versucht, aus der Vielzahl existierender Replikationsverfahren ein für den mobilen Einsatz geeignetes Verfahren auszuwählen und zu verbessern.

Inhaltsverzeichnis

Teil I

Einführung und Problemstellung

Kapitel 1

Einleitung

Dieses Kapitel umfaßt eine Einführung in das MoVi-Projekt, innerhalb dessen diese Studienarbeit entstand, die Grundlagen zum Thema Replikation und Mobilität sowie eine kurze Gliederung der folgenden Kapitel. Dazu werden zunächst die Begriffe *Mobilität, VDBMS* und *Replikation* erklärt und ihre Bedeutung für diese Studienarbeit veranschaulicht.

1.1 Das MoVi-Projekt

Diese Studienarbeit entstand im Rahmen des Projektes Mobile Visualisierung (MoVi), in dem eine Visualisierungsarchitektur für ein global verteiltes Informationssystem geschaffen werden soll, das den mobilen Zugang zu allen denkbaren Daten und Diensten ermöglicht. Dabei sollen dem Benutzer die Informationen an seinen mobilen Kontext angepaßt werden. Der mobile Kontext beschreibt die speziellen Charakteristika der Arbeit eines Benutzers (z.B. mit Datenbanksystemen) in mobilen Umgebungen. Dabei treten folgende Problemaspekte auf:

- Konsistenz replizierter, dynamischer Daten
- Reduktion von Datenmengen
- erhöhte Sicherheitsrisiken
- ortsabhängige Zugriffe und andere umgebungsspezifische Anwendungen
- resourcenabhängige Optimierung

Hier soll insbesondere der erste Problemkreis betrachtet werden. Die speziellen Anforderungen des Projektes MoVi setzen den Möglichkeiten der Realisierung von Replikation in einem VDBMS enge Grenzen. Das Problem besteht allgemein beim Aufbau eines VDBMS, das unter anderem auch aus mobilen Rechnern bestehen und in welchem Replikationsverfahren die Verfügbarkeit erhöhen und die Netzbelastung für Anfragen und Anfrageergebnisse verringern sollen.

Die Aufgabe dieser Studienarbeit besteht darin, die bereits aus der Literatur bekannten Verfahren zur Replikation in Verteilten Datenbankmanagementsystemen unter dem Aspekt eines mobilen Szenariums zu bewerten und ein Verfahren für ausgewählte Bereiche dieses Szenariums anzupassen. Als Datenbankmanagementsysteme werden hier nur solche betrachtet, die auf dem Relationenmodell basieren, was aber für die Arbeitsweise der Replikationsverfahren an sich keine Einschränkung darstellt.

1.2 Grundlagen

Im folgenden werden die Begriffe *Mobile Arbeit, VDBMS* und *Replikation* kurz erklärt und ihre Bedeutung für diese Studienarbeit erläutert.

1.2.1 Merkmale mobiler Arbeit

Die Anwendungsbandbreite von mobilen Rechnern, die über drahtlose Medien mit Hostsystemen und den darauf laufenden Datenbanksystemen kommunizieren können, ist sehr umfassend und hardwareseitig schon heute realisierbar. Allerdings stellt die Arbeit an mobilen Rechnern der Verwirklichung von verteilten Anwendungen einige Hindernisse in den Weg, die der Mobilität und der mobilen Kommunikation unter den heutigen technologischen Bedingungen angeboren sind. Besondere Grundmerkmale der mobilen Arbeit sind nach [Sa96]:

1. Mobile Elemente sind weniger leistungsfähig als statische Elemente.
2. Mobile Verbindungen variieren stark in Bandbreite und Zuverlässigkeit.
3. Mobile Elemente haben nur eine begrenzte Energiequelle.
4. Mobilität ist naturgemäß gefährdet (in Bezug auf das Entwenden des Mobilrechners)

Hinzu kommen folgende zusätzliche Merkmale auf die Arbeitsweise eines mobilen Nutzers:

1. Die übertragbaren Datenmengen sind durch Zeitbegrenzungen seitens des Nutzers und seiner mobilen Arbeit sowie durch die niedrigen Datenübertragungsraten in ihrem Umfang begrenzt.
2. Die Nutzung mobiler Verbindungseinrichtungen erzeugt hohe Kosten.
3. Der mobile Nutzer ist aus diesen Gründen selten mit dem Festnetz verbunden.
4. Wenn der mobile Nutzer mit dem Festnetz verbunden ist, dann meist nur für kurze Zeit.

Aus diesen Punkten läßt sich bereits erkennen, daß die Konsistenzsicherung mehrerer Datenbestände untereinander aufgrund von langsamen oder häufig nicht verfügbaren Verbindungen stark erschwert wird. Bei der Betrachtung der Auswirkungen mobiler Arbeit im Abschnitt 4 wird sich diese Problematik als die bedeutendste innerhalb eines verteilten, replizierten und mobilen Datenbankmanagementsystems erweisen.

1.2.2 VDBMS

Zunächst sollen einige Begriffe für den Einsatz von Datenbanken in verteilten Systemen kurz dargelegt werden.

Definition 1 ***Verteilte Datenbank***
Eine Verteilte Datenbank ist eine Menge von mehreren, logisch untereinander verknüpften Datenbanken, die über ein Computernetzwerk verteilt sind [ÖV91].

Ein VDBMS setzt ein funktionsfähiges Computernetzwerk voraus. Das bedeutet, das die Verfügbarkeit eines VDBMS im Gegensatz zu der eines einfachen DBMS nicht nur von der Verfügbarkeit und Fehlertoleranz einen Rechners und des darauf eingerichteten Betriebssystems, sondern zusätzlich stark von der Verfügbarkeit des Netzwerkes abhängt. Deshalb beeinflussen Netzwerkfehler die Funktionsweise eines VDBMS besonders stark.

Definition 2 ***VDBMS***

Ein verteiltes Datenbankmanagementsystem ist die Software, die die Verwaltung der verteilten Datenbanken erlaubt und die Verteilung für den Benutzer transparent macht [ÖV91].

Für die Realisierung eines Verteilten DBMS existieren mehrere Möglichkeiten.

1. Einfach-VDBMS (eine globale Sicht auf die verteilten Datenbanken)
2. Multi-DBMS (mehrere globale Sichten auf die verteilten Datenbanken)
3. Föderiertes DBMS (wie Multi-DBMS, aber die verteilten Datenbanken haben einen höheren Grad an Autonomie)

Die Granularität der Verteilung hat prinzipiell mehrere mögliche Stufen, hier soll nur die Verteilung von einzelnen Relationenschemata betrachtet werden. Die Verteilung der Datenbankschemainformationen wird unterhalb der globalen konzeptuellen Ebene definiert. Dabei wird im *Fragmentationsschema* die Zerlegung eines Relationenschemas

- horizontal (durch Selektion)
- abgeleitet horizontal (durch Selektion mit einem Fremdschlüssel)
- vertikal (durch Projektion)
- gemischt (horizontal und vertikal)

beschrieben. Die dadurch entstehenden *Fragmente* bilden disjunkte Teilrelationen. Im *Allokationsschema* wird festgelegt, auf welchen Rechnerknoten diese Fragmente abgelegt und verwaltet werden sollen. Diese Rechnerknoten sind dann für die Ausführung von Transaktionen, die Sperrverwaltung, die Konsistenzsicherung und die Anfragebearbeitung auf dieser lokalen Instanz des Fragmentes verantwortlich.

Die *Verfügbarkeit* eines VDBMS ist als Gesamtsystem höher als die eines zentralen Systems. Zwar ist die Ausfallwahrscheinlichkeit eines Rechnerknotens des VDBMS höher als die eines zentralen Rechnersystems, wodurch ohne Replikation die Verfügbarkeit eines Fragmentes geringer ist. Dagegen aber steht die **graceful degradation** im Falles des Ausfalles eines Rechnerknotens, d.h. ein Arbeiten mit dem System ist in eingeschränktem Maße auch nach Ausfall einer (oder mehrerer) Komponenten möglich, im Gegensatz zu zentralen Systemen. Um die Verfügbarkeit zu maximieren, ist es in vielen Fällen sinnvoll, nicht nur eine Instanz eines Fragmentes auf einem Rechnerknoten zuzulassen, sondern mehrere Instanzen räumlich und auf unterschiedlichen Rechnerknoten zu verteilen, so daß im Extremfall jeder Rechnerknoten Kopien alle möglichen Instanzen halten darf. Diese Vorgehensweise nennt man Replikation.

1.2.3 Replikation in VDBMS

Um Daten in VDBMS replizieren zu können, müssen sie als Fragmente vorliegen. Im Allokationsschema ist es dann möglich, ein Fragment auf mehrere Rechnerknoten zu verteilen. Dabei kommt als neue Aufgabe die wechselseitige Konsistenzsicherung der lokalen Kopien untereinander hinzu. Die anderen Aufgaben (z.B. Sperrverwaltung) werden komplexer, da die replikathaltenden Rechner auch noch untereinander kommunizieren müssen, um Lese- und Schreibzugriffe auf den lokalen Kopien zu synchronisieren.

Definition 3 ***Replikat***

Ein ***Replikat*** *ist eine von mehreren Instantiierungen eines Fragmentes.*

Im Bereich der verteilten Datenbanken muß der Begriff der Serialisierbarkeit auf den Einsatz von Replikationsverfahren ausgeweitet werden :

Definition 4 ***1-Kopie-Serialisierbarkeit***

Ein Schedule S einer replizierten Datenbank heißt ***1-Kopie-serialisierbar****, wenn es mindestens eine serielle Ausführung der Transaktionen dieses Schedules auf einer nicht replizierten Datenbank gibt, welche die gleiche Ausgabe sowie den gleichen Datenbankzustand wie S auf der replizierten Datenbank erzeugt (nach [AS97])*

Das bedeutet, für eine Anwendung soll sich eine replizierte Datenbank genauso verhalten wie eine nicht replizierte Datenbank (*Allokationstransparenz*). Die Algorithmen, die für die wechselseitige Konsistenz der Kopien sorgen, nennt man Replikationsverfahren. In festnetzbasierten VDBMS soll der Einsatz von Replikation die Verfügbarkeit des Gesamtsystems durch Redundanz der verteilten Daten und die Verarbeitungseffizienz durch die Lokalität der benötigten Daten auf einem Knoten erhöhen. Dadurch wird der Schwerpunkt der Kommunikation auf das Propagieren von lokal geänderten Daten und die Sperrverwaltung verschoben. Das bedeutet, daß die Netzlast um so größer wird, je mehr Änderungsoperationen ausgeführt werden. Analog dazu werden die sinkenden Anforderungen an die nötige Verfügbarkeit der replikathaltenden Knoten durch höhere Anforderungen an die Verfügbarkeit der Verbindungseinrichtungen erreicht. Um Replikate optimal einsetzen zu können, müssen bei der Festlegung des Replikationsschemas bereits die voraussichtlichen Read-/Write-Charakteristika der einzelnen Knoten auf den Fragmenten bekannt sein. Eine Ausnahme bilden die dynamischen Replikationsverfahren, die sich während der Laufzeit dem Optimum der Anzahl und Verteilung der Replikate anpassen. Somit ergeben sich folgende Vor- und Nachteile beim Einsatz von Replikation in VDBMS:

- Vorteile :
  - geringe Kommunikationskosten für Anfragen
  - hohe Effizienz bei der Arbeit mit dem Replikat
  - hohe Verfügbarkeit bei sicheren Verbindungseinrichtungen
- Nachteile :
  - hohe Kommunikationkosten für Updates
  - geringe Verfügbarkeit (Bandbreite, Lifelocks) bei unsicheren Verbindungseinrichtungen

Die Widersprüche zwischen den oben genannten Vor- und Nachteilen ergeben sich aus einer allgemeinen Betrachtung der breiten Möglichkeiten von Replikation, die stark vom Verfahren und dessen Implementierung abhängen. Ein Verfahren, welches z.B. die Verfügbarkeit durch eine große Anzahl von Replikaten eines Fragmentes erhöht, wird unter Umständen Schwierigkeiten mit der ständigen Aktualisierung der Replikate und der Effizienz der Sperrverwaltung über das Netzwerk haben, wenn die Bandbreite des Netzwerkes nicht auf die Leistungsfähigkeit der Rechner abgestimmt ist. Da ein VDBMS relativ große Netzlast erzeugt, werden damit andere Netzwerk-Applikation und -Dienste in ihrer Effizienz eingeschränkt oder das Netzwerk wird überlastet.

Um Replikationsverfahren einteilen zu können, werden sie nach der Art und Weise der Konsistenzsicherung und der Natur der Informationen, auf denen das Verfahren seine Konsistenzsicherungsstrategie aufbaut, unterschieden.

Pessimistische / optimistische Verfahren

Man kann die existierenden Replikationsverfahren entsprechend der Strategie für Konsistenzsicherung zum einen in *pessimistische* und zum anderen in *optimistische* Verfahren einteilen.

Die *pessimistischen Verfahren* gehen von der Grundannahme aus, daß im Fehlerfall ein "Auseinanderlaufen"der Replikate sehr wahrscheinlich und eine Reintegration relativ komplex, aufwendig und nicht immer vollständig möglich ist. Daher werden im Fehlerfall die Zugriffsmöglichkeiten auf die Replikate so stark eingeschränkt, daß es zu keinen Differenzen kommen sollte. Damit ist in der Reintegrationsphase nur ein Nachvollziehen der Transaktionen während des Fehlers nötig, was problemlos automatisiert werden kann.

Den pessimistischen Verfahren stehen die *optimistischen Verfahren* gegenüber. Diese Verfahren gehen davon aus, daß ein Auseinanderlaufen der Replikate unwahrscheinlich, selten bzw. relativ leicht behebbar ist. Somit ist in der Reintegrationsphase (nach der Wiederverbindung mit den anderen Rechnerknoten) die Abarbeitung eines Logs notwendig, um alle auf einem Fragment ausgeführten Transaktionen zu vergleichen und, wenn nötig, zurückzusetzen.

Syntaktische / Semantische Verfahren

Eine andere Einteilung von Replikationsverfahren berücksichtigt die Art der Informationen, aufgrund derer die Entscheidungen über die Zugriffe auf Replikate getroffen werden. Bei den *syntaktischen Verfahren* werden zur Entscheidungsfindung die Reihenfolge der Transaktionen und die potentielle Möglichkeit, daß sich zwei Zugriffe auf demselben Datenobjekt gegenseitig beeinflussen (read-write-Konflikte, write-write-Konflikte), genutzt. Aufgrund der Nutzung syntaktischen Wissens sind diese Verfahren unabhängig von den Anwendungen und somit universell einsetzbar.

Dagegen wird bei den *semantischen Verfahren* Wissen über den Zweck von Transaktionen genutzt, um die Zugriffe konfliktfrei ablaufen zu lassen. So ergeben sich bei arithmetischen Transaktionen große Optimierungspotentiale, z.B. durch die Kommutativität solcher Transaktionen. Diese Art von Verfahren versagt für gewöhnlich bei Transaktionen, über deren Sinn kein Wissen vorliegt. Deshalb sind die semantischen Verfahren meist nur in speziellen Umgebungen, in denen ausschließlich Transaktionen mit bekannter Semantik erzeugt und verarbeitet werden, einsetzbar.

1.3 Gliederung

Im folgenden Kapitel 2 wird zunächst ein mögliches mobiles Szenarium abstrakt vorgestellt. Es folgt eine Anwendung dieses Szenariums auf dem Gebiet der Walddatenverwaltung in Kapitel 3, in welchem das begleitenden Beispiel der Waldzustandsdaten vorgestellt werden wird. Kapitel 4 enthält die Zusammenfassung der Problemstellung, bezogen auf das mobile Szenarium und Replikation. In Kapitel 5 wird ein Klassifikationsschema vorgestellt, mit dessen Hilfe eine ungefähre Einordnung und Vergleich der im folgenden Kapitel 6 vorgestellten Replikationsverfahren in Bezug auf die Anforderungen mobiler Arbeit erfolgen kann, wie in Kapitel 7 dargestellt wird. In Kapitel 8 werden die vorgestellten Verfahren anhand der Kriterien des mobilen Szenariums untersucht und bewertet. Aus diesen Möglichkeiten wird in Kapitel 9 eine ausgewählt und erweitert, die dann hinsichtlich der in Kapitel 4 geforderten Eigenschaften bewertet wird. Ein Simulationssystem für Replikationsvefahren wird in Kapitel 10 dargestellt. Kapitel 10.7 bildet mit einer Zusammenfassung und einem Überblick auf noch zu behandelnde Themen den Abschluß.

Kapitel 2

Ein mobiles Szenarium

In diesem Kapitel sollen die Grundlagen für den Aufbau von Funknetzen, soweit für die mobile Arbeit nötig, sowie eine verallgemeinerte Netzstruktur und die Charakteristiken der wesentlichen Bestandteile eines Funknetzes für den mobilen Einsatz vorgestellt werden.

2.1 Zellularnetze

Das hier vorgestellte Szenarium basiert auf der physikalischen Struktur von Funkzellnetzen [IB92a]. Dabei sind einzelne Rechnerknoten, die home base nodes (HBN), für die Versorgung einer Funkzelle verantwortlich. Innerhalb dieser Zellen befinden sich die mobilen Rechner. Die HBN stellen die Verbindung von allen in ihrer Funkzelle befindlichen mobilen Geräten mit einem Festnetz her (Abb.2.1).

2.2 Struktur des Gesamtnetzes

Für die weiteren Betrachtungen des mobilen Szenariums sollen zunächst die Eigenschaften der verwendeten Rechnerknoten und der Verbindungselemente spezifiziert werden. Zur Charakterisierung der Rechner und Verbindungseinrichtungen sollen folgende Kriterien benutzt werden :

- Rechnerknoten
  - Rechenleistung
  - Primärspeicher
  - Sekundärspeicher
  - Stromversorgung
  - Verfügbarkeit(technische Sicherheit)
- Verbindungseinrichtungen
  - Datenraten
  - Kosten
  - Verfügbarkeit(technische Sicherheit)

Ein mobiles Szenarium (Abb. 2.1) besteht somit meist aus einer Menge von Rechnern, die sich stark in Rechenleistung, Primär- und Sekundärspeichergröße sowie

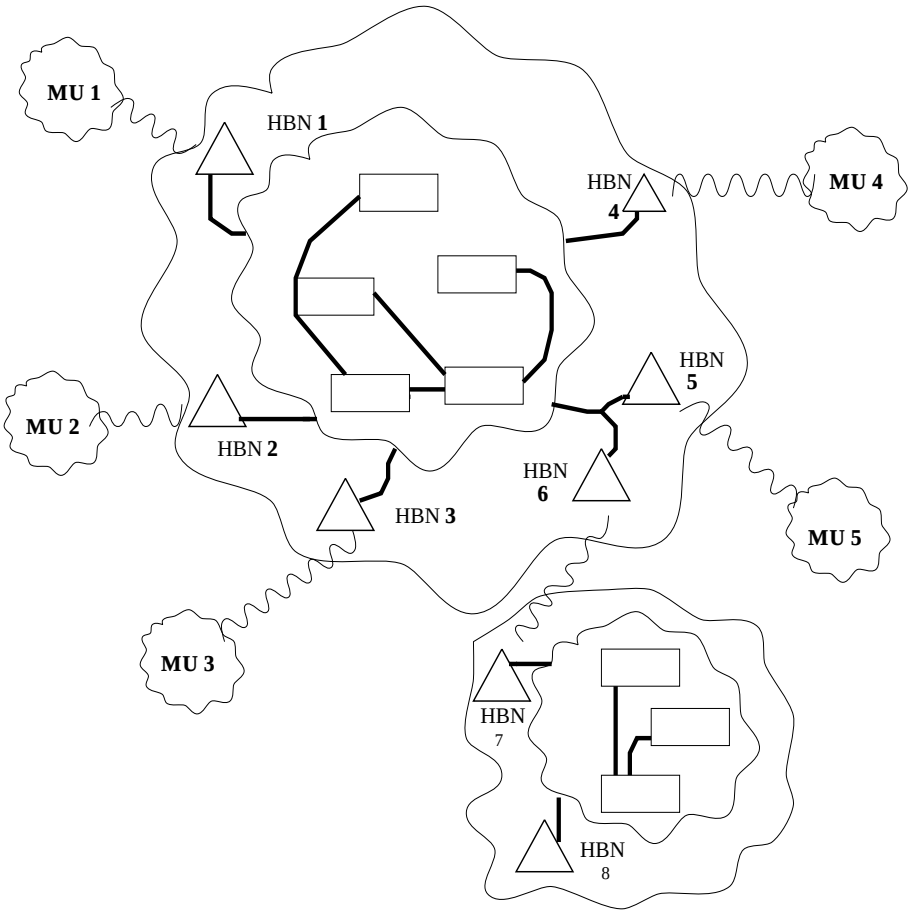

Abbildung 2.1: Mobiles Szenarium

Verfügbarkeit unterscheiden, und aus Verbindungseinrichtungen, die sich in ihrer Verfügbarkeit und Effizienz (Verbindungsstörungen, geringe Datenraten) unterscheiden. Die Rechnerknoten und die Verbindungen teilen sich jeweils in 2 Klassen auf:

1. schnelle, sichere Rechner/Verbindungseinrichtungen (Workstation / LAN)
2. langsame, unsichere Rechner/Verbindungseinrichtungen (PDA / Funknetz)

Darauf aufbauend arbeitet ein VDBMS, das mit Hilfe von Replikationsverfahren die Verfügbarkeit von ausgewählten Daten erhöhen soll. Um die praktischen Aspekte einer solchen mobilen Arbeitsumgebung zu verdeutlichen, wird im folgenden das Walddatenszenarium vorgestellt.

Kapitel 3

Das Beispiel des Walddatenszenariums

Die folgenden Abschnitte dieses Kapitels beschreiben ein Szenarium, welches ein für mobile Arbeit mögliches Anwendungsbeispiel darstellt. Es basiert auf einem Teil der im Rahmen des MoVi-Projektes im Zusammenhang mit dem Waldzustandsbericht erfaßten Daten und dem Waldschutzkontrollbuch.

3.1 Einführung

Als begleitendes praktisches Beispiel soll hier die Erfassung von Waldzustandsdaten mit mobilen Rechnern dargestellt werden. Die Waldgebiete sind in Zellen eingeteilt, für die jeweils ein Förster zuständig ist. (siehe Abb. 3.1). Die Daten betreffen hauptsächlich den Schädlingsbefall verschiedener Arten und Schäden durch Wild sowie Wetter- und Klimaauswirkungen. Sie werden in Intervallen von 1-2 Monaten erhoben und in ein Waldschutzkontrollbuch eingetragen. Hinzu kommen Daten, die über längere Zeiträume gesammelt werden (Statistik über 2 Jahre). Für jedes Forstrevier existiert einWaldschutzkontrollbuch, das nach zwei Jahren an das übergeordnete Forstamt geschickt wird. Dort werden die Walddaten auf Disketten übertragen, welche dann an das Forstministerium transportiert und dort zusammengefaßt werden. Die Gliederung der Forstbehörde ist also hierarchisch und ortsabhängig. Somit sind auch die Daten der Forstreviere überschneidungsfrei. Jedes Gebiet, für das ein Revierförster zuständig ist, ist in Abteilungen, Unterabteilungen und Teilflächen, wie 182c, 181e, 188b (Abb. 3.2) unterteilt. Allerdings kann man erkennen, daß sich die Struktur der Zellen nicht mit den botanischen Formationen deckt. Wie man auf der Abbildung erkennen kann, wurde das Problem zunächst durch Zerteilung der zellüberschreitenden Strukturen (den Teilflächen) gelöst. Beim Einsatz mobiler Erfassungsgeräte bietet es sich an, die Erfassung entsprechend den natürlichen vorgegebenen Strukturen durchzuführen, um die Arbeit des mobilen Nutzers zu erleichtern. Dabei würden sich die Arbeiten in den Revieren überschneiden.

Im folgenden soll nun ein ER-Schema vorgestellt werden, daß die Struktur der wesentlichen Informationen über das Waldschutzkontrollbuches grob modellieren soll.

3.2 Das ER-Schema des Walddatenszenariums

Im Waldschutzkontrollbuch werden monatlich die Schäden am und im Wald protokolliert, die für diesen Monat typisch sind. Hinzu kommen einige Schadarten,

Forstdirektion
Forstamt
Revier
Abteilung
Unterabteilung
Teilfläche

Abbildung 3.1: Struktur der Waldhierarchie

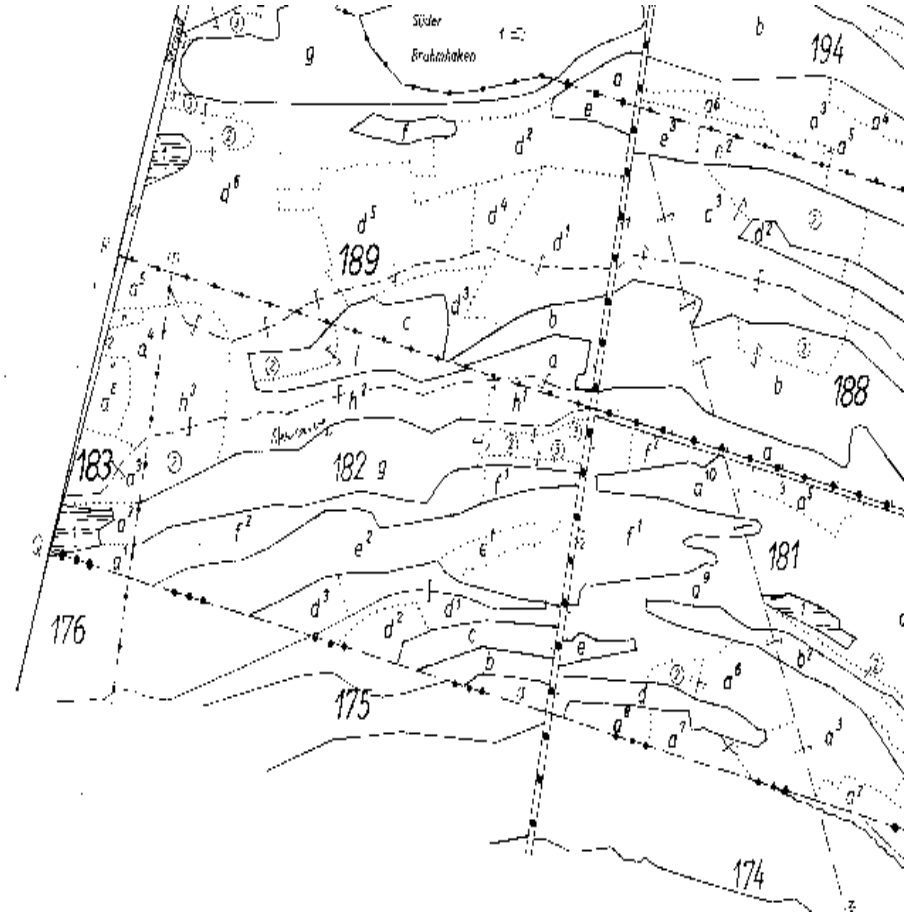

Abbildung 3.2: Ausschnitt einer Forstkarte

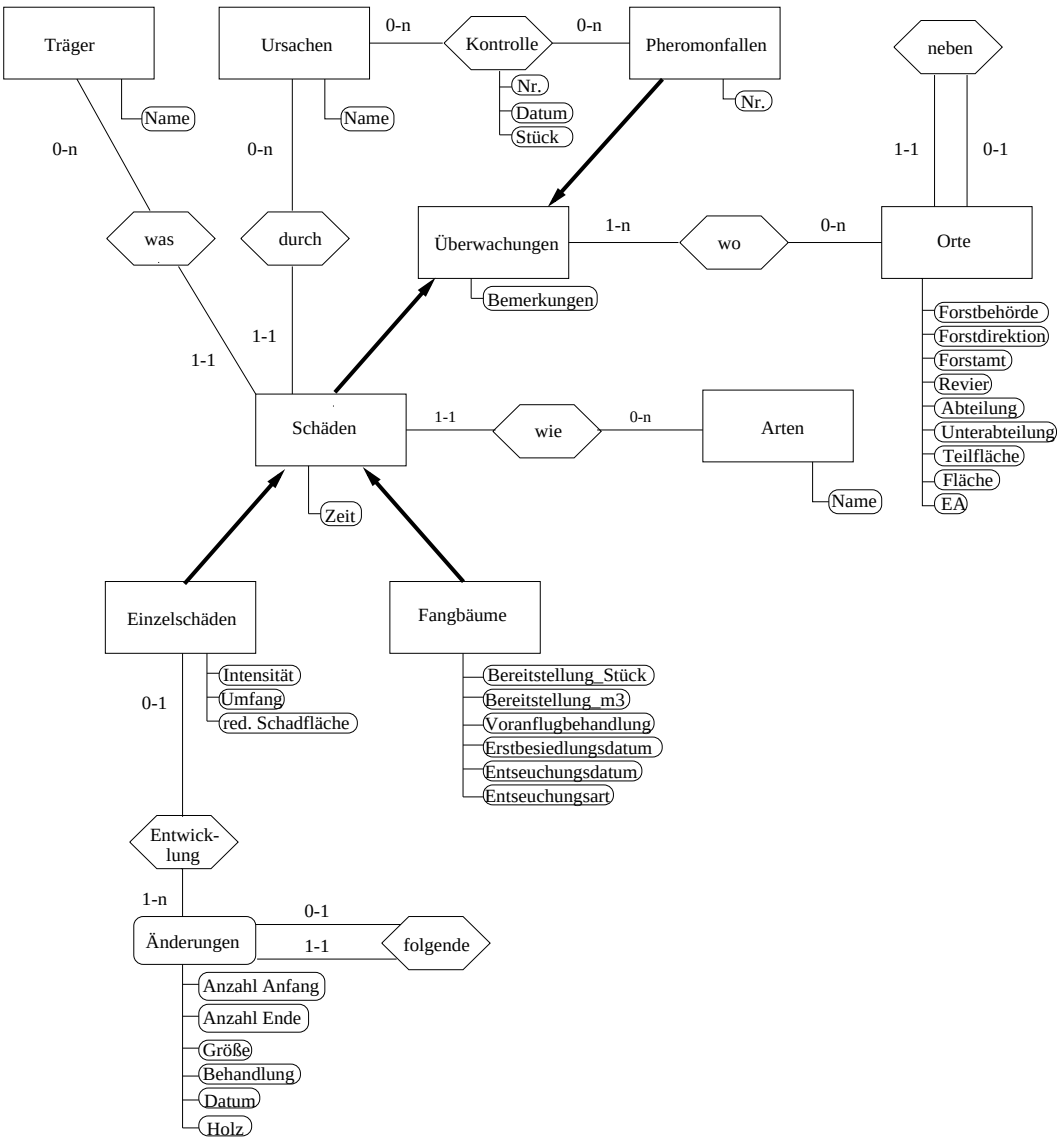

Abbildung 3.3: ER-Schema für Waldszenarium

die das ganze Jahr hindurch auftreten können. Um das ER-Schema übersichtlich halten zu können, wurden diese Unterteilungen aufgehoben. Somit kann das ER-Schema (Abb. 3.3) auch Daten aufnehmen, die im Waldschutzkontrollbuch nicht vorgesehen waren. Ähnlich wurden die Schadensinformationen von so unterschiedlichen Arten wie Mäusen, rindenbrütenden Insekten und Wetter behandelt. Damit ergibt sich als eine zentrale Entität die **Überwachungen**. Sie faßt alle zu erfassenden Daten zusammen und ist zunächst nur durch einen Ort gekennzeichnet. Diese Eigenschaft wird an die Entitäten **Pheromonfallen** und **Schäden** weitergegeben. Die **Schäden** als der zweite zentrale Bestandteil des ER-Schemas sind durch eine erkennbare **Ursache**, einen Schadensträger(**Träger**) und die Schadensart(**Art**) gekennzeichnet. Zum Beispiel kann der Buchdrucker Fraßschäden an Laubbäumen verursachen. Außerdem wird einem Schaden auch immer ein Zeitpunkt der Erkennung zugeordnet. Hinzu kommt, daß ein Schaden eine zeitliche **Entwicklung** haben kann: einerseits verändert sich der natürliche Schadensumfang mit der Zeit, andererseits sollen Maßnahmen wie das Abholzen befallener Bäume ein weiteres Ausbreiten des Schadens verhindern und das Ausmaß des Schadens reduzieren. Zusätzlich zu Schadinformationen beinhaltet das Waldschutzkontrollbuch Beobachtungsdaten von **Fangbäumen** und **Pheromonfallen**. Da diese Daten auch ortsabhängig sind, wurden sie mit in das ER-Schema aufgenommen. Die **Fangbäume** kann man auch als eine Art Schaden betrachten, deshalb wurden sie neben den Einzelschäden als **Schäden** modelliert. Die **Pheromonfallen** sind als **Überwachungen** auch ortsabhangig, haben aber nicht die Charakteristik von Schäden. Deshalb wurden sie als Spezialisierung von **Überwachungen** modelliert, die zusätzlich eine Verbindung zu der **Ursachen**/Schädlings-Entität hat. Die Entität **Überwachung** und alle ihre Spezialisierungen sind direkt mit einem oder mehreren **Orten** verknüpft. Die **Orte** sind Teilflächen, die die kleinsten Flächenelemente im Wald darstellen. Die Ordnung der Teilflächen wird durch die Ortshierarchie, bestehend aus der Forstbehörde, der Forstdirektion, dem Forstamt, dem Revier, der Abteilung und der Unterabteilung definiert. Zusätzlich werden für die spätere Fragmentierung Informationen über benachbarte Teilflächen an den Grenzen zwischen zwei Revieren durch die Relationship **neben** definiert.

Leider können durch die groben Vereinfachungen des Schemas gegenüber dem Originalbuch viele sinnlose Daten (zum Beispiel Waldbrände im Dezember) dargestellt werden. Im Gegenzug würde die Modellierung aller Zusammenhänge des Waldschutzdatenkontrollbuches das ER-Schema qualitativ verschlechtern. Um das zu verhindern, müssen zusätzliche Integritätsbedingungen angegeben werden, die im ER-Schema nicht dargestellt werden können. Das ER-Schema wurde dann in die Relationenalgebra überführt.

3.3 Abbildung in die Relationenalgebra

Die Umsetzung des ER-Schemas in die Relationenalgebra (Abb. 3.4) weist keine Besonderheiten auf. Alle Entities erhalten je einen künstlichen Schlüssel, um die Relationen, die aus den Relationships entstehen, konsistent zu halten und die Arbeit mit dem Schema zu vereinfachen und zu beschleunigen. Die Relationen **Träger**, **Ursachen** und **Arten** wurden durch eine Relation **Namen** ersetzt, die neben einem Bereichsschlüssel einen Unterschlüssel für die dort verwendeten Begriffe enthalten. Durch den Bereichsschlüssel werden die Namensbereiche für die Schadenträger, die Ursachen und die Arten der Schäden definiert, innerhalb derer die Namen durch den Unterschlüssel identifiziert werden. Zusätzlich zu den Namen für die Schadereger, die befallenen Baumarten und die Art und Weise des Befalls werden in der Namensrelation auch die Namen für die Behandlungs- bzw. Entseuchungsarten, die Eigentumsarten und die Bezeichnungen für die Forstbehörden, die Forstdirektionen,

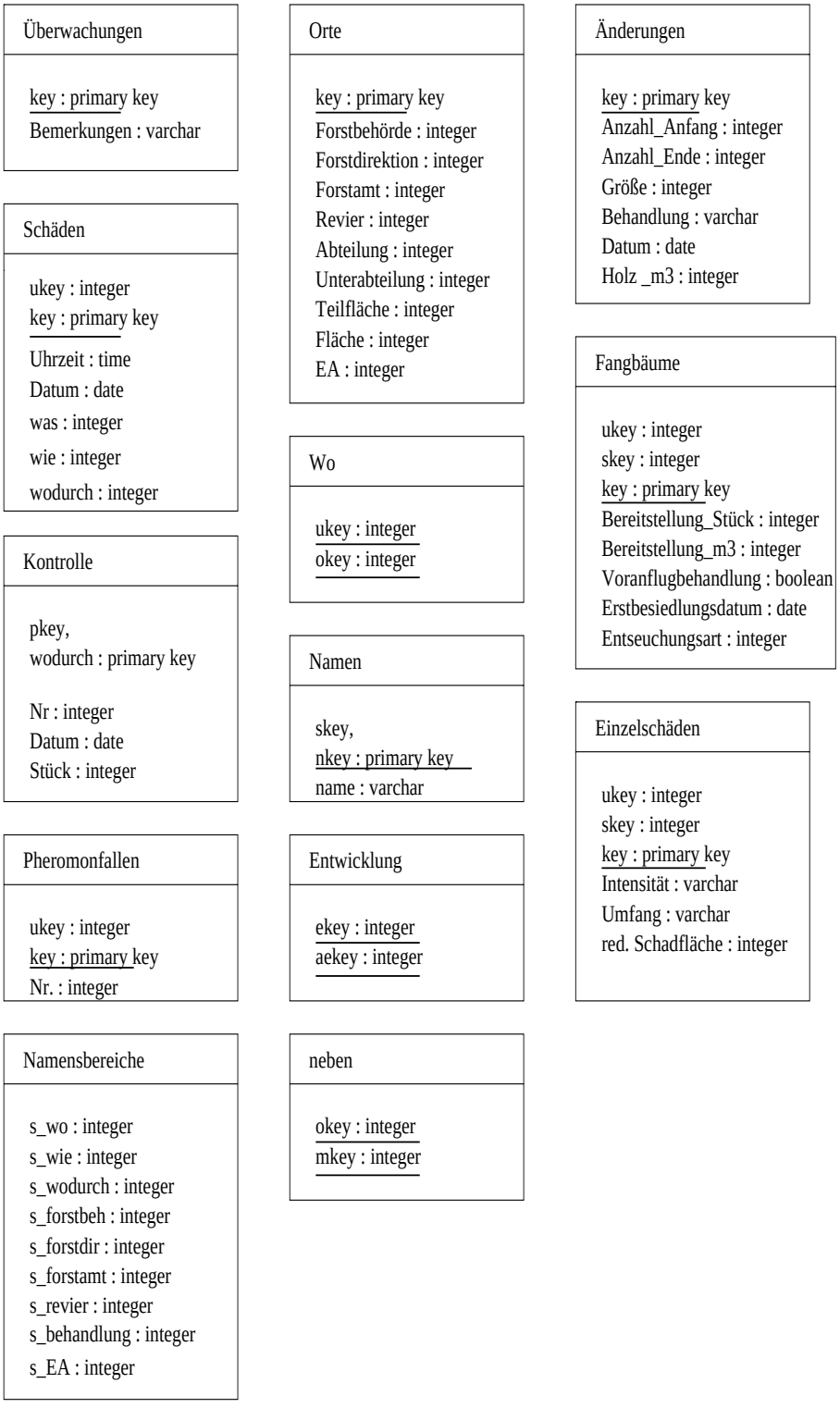

Abbildung 3.4: Rel-Schema für das Waldszenarium

die Forstämter und die Reviere verwaltet. Fr die Festlegung der Werte für die Bereichsschlüssel wurde eine eigene Relation erstellt (**Namensbereiche**) (siehe auch Abbildung 3.4).

3.4 Fragmentation des Relationenschemas

Der Ansatzpunkt für die Fragmentation des Schemas liegt in der Relation **Orte**, die die Informationen enthält, nach denen diese und andere Relationen aufzuteilen sind, um einem Benutzer dieser Datenbank den Zugriff auf die örtlichen Daten (hauptsächlich in seinem Revier) zu ermöglichen, ohne daß diese Daten über die Funkanbindung des mobilen Rechners bereitgestellt werden müssen. Da der Benutzer aber auch Zugriff auf räumlich benachbarte Gebiete seines Revieres haben soll, um eventuelle Entwicklungen früh erkennen zu können, soll auch ein definierter Randbereich, bestehend aus Teilflächen um das Revier herum, repliziert werden. Die räumliche Nachbarschaft wird durch die Relation **neben** definiert. Zu einigen an der Grenze zwischen zwei Revieren liegenden Teilflächen werden benachbarte Teilflächen (im jeweils anderen Revier) angegeben, und zwar genau dann, wenn die genannten Teilflächen eine logische Einheit (Fläche, See, Waldstück) bilden. So werden die Fragmente definiert, die auf mehreren Rechnern allokiert werden sollen. Außerdem bilden alle Teilflächen eines Revieres abzüglich denen, die in der Relation **neben** referenziert werden, ein eigenes Fragment für jedes Revier. Deshalb wird die Relation **Orte** nach einem Teilschlüssel, dem Revier, fragmentiert. Davon werden alle in der **neben** - Relation miteinander verbundenen Fragmente subtrahiert, das Ergebnis ist das Fragment für ein Revier (siehe auch Anhang B.1).

3.5 Allokation der Fragmente des Relationenschemas

Die Fragmente für ein Revier werden nur dem Rechnerknoten, der für dieses Revier zuständig ist, zugeordnet, weil diese umfangreich sind und ein Schreibzugriff nur vom jeweiligen Revier aus sinnvoll ist. Zusätzlich werden jedem Rechnerknoten alle Fragmente zugeordnet, die durch die **neben**-Relation definiert sind, wovon mindestens ein Fragment in seinem Revier steht (siehe auch Anhang B.2). Um die Anforderungen an Replikationsverfahren in mobilen Szenarien genauer zu umreißen, soll im folgenden die Problemstellung noch einmal beschrieben werden.

Kapitel 4

Problemstellung

Dieses Kapitel soll die aus dem Konflikt zwischen replizierten festnetzbasierten VDBMS und den Anforderungen mobiler Arbeit entstehenden Probleme verdeutlichen und zusammenfassen.

4.1 Auswirkungen der Mobilität auf die Replikation

Herkömmliche Replikationsverfahren für VDBMS mit Nicht-Funknetzen werfen bei der Betrachtung des vorgestellten mobilen Szenariums Probleme auf. Das Hauptproblem besteht darin, daß beabsichtigte oder unbeabsichtigte Trennungen des mobilen Rechners vom Festnetz durch Fehler oder Abschalten des Funkmodems bzw. des Rechners, z.B. aus Zeit-, Kosten- und Energiegründen, jederzeit möglich sind.

Ein weiteres Problem ist die Forderung nach Nutzung des Replikates auch nach dem Abschalten der teuren Funkverbindung, wenn nicht durch ein optimistisches Protokoll die Konsistenz zeitweise (für die Dauer der Trennung des mobilen Rechners vom Festnetz) aufgegeben und später in einer Reintegrationsphase wiederhergestellt wird. Andere Probleme ergeben sich bei Fehlerfällen wie Verbindungsunterbrechungen oder Netzpartitionierungen.

So ergeben sich für den Replikationseinsatz in mobilen VDBMS folgende Vorteile (vgl. Abschnitt 1.2.3):

- geringe Kommunikationskosten für Anfragen
- hohe Effizienz bei der lokalen Arbeit mit dem Replikat
- Mögliche Transaktionen auf 1..n verbindungslosen Replikaten
- höhere Verfügbarkeit bei sicheren Funkverbindungen

und Nachteile :

- hohe Kommunikationskosten für Updates
- geringe bis keine Verfügbarkeit bei unsicheren Funkverbindungen [1]
- mögliche Inkonsistenzen

[1] Dazu ist anzumerken, daß Funkverbindungen in Zukunft immer sicherer werden, da die Erreichbarkeit von Mobilrechnern steigen wird. Allerdings ist anzunehmen, daß der mobile Nutzer auch dann selbst bestimmen wollen wird, ob, wann und wie lange er das Funknetz nutzt.

Da die Ziele des Replikationseinsatzes auch die Minimierung der Kommunikationskosten beinhalten, wäre für den Replikatseinsatz eine hauptsächlich anfrageorientierte Arbeitsweise des mobilen Rechners auf einem relativ unveränderlichen Datenbankbereich optimal. Je stärker sich der Datenbestand auf beiden Seiten der Funkverbindung ändert, um so umfangreicher wird die Konsistenzsicherung, die die Verbindung entsprechend belastet.

4.2 Forderungen an ein Replikationsverfahren für ein VDBMS im mobilem Umfeld

Folgende Punkte sollen insgesamt von einem Replikationverfahren erfüllt werden, das zur Unterstützung von mobiler Arbeit eingesetzt werden kann:

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 ausgefallenen 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*

Im späteren Verlauf soll untersucht werden, inwieweit diese Forderungen von den bekannten Replikationsverfahren erfüllt werden können.

Der Vergleich der Replikationsverfahren, die für die Lösung dieses Problems in Betracht gezogen werden können, soll aufgrund der Hauptanforderungen der mobilen Arbeit mit Datenbanken erfolgen. Dazu soll ein einfaches Klassifikationsschema entwickelt werden.

Kapitel 5

Das Klassifikationsschema

Es soll ein Klassifikation entwickelt werden, das einen groben Vergleich von Replikationsverfahren anhand der Maßstäbe des mobilen Szenariums ermöglichen soll. Dabei sollen anfangs nur **Konsistenz** und **Verfügbarkeit** der Daten besonders in Fehlerfällen wie Verbindungsunterbrechungen oder Ausfällen von Rechnerknoten betrachtet werden. Das spezielle Problem der Netzpartitionierungen wird dabei nur am Rande betrachtet, da diese nicht als normale Fehler eingeordnet werden können und die Bewertung stark erschweren.

Dieses Schema kann generell keine allgemeingültigen Aussagen über optimistische Verfahren machen, da deren Einsatz und Erfolg zu einem großen Teil von der Verfügbarkeit von a-priori-Wissen und anwendungsspezifischem semantischen Wissen abhängt. Da diese Größen hier schwer einzuordnen und zu bewerten sind, kann ein optimistisches Verfahren sowohl den schlechtesten als auch den optimalen Erfolg in Bezug auf Konsistenzsicherung und Verfügbarkeit erreichen, ohne daß sich dabei das Verfahren ändern muß.

Die **Konsistenz** wird hier als die Summe der Konsistenz über alle Fragmente betrachtet, um das VDBMS in seiner Gesamtheit beurteilen zu können. Diese Größe soll am Ende des automatischen Teiles der Reintegration von zeitweise ausgefallenene Replikaten entstehen. **Verfügbarkeit** sei hier die Möglichkeit, auf den Daten eines Fragmentes Transaktionen ausführen zu können, unabhängig von der Allokation eines Replikates dieses Fragmentes.

5.1 Konsistenz und Verfügbarkeit in VDBMS ohne Replikation

Wenn die Daten in einem VDBMS nichtredundant gespeichert und verwaltet werden, dann führt der Ausfall eines Rechners bzw. der Ausfall seiner Verbindungseinrichtungen zu einer Verringerung der Verfügbarkeit des Gesamtsystems derart, daß im Idealfall nur die auf ihm verwalteten Daten nicht mehr gelesen oder geschrieben werden können. Das heißt, daß das Gesamtsystem zwar immer noch funktionsfähig ist, aber schon ein Ausfall hat immer eine Einschränkung der Transaktionen auf Fragmenten zur Folge (siehe Abb. 5.1). Durch die nichtredundante Datenhaltung werden aber auch Konsistenzprobleme vermieden. So werden selbst bei einer Netzpartitionierung im Idealfall die Datenbestände nicht auseinanderlaufen, vorausgesetzt, die referentielle Integrität wird ständig überwacht und so der Ausfall eines referenzierten Fragmentes erkannt. Dann werden die damit verbundenen Transaktionen verboten, so daß nur die Daten in den Netzpartitionen bearbeitet werden können, die nicht mit den Daten anderer Partitionen zusammenhängen. Somit wird

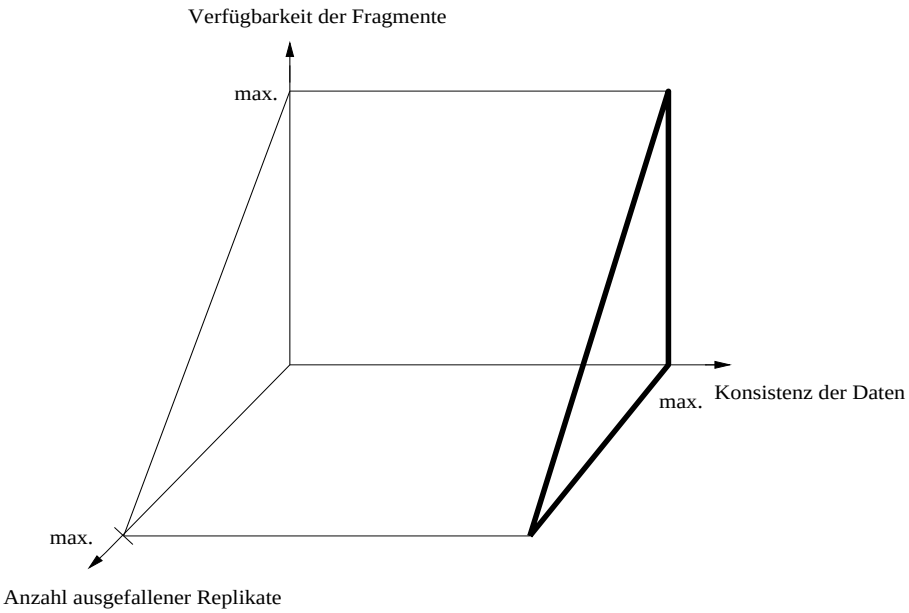

Abbildung 5.1: VDBMS ohne Replikation (pessimistisch)

mit zunehmender Zahl von Verbindungsausfällen die Verfügbarkeit der Daten geringer, die Konsistenz bleibt erhalten.

5.2 Konsistenz und Verfügbarkeit in VDBMS mit Replikation

Die Verfügbarkeit von Datenbeständen innerhalb eines VDBMS kann durch Replikation vergrößert werden. Dabei sind die Daten, die die Knoten verwalten müssen, nicht mehr disjunkt. Somit kann im Idealfall selbst bei Ausfall eines oder mehrerer Knoten bzw. Verbindungseinrichtungen die Verfügbarkeit des Gesamtsystems in Bezug auf den Datenbestand gewährleistet werden. Zugleich wird die Konsistenzsicherung komplizierter. Sobald ein Knoten nicht mehr erreichbar ist, können die bei ihm replizierten Fragmente nicht mehr synchron zu den anderen Replikaten dieses Fragmentes verändert werden. Somit braucht man ein Replikationsverfahren, das Änderungen auf nicht erreichbaren Replikaten bei einer späteren Reintegration nachführt (pessimistisch), oder das dafür sorgt, daß diese Reintegration möglichst automatisch und mit wenigen zurückzusetzenden Transaktionen erfolgt (optimistisch). Bei Netzpartitionierungen ist die Gefahr, daß die replizierten Datenbestände auseinanderlaufen, durch die Replikation größer. Vorausgesetzt, die Partitionierung ist rechtzeitig entdeckt worden, kann ein pessimistisches Verfahren die Konsistenz der Replikate nur durch Verbieten der Transaktionen auf den Replikaten gewährleisten. Dabei sind in einer Partition weiterhin Transaktionen auf den Replikaten möglich, in allen anderen Partitionen hingegen nicht. Insgesamt nimmt die Verfügbarkeit erst ab einer kritischen Anzahl von Knoten- bzw. Verbindungsunterbrechungen ab (siehe Abb. 5.2), die bestimmt wird von der Anzahl und der Verteilung der Replikate sowie dem verwendeten Replikationsverfahren. Je nach Replikationsverfahren (optimistisch) kann auch die Konsistenz sinken.

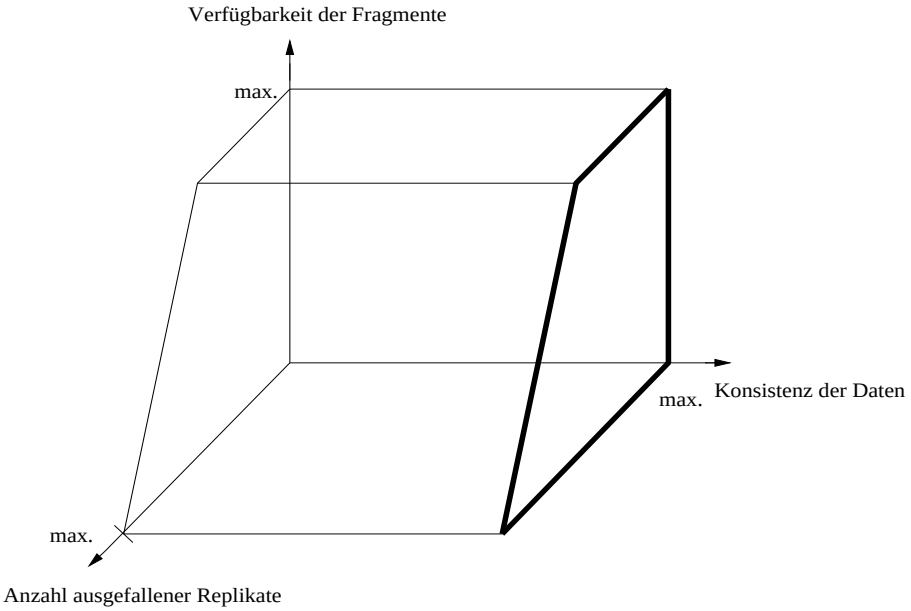

Abbildung 5.2: VDBMS mit Replikation (pessimistisch)

5.3 Konsistenz und Verfügbarkeit in mobilen VDBMS mit Replikation

Weil bei der mobilen Kommunikation auch die **Kosten** für Datenübertragungen eine gewisse Rolle spielen, werden diese Kosten den dritten Maßstab bilden. Zusammen mit der Konsistenzsicherung und der Verfügbarkeit ergeben sich so 3 Forderungen an ein mobiles VDBMS, die einen Raum aufspannen, in den die Replikationsverfahren grob eingeordnet werden können.

Wie schon bei der Vorstellung des mobilen Szenariums festgestellt wurde, sind willkürliche und lange Verbindungsunterbrechungen ein Merkmal mobiler Arbeit. Wenn man diese als eine spezielle Art von Fehler betrachtet, kann man feststellen, daß diese Unterbrechungen die Konsistenzsicherung in einem normalen replizierten VDBMS so einschränken können, daß die Verfügbarkeit auf ein Minimum reduziert werden muß, um die Konsistenz der Daten sichern zu können (pessimistisch), oder umgekehrt die Verfügbarkeit bis zu einem bestimmten Grade durch die Zulassung von möglichen Inkonsistenzen substituiert wird (optimistisch). Wenn man nun ein mobiles VDBMS wie ein normales repliziertes VDBMS auffaßt, gilt dasselbe Diagramm wie bei VDBMS mit Replikation. Der wichtigste Unterschied besteht darin, daß die typische Fehleranzahl bzw. der Umfang der auftretenden Fehler (Dauer der Unterbrechung, Einschränkung der Bandbreite) im Bereich des Maximums liegt.

Die Mobilität des Nutzers kann im allgemeinen nur durch ein Funknetz realisiert werden. Zusätzlich zu den dadurch entstehenden Kosten kommt der Energiebedarf des Mobilrechners und des Funkmodems. Daher liegt es nahe, die Verbindungszeit mit dem Festnetz so kurz wie möglich zu halten. So wird ein Großteil der mobilen Arbeit getrennt vom Festnetz durchgeführt, was die Konsistenzsicherung stark erschwert (siehe auch Kapitel 1). Wenn andererseits ein paralleles Arbeiten von mehreren (mobilen und nichtmobilen) Nutzern gefordert wird, so ist die Konsistenz nur durch längere und häufige Verbindungen mit den Festnetz zu sichern. Insgesamt aber scheint ein mobiles Arbeiten ohne Replikation wenig sinnvoll, da ohne Replikate ein Arbeiten an Mobilrechnern nur beim Einsatz von schnellen, billigen

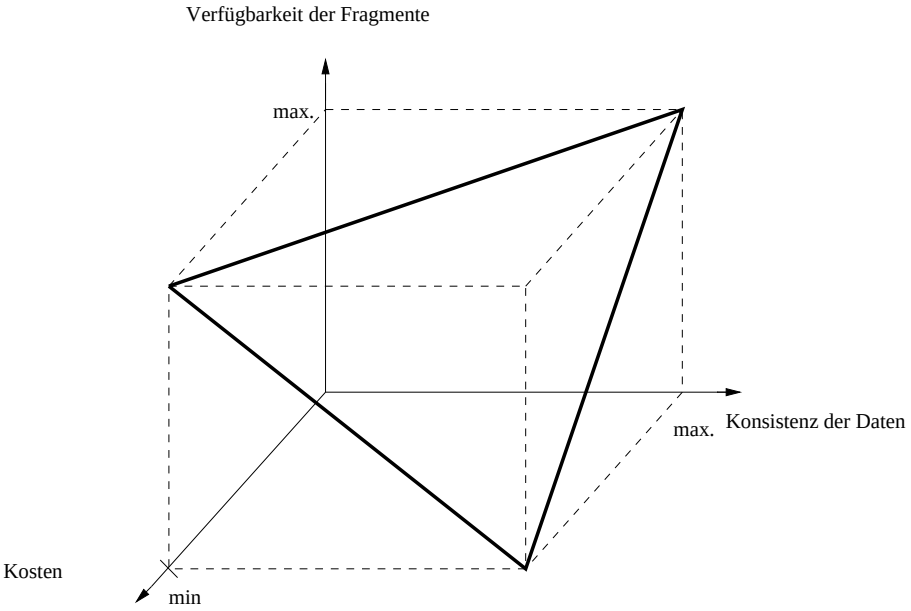

Abbildung 5.3: mobiles VDBMS mit Replikation

und sicheren Verbindungseinrichtungen möglich ist.

Im folgenden soll nun eine Bewertung der Voraussetzungen und der Aussagefähigkeit des Schemas versucht werden.

5.4 Bewertung des Schemas

Die folgende Annahme bezieht sich auf die oben beschriebene Struktur eines mobiles Szenariums:

Die Forderungen nach **Verfügbarkeit, Konsistenz** und **Kommunikationskostenminimierung** sind jeweils paarweise unabhängig voneinander erfüllbar, wobei anzunehmen ist, daß es kein Verfahren gibt, daß alle 3 Kriterien zugleich erfüllen kann. In einem festnetzbasierten VDBMS können sowohl Verfügbarkeit als auch Konsistenz voll erfüllt werden. Um aber die Kostenminimierung bei der Benutzung einer Funkverbindung wenigstens teilweise erreichen zu können, müssen zwangsläufig Abstriche entweder bei der Konsistenz (wenn spätere Reintegration möglich bzw. der dazu nötige Aufwand abschätzbar ist) und/oder Verfügbarkeit (schränkt die Arbeitsweise der Benutzer um so stärker ein, je sicherer ein Auseinanderlaufen von Replikaten verhindert werden soll) gemacht werden. Ein Mittel, das es erlaubt, in Spezialfällen trotz dieser Einschränkungen mehrere Kriterien gleichzeitig (wenn auch in unterschiedlichem Grade) erreichen zu können, ist die Nutzung von a-priori- (bzw. externem) Wissen. Zum Beispiel erlaubt es die Kenntnis der Tatsache, daß die Trennung vom Festnetz vom Nutzer getroffen wird, diese Trennung dem System vorher anzukündigen und dabei weitere Entscheidungen für das Verhalten des Systems zu treffen.

Da es dem Benutzer eines mobilen Rechners erlaubt sein soll, die Verbindung zum Festnetz abzuschalten, kann die *Netztransparenz* nicht mehr aufrechterhalten werden, d.h. er muß über die Folgen der Trennung unterrichtet werden.

Im nächsten Kapitel werden nun einige wichtige Replikationsverfahren beschrieben und erläutert werden, die im darauffolgenden Kapitel anhand des Schemas verglichen werden sollen.

Teil II

Vorstellung und Bewertung existierender Replikationsverfahren

Kapitel 6

Klassische und neue Verfahren

Dieses Kapitel soll eine Übersicht über die wichtigsten, bereits vorhandenen und untersuchten Replikationsverfahren geben.

6.1 Klassische Optimistische Verfahren

Die hier beschriebenen Verfahren haben die Verringerung des Log-Umfanges zum Ziel und verwenden vorwiegend semantische Methoden, um das zu erreichen. Da als optimistische Verfahren nur solche Verfahren bekannt geworden sind, die sich der Semantik der Transaktionen auf den Daten bedienen, ist die Auswahl auf diese speziellen Vertreter beschränkt.

6.1.1 Transformation log

Das Transformation-log-Verfahren versucht, die Menge der nachzuführenden Transaktionen bei der Reintegration durch Transformationen mit Hilfe von semantischem Wissen über diese Transaktionen zu verringern. Dazu dienen :

- **Kommutativität**
  → Vertauschen benachbarter Transaktionen
- **Überdeckung einer Transaktion** (durch eine andere)
  → Löschen der ersten Transaktion
- **Zurücksetzen einer Transaktion** (durch eine benachbarte)
  → Löschen beider Transaktionen

Dadurch wird die Menge der nachzuführenden Transaktionen geringer, und der Durchsatz steigt.

6.1.2 Data patch

Das Data-patch-Verfahren löst Inkonsistenzen mit Hilfe von Regeln auf, die bereits beim Datenbankentwurf für jede Relation festgelegt worden sind. Dabei wird jeder Relation je ein Vertreter der folgenden Regeln zugeordnet:

Einfügeregel Wenn ein Tupel nur in einem Replikat vorhanden ist, soll eine der folgenden Regeln angewandt werden :

- **Keep-Regel**
  Füge das Tupel in alle Replikate ein
- **Remove-Regel**
  Lösche das Tupel im Replikat
- **Program-Regel**
  Beauftrage das Programm mit der Konfliktlösung
- **Notify-Regel**
  Beauftrage den Datenbankadministrator mit der Konfliktlösung

Integrationsregel Wenn ein Tupel mit gleichem Schlüssel in mehreren Replikaten vorhanden ist, soll eine der folgenden Regeln angewandt werden :

- **Latest-Regel**
  Das zuletzt eingefügte Tupel ist das gültige.
- **Primary-Regel**
  Das Tupel auf dem Rechnerknoten R ist das gültige
- **Arithmetic-Regel**
  neuer Wert = Summe aller Werte über alle Replikate - alter Wert
- **Program-Regel**
  Beauftrage das Programm mit der Konfliktlösung
- **Notify-Regel**
  Beauftrage den Datenbankadministrator mit der Konfliktlösung

Im nächsten Abschnitt sollen pessimistische Replikationsverfahren vorgestellt werden, die sich auf syntaktische Informationen der Transaktionen stützen, um die Konsistenz der Daten zu sichern.

6.2 Klassische pessimistische Replikationsverfahren

Das Hauptunterscheidungsmerkmal ist die Art und Weise der Entscheidung, ob ein Replikat den aktuellen Datenbankzustand seines Fragmentes widerspiegelt. Je nachdem, ob es ein ausgezeichnetes Replikat gibt und ob für einen Zugriff die Zustimmung mehrerer Replikate werden soll, ergeben sich drei Klassen von pessimistischen Replikationsverfahren :

1. Primary-Verfahren
2. ROWA / Write-all-available
3. Voting

6.2.1 Primary-Verfahren

Die *Primary*-Verfahren basieren auf der Festlegung eines ausgezeichneten Rechnerknotens für alle Replikate aller Fragmente (primary site) oder für je ein Replikat eines Fragmentes (primary copy). Eine Transaktion auf einem Fragment ist dann möglich, wenn die primary site bzw. der Rechnerknoten mit dem ausgezeichneten Replikat (primary copy) verfügbar ist. So kann auch bei Ausfall eines Rechnerknoten mit einem Replikat eines Fragmentes auf diesem Fragment weitergearbeitet werden. Sobald der ausgefallene Knoten wieder mit dem Netz verbunden ist, werden alle zwischenzeitlichen Änderungen vom ausgezeichneten Replikat aus auf dem veralteten Replikat nachgeführt. Nach [AS97] ergeben sich 2 Möglichkeiten zur Realisierung der primary-Verfahren:

1. Verteilte Sperren auf allen replikathaltenden Rechnern
   (Bei Schreibzugriffen werden zuerst auf allen replikathaltenden Rechnern Sperren angefordert. Diese werden solange gehalten, bis alle Replikate aktualisiert sind. Die Transaktion selbst gilt bereits nach der erfolgreichen Aktualisierung des ausgezeichneten Replikates als erfolgreich abgeschlossen.)

2. Keine verteilten Sperren, (Schreib-)Sperren nur auf dem ausgezeichneten Replikat
   (Die Scheiboperationen werden nach der Aktualisierung des ausgezeichneten Replikates sobald als möglich an die anderen Replikate weitergeleitet.)

Dabei entstehen folgende Probleme:

- Zu 1.) Bei Rechnerausfällen während der Aktualisierung der Replikate wird das Ende der Transaktion unter Umständen stark verzögert. (Rechnerausfall schnell erkennen)

- Zu 2.) Leseoperationen können noch nicht aktualisierte Daten lesen.

Das 2. Problem kann nach [AS97] durch eine der folgende Festlegungen vermieden werden:

1. Nur auf dem ausgezeichneten Replikat kann gelesen werden. Dadurch wird das VDBMS effektiv wieder zu einem nichtreplizierten VDBMS.

2. Lesesperren werden nur auf dem ausgezeichneten Replikat gesetzt, der Lesezugriff selbst wird lokal durchgeführt.

3. Durch ein Mehrversionenkonzept werden veraltete Kopien erkannt und aktualisiert.

4. Die möglicherweise veraltete und inkonsistente Sicht auf die Daten wird akzeptiert.

Für die weiteren Betrachtungen des primary-Verfahrens wird der Zweckmäßigkeit halber die Implementierung mit verteilten Sperren benutzt werden, obwohl das Verfahren ohne verteilte Sperren mit der Lesesperre auf dem Primary sehr effizient zu sein scheint. Leider kann man annehmen, daß es sinnlos ist, von einem mobilen Rechner aus für den Zugriff auf das lokale Replikat Sperren auf einem Rechner aus dem Festnetz anzufordern, da das die rein mobile Arbeit ohne eine Festnetzverbindung verhindern würde. Allerdings ist das Verfahren leichter zu implementieren, weshalb es noch einmal bei der Implementierung des Simulationssystems (Kapitel 10) zur Anwendung kommt.

6.2.2 ROWA / Write-all-available

Bei dem **ROWA-Verfahren** (Read-one write-all) werden alle Änderungen in einem Replikat sofort an die anderen Replikate propagiert. Somit werden alle Schreibzugriffe weitergeleitet, nachdem sie lokal ausgeführt wurden. Lesezugriffe können stets auf dem nächsten erreichbaren Replikat erfolgen und sind so sehr schnell und verursachen geringe Kosten. Wenn allerdings ein Replikat nicht erreichbar ist, wird dadurch das Ende der Transaktion so lange verzögert, bis das Replikat wieder bereit ist.

Das **Write-all-available-Verfahren** arbeitet prinzipiell wie das ROWA-Verfahren, aber bei einem Replikatsausfall wird die Transaktion auf dessem Fragment als erfolgreich beendet, wenn alle erreichbaren Replikate (mindestens eines) die Transaktion nachvollziehen konnten. Für die nicht erreichbaren Replikate des Fragmentes werden die folgenden Transaktionen bis zur Reintegration dieser Replikate mitgeloggt und dann nachgeholt. So blockieren ein oder mehrere ausgefallene Knoten nicht die Arbeit auf den Fragmenten ihrer Replikate.

6.2.3 Voting

Die Voting-Verfahren basieren auf dem Prinzip der Abstimmung zur Durchführung von Transaktionen auf replizierten Daten. Dabei wird einer Transaktion der Zugriff auf Daten eines Replikates nur dann erlaubt, wenn diesem Zugriff ein entsprechendes **Quorum** zustimmt.

Definition 5 ***Quorum*** *Ein* ***Quorum*** *ist eine entscheidungsfähige Anzahl an Stimmen, die zu einem bestimmten Datenzugriff (Lesen/Schreiben) auf die Daten eines Fragmentes angefordert werden.*

Dabei wird meist zwischen einem Schreibquorum Q_W, das für die Zulässigkeit einer Schreiboperation erforderlich ist, und einem Lesequorum Q_R für einen Lesezugriff unterschieden. Um Dateninkonsistenzen zu vermeiden, sind folgende Ungleichungen einzuhalten:

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

Voting-Verfahren lassen sich grob durch folgende Kriterien einteilen :

- **Unstrukturiertes/Strukturiertes Quorum**
  Alle Stimmen sind für die Abstimmung zugelassen/Die nötigen Stimmen können nur noch von bestimmten Knoten in der gleichen Struktur der Verteilung der Replikate eingesammelt werden.
- **Statisches/Dynamisches Quorum**
  Die Anzahl aller Stimmen ist konstant/Die Stimmenanzahl kann sich ändern.

Je nach Verfahren können die replikathaltenden Rechner und andere Hilfseinrichtungen, wie Ghosts, Witnesses, Bystanders und Referees abstimmen. Dabei können die Stimmen sowohl gewichtet als auch mehrfach verteilt werden. Wenn die Verteilung der Replikate entsprechend einer Struktur wie z.B. einem azyklischen Graphen erfolgt, kann die Abstimmung strukturiert erfolgen, so daß die Zustimmung von übergeordneten Rechnerknoten mehr wert ist als die Zustimmung untergeordneter Rechnerknoten, oder die Stimmen von Rechnerknoten im selben Teilbaum sind wichtiger als die Stimmen von Rechnerknoten in einem benachbarten Teilbaum.

6.3 Neue Verfahren

In vielen Fällen konnten die klassischen Replikationsverfahren nicht den gewünschten Erfolg erzielen. So wurden neue Verfahren entwickelt, die versuchten, die Schwächen der klassischen Verfahren zu vermeiden bzw. zu verbessern. Insbesondere die Probleme bei der Ausführung von Zugriffen auf voneinander getrennte Replikate wurden intensiv untersucht, um bei unvermeidbaren Verbindungsunterbrechungen weitere Arbeiten auf dem mobilen Replikat zu ermöglichen. Da es sich fast ausschließlich um pessimistische Verfahren handelt, fassen diese eine unterbrochene Verbindung meist als Fehler auf. Dabei sind vielfach Anpassungen an spezielle Umgebungen gemacht worden. So beziehen sich einige der veröffentlichten Verfahren z.B. auf Filesysteme und Cachestrategien. Diese sollen hier nicht betrachtet werden, da sie sich, wie in [SKB+93] bereits festgestellt wurde, verhältnismäßig gut über optimistische Protokolle behandeln lassen. Der Grund dafür lag in der Arbeitsweise der mobilen Nutzer, die fast nur auf den eigenen Datenbeständen arbeiten, womit die Wahrscheinlichkeit für Inkonsistenzen sehr gering ist. Hier sollen einige Verfahren vorgestellt werden, die auf einer Erweiterung der klassischen Verfahren beruhen.

6.3.1 Virtual Primary Copy (VPC)

In [?] wird ein Verfahren vorgestellt, das das Primary-Copy-Verfahren um die Möglichkeit der Trennung des ausgezeichneten Replikates auf einem Mobilrechner vom Festnetz erweitert. Dabei wird von einer Netzstruktur wie in [IB92a] ausgegangen. Das VPC-Verfahren setzt voraus, daß die Primärkopie des Replikates auf dem Mobilrechner liegt. Sobald dieser aber nicht mehr in Verbindung mit den anderen Rechnerknoten steht, kann auf dem Fragment dieses Replikates nicht mehr von einem anderen als dem *primary* gearbeitet werden. Um trotzdem Transaktionen auf dem Replikat des mobilen Rechners ausführen zu können, während dieser vom Netz getrennt ist, wird ein Stellvertreter für diesen Mobilrechner und seine Replikate benötigt. Für diese Aufgabe wurde der Home-Base-Node (HBN) des Mobilrechners gewählt. Dadurch wird die Reintegration zwischen dem Festnetz und dem Mobilrechner erleichtert, weil nur diese beiden Rechner für die Synchronisierung beider Replikate miteinander kommunizieren müssen und beide eine direkte Verbindung zueinander haben. Da das Verfahren die Probleme bei der mobilen Replikation mit einem Client auf die Replikation über eine direkte Netzwerkverbindung umlenkt, muß es mit einem anderen Verfahren zur Integritätssicherung kombiniert werden, welches dann nur den Abgleich zweiter Replikate durchführen muß und das deshalb vereinfacht werden kann. Denkbar wäre zum Beispiel der Einsatz eines optimistischen Replikationsverfahrens. In [?] wird zur Behebung der Konflikte die Ausführung von kompensierenden Transaktionen vorgeschlagen.

6.3.2 Voting - Erweiterungen

Die Voting-Verfahrensklasse ermöglicht bei geeigneten Kommunikationsmitteln eine verteilte Entscheidungsfindung. Einige Verfahren können mit einer veränderlichen Anzahl von Replikaten umgehen, auch im Fehlerfall, wenn nicht alle Replikate erreichbar sind. Der Vorteil der Kombination von Voting und anderen Verfahren liegt darin, daß (vorausgesetzt, die Fehlererkennung ist möglich) in einem fehlerfreien Zustand ein relativ einfaches, aber auch schnelles und ein an Kommunikationskosten billiges Verfahren zum Einsatz kommt, während erst im Fehlerfall ein komplexes und teures Voting-Verfahren benutzt wird. (siehe auch **Missing Writes**).

Über die Erweiterung klassischer Techniken hinaus wurden auch völlig neue Replikationstechniken entwickelt, die unter anderem dynamische Replikation und Verfahrenswechsel in Fehlerfällen erlauben.

6.3.3 ADR-Verfahren

[WJH97] beschreibt ein dynamisches Verfahren, bei welchem zur Laufzeit durch Statistiken die Schreib-/Leseanforderungen von Knoten an ein Fragment ermittelt werden. Das Verhältnis der Schreib-/Leseanforderungen gibt darüber Auskunft, inwieweit sich die Replikation des Fragmentes auf einem bestimmten Rechnerknoten positiv oder negativ auf die Kommunikationskosten und -zeiten auswirkt. Die Knoten, die (im Verhältnis zum Schreiben) relativ viel lesen, erhalten ein Replikat, den Knoten, die (im Verhältnis zum Lesen) relativ viel schreiben, wird das Replikat entzogen. Die Replikate werden also je nach Schreib-/Lese-Anforderungen der Rechnerknoten erzeugt, verschoben oder gelöscht. Innerhalb einer bestimmten Zeit nähert sich so das Replikationsschema dem optimalen Zustand. Der optimale Zustand wird aber nur dann erreicht, wenn sich die Schreib-/Leseanforderungen in der Anpassungszeit nicht mehr ändern. Als Replikationsverfahren selbst findet das **Primary Missing Writes**- Verfahren Anwendung.

6.3.4 Wechsel der Verfahren im Fehlerfall

Einige Verfahren wurden speziell für häufige Verbindungsunterbrechungen entwickelt. Sie können das Replikationsverfahren (und auch -schema) zur Laufzeit wechseln, wenn ein Fehler beim Zugriff auf ein Replikat gefunden wurde.

Missing Writes Das Verfahren [Bernstein et al. 1987] eignet sich für eine feste Anzahl von Replikaten. Im normalen, fehlerfreien Fall wird das ROWA-Verfahren benutzt. Wenn eine Verbindungsunterbrechung gefunden wurde, wird auf das **Quorum Consensus**-Verfahren (Gifford [1979], Thomas [1979]) umgeschaltet. Zusätzlich wird a-priori- Wissen über die Gesamtanzahl der Replikate benutzt, um herauszufinden, welche Partition eine Mehrheit von Replikaten besitzt. Diese Partition darf dann als einzige auf diese Replikate schreibend zugreifen [WJH97].

Primary Missing Writes In [WJH97] wird dieses Verfahren zur Replikationskontrolle im ADR-Verfahren vorgestellt. Dabei handelt es sich nicht im eigentlichen Sinne um einen Verfahrenswechsel, denn als Verfahren wird ROWA benutzt. Im Fehlerfall wird die Anzahl der Replikate auf 1 heruntergesetzt. Wenn die Fehler wieder behoben wurden, werden die Transaktionen auf den inkonsistenten Replikaten nachgeführt, wodurch alle Replikate wieder den gleichen Zustand haben. Dann wird die Anzahl der gültigen Replikate auf den Wert im fehlerfreien Fall wieder erhöht.

6.3.5 Clusterbildung

In [Pi96] wird ein Replikationsverfahren vorgestellt, das zwischen vollständiger Konsistenz und Grad-m-Konsistenz unterscheidet. Das Konzept basiert auf der Clusterbildung von Replikaten, die untereinander vollständig konsistent (strikte Konsistemz) sein sollen. Für die Konsistenz der Cluster untereinander wird die abgeschwächte Konsistenz gefordert. Unter Zuhilfenahme der vorhersagbaren Trennungen des Mobilrechners vom Netz und von den Applikationen gestellten Konsistenzanforderungen werden in einem Cluster alle Mobilrechner zusammengefaßt, deren Replikate untereinander konsistent gehalten werden sollen. Um diese Art der Datenhaltung zu unterstützen, wird auch bei Transaktionen zwischen strikten und schwachen Transaktionen unterschieden. Dabei werden strikte Transaktionen nach einem pessimistischen Konsistenzsicherungsverfahren (Quorum consensus) ausgeführt, schwache dagegen nach einem optimistischen.

6.3.6 Replikation in MAID

Im Rahmen von MAID wurde ein auf mobilen Einsatz optimiertes Replikationsverfahren entwickelt [OZ97], das unter anderem Akzeptanzkriterien für den lokalen Zugriff auf Daten unterstützt. Gleichzeitig werden nicht nur die Daten von Fragmenten, sondern auch entsprechende aktive Regeln repliziert, die z.B. die Nutzung der replizierten Daten festlegen. Durch ein für aktive Regeln entwickeltes Replikationsverfahren wird außerdem sichergestellt, daß die Regeln nur einmal ausgeführt und die Ausführung an den Zustand der Netzwerkverbindungen angepaßt werden kann. Beim Einsatz des Verfahrens werden 3 mögliche Zustände der Verbindungseinrichtungen unterschieden :

1. stark verbundener Zustand,
2. schwach verbundener Zustand und
3. verbindungsloser Zustand

Im *stark verbundenen Zustand* kann 1-Kopien-Serialisierbarkeit gefordert werden, da die Verbindungseinrichtung fehlerfrei arbeitet. Zum Einsatz kommt hier das *primary-copy*-Verfahren. Die Verbindungsqualität reicht im *schwach verbundenen Zustand* nicht mehr aus, um alle Zugriffe auf der Primärkopie ablaufen zu lassen. Deshalb wird eine Art Datenbank-Cache auf dem Mobilrechner eingerichtet, der insbesondere Lesezugriffe beschleunigen soll. Weil die Verbindungseinrichtung im *verbindungslosen Zustand* ausgefallen oder ausgeschaltet worden ist, wird nur auf den lokal replizierten Daten gearbeitet. Dazu wurde ein virtuelles Primärkopieverfahren (VPK) entwickelt. Dabei wird die nicht mehr erreichbare Primärkopie durch eine virtuelle Primärkopie auf dem mobilen Rechner ersetzt. Alle lokalen Zugriffe auf diese Kopie werden protokolliert. Beim Abgleich der VPK und der Primärkopie werden mit Hilfe der Akzeptanzkriterien die Transaktionen von der VPK auf der Primärkopie nachgeführt. Zusätzlich wird die Entstehung von *ad-hoc-Netzverbindungen* unterstützt. Dabei geht es um einen Zusammenschluß mehrerer mobiler Rechner über z.B. eine Infrarotverbindung. Innerhalb dieses Netzes kann dann eine hohe Konsistenz der Replikate untereinander gefordert werden. Darin ähnelt das Verfahren dem der *Clusterbildung*.

Die vorgestellten Replikationsverfahren sollen im folgenden anhand ihrer Strategie mit Hilfe des Klassifikatiosschemas verglichen werden.

Kapitel 7

Einordnung der Verfahrensklassen

Um die im letzten Kapitel vorgestellten Replikationsverfahren grob vergleichen zu können, sollen hier die Grundklassen der Replikationsverfahren in das Klassifikationsschema eingeordnet werden. Dabei werden insbesondere die klassischen pessimistischen Verfahren (*Primary site/primary copy, ROWA/Write-all-available, Voting*) betrachtet, da sie die Grundlage für die bereits vorgestellten Erweiterungen bilden. Zuerst soll die optimistischen Verfahrensklasse kurz bewertet werden.

7.1 Optimistische Verfahren

Die optimistischen Verfahren nehmen auftretende Inkonsistenzen in Kauf, um die Verfügbarkeit der Replikate zu gewährleisten. Bei einem späteren Abgleich erkannte Differenzen werden entweder automatisch oder mit Hilfe des Nutzers beseitigt. Im Durchschnitt gesehen, lassen die optimistischen Replikationsverfahren eine Verringerung der Konsistenz zu, um die Verfügbarkeit der Replikate zu gewährleisten (Abb. 7.1). Die hier vorgestellten Verfahren nutzen, soweit vorhanden, semantisches Wissen über Transaktionen aus. Deshalb sind sie nicht allgemein anwendbar und müssen an spezielle Umgebungen angepaßt werden. Das macht sie für Spezialfalle interessant, bei denen genug Wissen über die auszuführenden Transaktionen vorliegt.

7.2 Pessimistische Verfahren

Die pessimistischen Verfahren schränken die Verfügbarkeit der Replikate ein, wenn die Sicherung ihrer Konsistenz untereinander nicht mehr gewährleistet werden kann (Abb. 7.2). Dadurch ist eine Reintegration von zeitweise nicht verfügbaren Replikaten problemlos möglich. Diese Verfahren sind syntaktischer Natur und somit allgemein verwendbar.

7.2.1 Primary site/Primary copy

Die Verfügbarkeit eines VDBMS mit dem Primary-Verfahren hängt fast vollständig von der Verfügbarkeit des Rechners mit den ausgezeichneten Replikaten ab. Es bietet sich für eine stark heterogene Umgebung an, z.B. ein VDBMS auf einem Mainframe und mehreren PCs [BD96], wobei dem Mainframe die Rolle des Rechners mit den ausgezeichneten Replikaten bzw. des Primary zugeordnet wird. Die

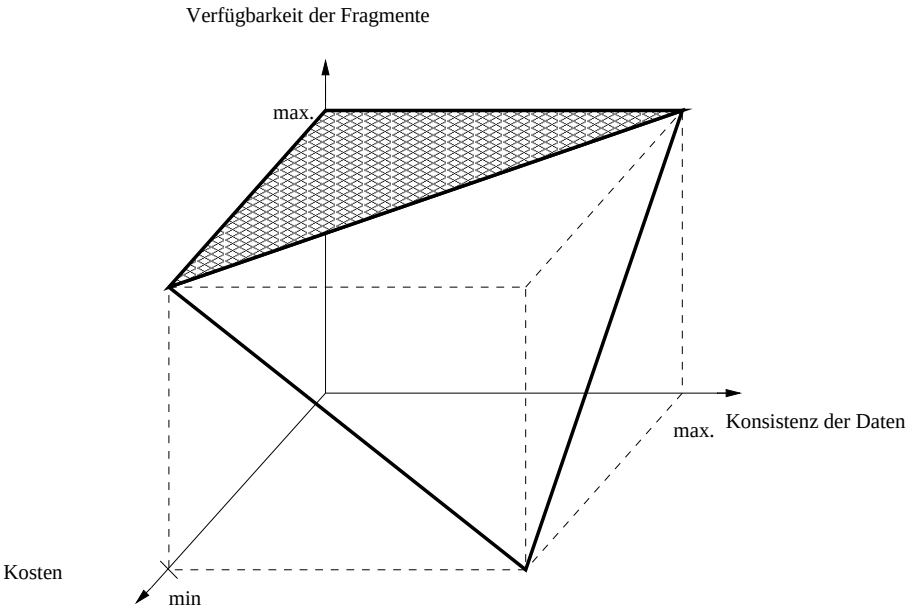

Abbildung 7.1: optimistische Verfahren (Abschätzung)

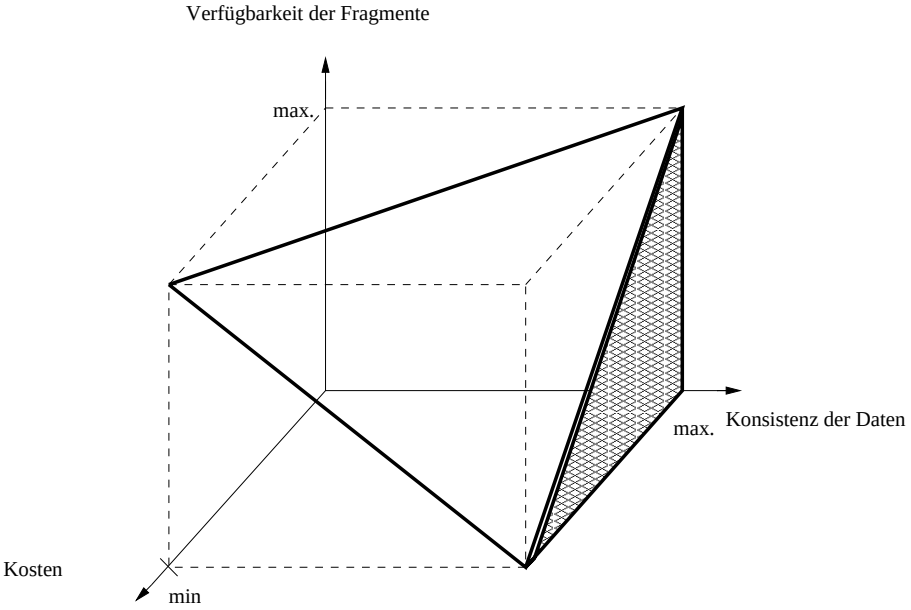

Abbildung 7.2: pessimistische Verfahren (Abschätzung)

Verfügbarkeit kann durch die Festlegung eines neuen Rechnerknotens für die ausgezeichneten Replikate bei Ausfall des alten Rechnerknotens erhöht werden, was aber ein Auseinanderlaufen der Datenbestände im Falle einer Netzpartitionierung zur Folge haben kann.

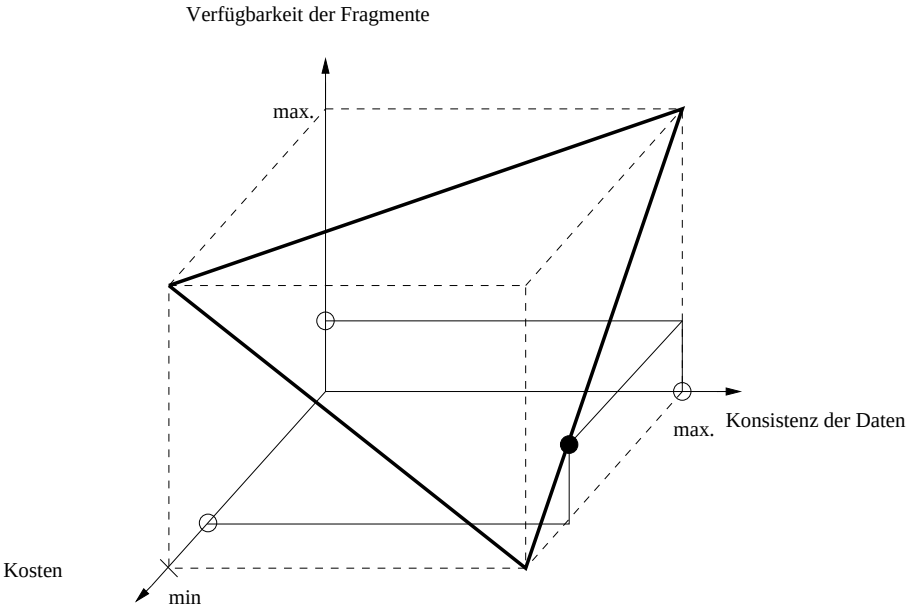

Abbildung 7.3: Primary site/ Primary copy (Abschätzung)

Die Vertreter der Klasse der Primary-Verfahren sind relativ leicht zu implementieren und erzeugen meist nur geringe Kommunikationskosten. Dafür ist die Verfügbarkeit im Fehlerfall am Primary am geringsten, verglichen mit den anderen pessimistischen Replikationsverfahren, bei gleichen Knoteneigenschaften. Das Verfahren selbst braucht nur einmal für den Server und die mehrfach für die Clients bereitgestellt zu werden. Diese Asymmetrie vereinfacht die Implementierung, da es sich um einen zentralen Algorithmus handelt. Die Konsistenz bleibt bei den Primary-Verfahren gesichert, da es pessimistisch arbeitet.

7.2.2 ROWA/Write-All-Available

Erst nachdem alle Replikate aktualisiert wurden, wird die Transaktion als abgeschlossen betrachtet. Das führt zu einer Verringerung der Verfügbarkeit beim Ausfall eines Rechnerknotens, da die Transaktion nicht beendet werden kann, bis sich der ausgefallene Rechnerknoten wieder funktionsfähig meldet.

Da beim Write-All-Available-Verfahren zur erfolgreichen Ausführung einer Schreiboperation nur die Aktualisierung der erreichbaren Replikate notwendig ist und die Änderungsoperationen auf den ausgefallenen Replikaten bis zu deren Reintegration verzögert werden kann, erhöht sich die Verfügbarkeit insbesondere bei Verbindungsunterbrechungen, kann aber bei Netzpartitionierungen zu einem Auseinanderlaufen der Datenbestände führen (7.4).

ROWA-bzw.- Write-all-available-Verfahren bieten, verglichen mit den Primary-Verfahren, höhere Verfügbarkeit, erfordern aber auch höhere Kommunikationskosten, da das Verfahren dezentral implementiert werden muß. Es gibt nur ein Verfahren, da nicht zwischen Servern und Clients unterschieden werden muß. Auch bei diesen Verfahren bleibt die Konsistenz gesichert.

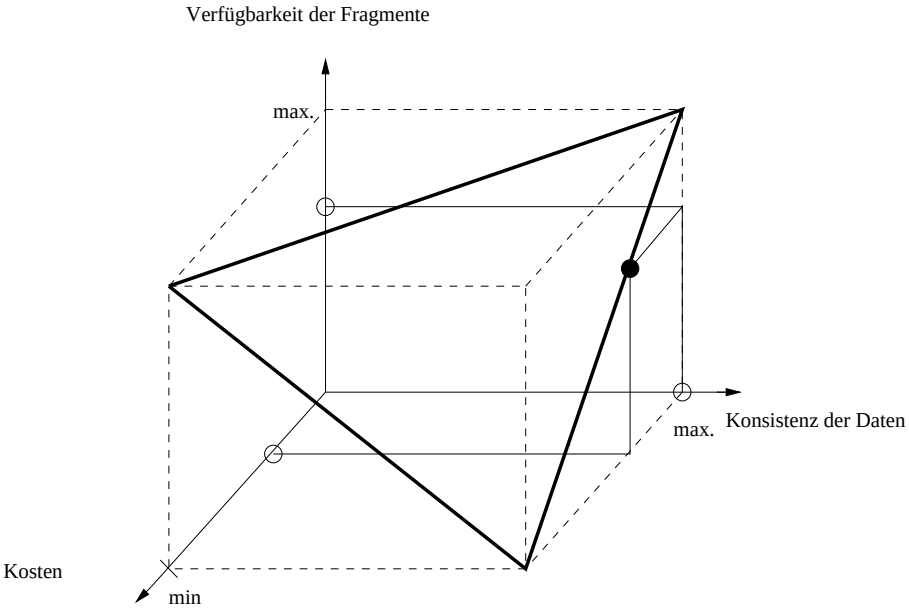

Abbildung 7.4: Write-all-available (Abschätzung)

7.2.3 Voting

Das Voting kommt der Forderung nach einem robustem, verteilten und effektiven Verfahren am nächsten. Dabei werden die Kapazitäten der Verbindungseinrichtungen sehr intensiv ausgenutzt, was eine verhältnismäßig hohe Netzlast zur Folge hat.

Nachdem die Verfahrensklassen im allgemeinen anhand der Kriterien des mobilen Szenariums verglichen wurden, sollen nun die Verfahren im einzelnen bewertet werden.

Aus den sich durch den Vergleich abzeichnenden Vor- und Nachteilen der Verfahren bei der Anwendung im mobilen Bereich sollen nun Lösungen des in Kapitel 3 dargelegten Problems vorgestellt werden.

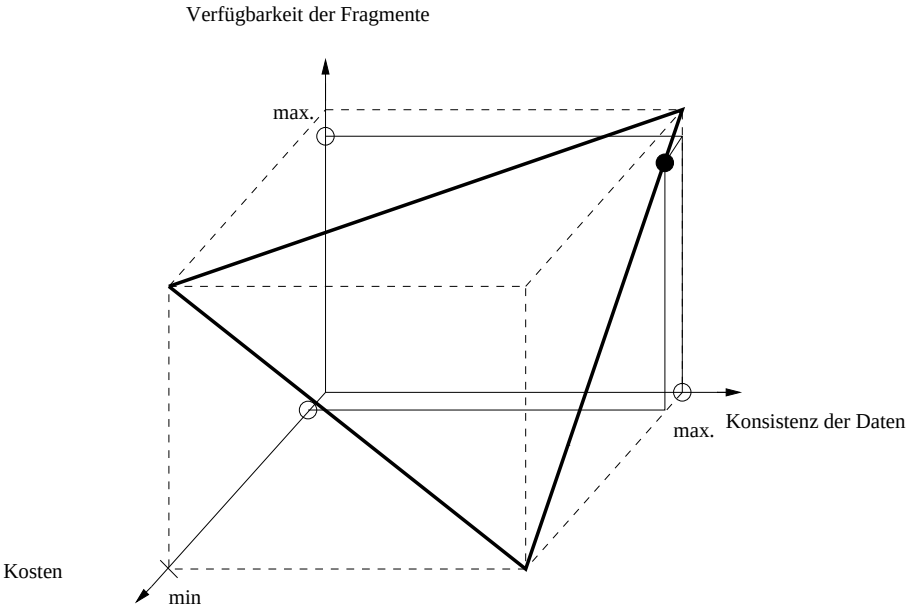

Abbildung 7.5: Voting - Verfahren (Abschätzung)

Kapitel 8

Allgemeine Bewertung der Verfahren

In diesem Kapitel sollen Replikationsverfahren bewertet werden, die bereits Ansätze für die Unterstützung von **disconnected operations** bieten und/oder besonders gut mit Verbindungsausfällen bzw. Netzpartitionierungen umgehen können.

8.1 Bewertung der einzelnen Verfahren

Die bisher vorgestellten Verfahren werden kurz anhand des Einsatzes im beschriebenen mobilen Szenarium bewertet. Daran schließt sich eine zweite Bewertung an, die an den Kriterien eines VDBMS für den mobilen Einsatz ausgerichtet ist.

8.1.1 optimistische, semantische Replikationsverfahren

Die optimistischen Verfahren sind bereits für den Zweck der Replikation ohne Konsistenzkontrolle entworfen worden. Prinzipiell eignen sie sich für den Einsatz in mobilen Umgebungen am besten.

Log transformation Da das Verfahren semantische Informationen nutzt, ist es nicht allgemein anwendbar. Der Einsatz ist dann von Vorteil, wenn die Abschätzung der zu erwartenden Inkonsistenzen gering und ein Rücksetzen von Transaktionen nicht kritisch ist. Im Zweifelsfall ist es wie alle semantischen Verfahren auf die Nutzerentscheidung angewiesen.

Data patch Prinzipiell gilt dasselbe wie bei der **log transformation**. Die Daten aus dem Datenbankentwurf sind im allgemeinen nicht ausreichend, um den Umfang der bei der Reintegration auftretenden Inkonsistenzen so zu verringern, daß eine Entscheidung eines Nutzers nicht mehr nötig ist und der gesamte Reintegrationsvorgang automatisch und transparent für den mobilen Nutzer bleibt. Da aber mehr semantisches Wissen durch den Mehraufwand beim Datenbankentwurf verwendet werden kann, verringert sich der Anteil der manuell aufzulösenden Inkonsistenzen.

8.1.2 Pessimistische, syntaktische Replikationsverfahren (Klassische Verfahren)

Obwohl die klassischen Verfahren an die Bedingungen eines festnetzbasierenden VDBMS angepaßt sind, sind sie in Grenzfällen auch in mobilen Umgebungen brauchbar. Die stark ausgeprägte Asymmetrie zwischen den eingesetzten Rechnern im

Fest- und Mobilnetz sowie zwischen den Verbindungseinrichtungen begünstigt z.B. das Primary-Verfahren.

Primary site/Primary copy Besonders das *Primary site*-Verfahren ist aufgrund der geringen Kommunikationskosten und der Struktur an das mobile Szenarium angepaßt. Der *Primary host* kann durch einen oder mehrere Rechner im Festnetz (hohe Verfügbarkeit) des Szenariums realisiert werden. Die mobilen Rechner sind, solange sie auf ihren Replikaten arbeiten, an den *Primary* gebunden. Solange dieser verfügbar ist, kann auf dem entsprechenden Fragment gearbeitet werden. Das gilt auch, wenn ein Replikat nicht mehr erreichbar ist. Bei der Reintegration werden alle Transaktionen von der Primärkopie nachvollzogen. Somit sind zwar Transaktionen auf dem getrennten Replikat möglich, aber sinnlos.

ROWA/Write-all-available Das *ROWA*-Verfahren ist für den mobilen Einsatz nicht geeignet, da es auf der ständigen Verfügbarkeit aller Replikate basiert. Außerdem sind die Kommunikationskosten höher als bei den *Primary*-Verfahren. Mögliche Probleme bei der Verfügbarkeit einzelner Replikate werden beim *Write-all-available*-Verfahren bereits berücksichtigt. Dabei hat die Struktur des physikalisch vorhandenen Netzes keinen Einfluß auf das Verfahren. Als Nachteil ist zu erkennen, daß die Informationen des Transaktionslogs ebenfalls repliziert gespeichert werden müssen, es sei denn, man hebt die Einheitlichkeit zwischen den Knoten auf und favorisiert einen Knoten mit höherer Verfügbarkeit als primären Logspeicher. Durch die Gleichheit der Knoten untereinander ist dieses Verfahren besonders für unstrukturierte Netze mit Rechnern von ähnlicher Verfügbarkeit, die auf untereinander replizierten Daten arbeiten, geeignet. Bei einfachen Verbindungsausfällen arbeitet das Verfahren problemlos, bei Netzpartitionierungen müssen Entscheidung bezüglich gültiger (arbeitsfähiger) Teilmenge an Knoten getroffen werden, z.B. über versionierte Token.

Voting Beim Voting ergibt sich eine sehr viel höhere Last auf den Verbindungseinrichtungen als bei den anderen Verfahren, da über jeden Zugriff entschieden werden muß. Die Verfahren können den physikalischen Gegebenheiten des Netzes, wie der Struktur und den Verfügbarkeiten der Verbindungseinrichtungen, angepaßt werden. Diese Verfahrensklasse kann wegen der intensiven Nutzung d er Verbindungseinrichtungen nur schwer an ein mobiles Netz mit seinen geringen Datenübertragungsraten und häufigen Ausfällen angepaßt werden. Ideal ist es für mobile Rechner, die über sicherere und schnellere Kommunikationswege miteinander kommunizieren können.

8.1.3 Pessimistische, syntaktische Replikationsverfahren (Ergänzung von Standardverfahren)

Die von den festnetzbasierenden Replikationsverfahren abgeleiteten Verfahren sind zum Teil direkt für den Einsatz in mobilen Umgebungen geschaffen worden.

Virtual Primary Copy (VPC) Der *home base node (HBN)* stellt die Vertretung für den vom Festnetz abgetrennten mobilen Host dar. Dadurch ist die Arbeit auf dem Replikat im Festnetz zwar möglich, aber wenn zusätzlich noch auf dem Replikat eines oder mehrerer mobiler Rechner gearbeitet wird, kann es bei der Reintegration dieser Replikate zu Konflikten kommen, die dann entweder wie beim ursprünglichen *Primary*-Verfahren über die höhere Priorität des *Primary* gelöst oder gesondert behandelt werden müssen, z.B. über die vorgeschlagenen kompensierenden Transaktionen.

ADR-Verfahren Da die hierzu erforderliche Kommunikation umfangreich ist, das Verfahren auf einer starren Punkt-zu-Punkt-Topologie des Netzes basiert und eigentlich kein Replikationsverfahren an sich darstellt, soll es hier nicht betrachtet werden, obwohl gerade hier die Kommunikationskosten durch die Optimierung der Plazierung der Replikate optimiert werden, was von keinem anderen Verfahren ermöglicht wird. Es eignet sich daher für ausgedehnte Mobilnetze, welche ausschließlich aus Teilstreckennetzen besteht, was auf das vorgestellte mobile Szenarium nicht zutrifft, da die Mobilrechner untereinander nicht kommunizieren.

Verfahrenswechsel im Fehlerfall Diese Verfahren setzen voraus, daß eine rechtzeitige Erkennung von Verbindungsunterbrechungen stets moglich ist. Dabei ist zu bemerken, daß der mobilen Einsatz fast ausschließlich aus Fehlersituationen wie Verbindungsunterbrechungen besteht. Beim *Missing Writes - Verfahren* wird standardmäßig ROWA benutzt. Wenn der Zugriff auf einige Replikate nicht mehr möglich ist, wird auf ein Voting-Verfahren gewechselt. Durch die primäre Verwendung des ROWA-Verfahrens, welches die Verbindungseinrichtungen relativ wenig belastet, wird die Transaktionsgeschwindigkeit erhöht. Da aber im Fehlerfall das *Quorum Consensus*-Verfahren eingesetzt wird, welches im vorgestellten mobilen Szenarium aufgrund seiner fehlenden Struktur nicht geeignet zu sein scheint, ist es optimal für ein wenig strukturiertes Netz.

Bei dem *Primary Missing Writes - Protokoll* wird im Falle von Fehlern die Anzahl der Replikate eines Fragmentes auf 1 veringert. Dann kommt statt des ADR-(bzw ROWA)-Verfahren das Primary Copy-Verfahren zum Einsatz. Dieses Verfahren ist für den Einsatz im mobilen Szenarium nicht optimal geeignet, da sich einerseits das Festnetz als replikathaltender Mobilnetzknoten anbietet, andererseits aber die Anzahl der Verbindungsunterbrechungen so hoch und ihre Dauer wahrscheinlich so groß sein wird, daß das Verfahren die Verfügbarkeit der noch erreichbaren Replikate zu stark einschränkt.

Clusterbildung Das *Cluster*-Verfahren ist für das mobile Szenarium nicht besonders geeignet, da ein Teil der Kommunikation über Verbindungen zwischen den Mobilrechnern abläuft. Ohne diese Kommunikation ist es aber nicht sinnvoll, einen Cluster zu bilden; die Mobilrechner kommunizieren im mobilen Szenarium fast ausschliesslich mit dem Festnetz und nur zu einem geringen Teil bzw. gar nicht miteinander.

Replikation in MAID Da das genannte Verfahren ähnlich der Clusterbildung auf den Zusammenschlußsen von mobilen Rechnern zu einem Subnetz aufbaut, ist dieser Teil für die Replikation im vorgestellten mobilen Szenarium nicht besonders geeignet.

8.2 Bewertung der Verfahren anhand der Kriterien des mobilen Szenariums

Die Replikationsverfahren sollen nun systematisch anhand der Kriterien des mobilen Szenariums bewertet und verglichen werden. Den Anfang bildet die Konsistenzsicherung.

8.2.1 Konsistenzsicherung

Für die Einteilung der durch die Replikationsverfahren gesicherten Konsistenz sollen die folgenden 5 Punkte dienen:

Vollständige Konsistenz Die Konsistenz wird durch die klassischen pessimistischen Replikationsverfahren gesichert. Die höchste Konsistenzsicherheit bietet das *ROWA*-Verfahren, da schon bei einer einfachen Verbindungsunterbrechung keine Transaktionen auf dem gesamten Fragment möglich sind. Da das Verfahren das restriktivste ist, bietet es im praktischen mobilen Einsatz nur geringen Nutzen und soll hier nur der Vollständigkeit erwähnt werden. Im allgemeinen bieten die *Voting*-Verfahren ebenfalls eine vollständige Konsistenzsicherung, da sie bei richtiger Konfiguration der Stimmen, Gewichte und Strukturen selbst in partitionierten Netzen die replizierten Daten durch Einschränkung bzw. Verbot von Transaktionen auf ihnen konsisten halten können. Ebenfalls dazu gehören Verfahren wie *Missing Writes*, die im Fehlerfall Voting-Verfahren nutzen. Eine Sonderstellung nimmt das *Primary-Missing-Writes*-Verfahren ein, das dieselbe Sicherung der Konsistenz bietet, weil es das einzige Nicht-Voting-Verfahren in dieser Kategorie ist. Für die letzten beiden Verfahren gilt als Einschränkung, daß die Erkennung von Verbindungsfehlern und die vollständige Ausführung des Verfahrenswechsels in allen Fällen möglich sein muß.

Vollständige Konsistenz ohne Netzpartitionierungen Etwas weniger sicher sind Verfahren wie das *Write-all-available*- Verfahren, die Schwierigkeiten bei der Behandlung von Netzpartitionierungen und den sich daraus ergebenden möglichen Differenzen zwischen den replizierten Daten haben. Wenn aufgrund des Einsatzgebietes keine Netzpartitionierungen auftreten können, sind diese Verfahren bezüglich der Konsistenzsicherung ebenso sicher wie die Voting-Verfahren. Auf der gleichen Stufe stehen die *Primary* -Verfahren, wenn auf den getrennten Replikaten jegliche Transaktionen verboten sind, was bei den Grundverfahren zutrifft.

Eingeschränkte Konsistenz Eine Zwitterstellung zwischen den relativ sicheren Verfahren und den ersten Inkonsistenzen nehmen Kombinationen von pessimistischen mit optimistischen Verfahren ein, wie z.B. eine Kombination aus *Write-all-available* und *Data patch* [BD96]. Wenn das optimistische Verfahren alle Inkonsistenzen bei der Reintegration beseitigen kann, ist es ebenso sicher wie die oben genannten Verfahren. Kann es hingegen gar keine Inkonsistenzen bereinigen, ist es nur so sicher wie optimistische Verfahren zwischen 2 Replikaten. Im Regelfall wird es dazwischen liegen.

Inkonsistenzen zwischen 2 Replikaten In der Folge der sinkenden Konsistenz der Replikate entstehen nun mögliche Inkonsistenzen zwischen 2 Replikaten. Solche Inkonsistenzen treten beispielsweise bei der *Virtual Primary Copy*-Methode und einer Variante des *Write-all-available*-Verfahrens auf. Sie entstehen durch die Zulassung von Transaktionen auf Replikaten, die aufgrund fehlender Verbindungseinrichtungen keine Konsistenzsicherung mit den anderen Replikaten desselben Fragmentes durchführen können (was einem rein optimistischen Ansatz gleichkommt). Diese Inkonsistenzen treten nur zwischen 2 Replikaten auf, wodurch der Aufwand einer Reintegration des getrennt arbeitenden Replikates auf Tests und Aktionen zwischen 2 Replikaten beschränkt bleibt, die gegebenenfalls durch den mobilen Nutzer unterstützt werden können. Um den Umfang der zu behebenden Inkonsistenzen in Grenzen zu halten und den (manuellen) Reintegrationsaufwand zu senken, kann zusätzlich ein optimistisches Verfahren herangezogen werden (siehe oben).

Inkonsistenzen zwischen mehreren Replikaten Die nächstgrößeren Inkonsistenzen können zwischen mehreren voneinander getrennt arbeitenden Replikaten auftreten, z.B. bei einer 3fachen Netzpartitionierung (und einem Verfahren, daß

keine Netzpartitionierungen erkennen und behandeln kann) oder dem Einsatz von rein optimistischen Verfahren.
Das nächste Kriterium in mobilen Szenarien ist die Verfügbarkeit.

8.2.2 Verfügbarkeit der replizierten Daten

In diesem Abschnitt wird nur die Verfügbarkeit eines Replikates im unverbundenen bzw. Fehlerzustand betrachtet, da die volle Verfügbarkeit bei einer stabilen Verbindung stets gegeben ist. Die Verfügbarkeit bezieht sich dabei auf die des von den anderen Replikaten getrennten Replikates, da auf den restlichen Replikaten noch Transaktionen möglich sein sollten. Bei der Verfügbarkeit sind die optimistischen Verfahren den pessimistischen überlegen. Dabei sind 3 Stufen unterscheidbar:

Hohe Verfügbarkeit aller Replikate auch im Fehlerfall Die höchste Verfügbarkeit bieten die optimistischen Verfahren, so zum Beispiel *Transformation log* und *Data patch*. Die Verfahren erlauben Transaktionen auf allen Replikaten, auch denen, die nicht mit den anderen verbunden sind. Die einzige Störung dieser Transparenz entsteht sich bei der Reintegration.

Geringe Verfügbarkeit im Fehlerfall Hier lassen sich alle Verfahren einordnen, die sowohl pessimistische als auch optimistische Replikationsstrategien verwenden, wie z.B. eine Kombination aus *Write-all-available* und *Data patch*. Die Verwendung eines optimistischen Verfahren garantiert auch hier die Möglichkeit der Arbeit sowohl auf den untereinander erreichbaren wie auch den nicht mehr verbundenen Replikaten. Da die Behandlung von schwacher Konsistenz auch ein optimistisches Verfahren darstellt, sind auch die *Clusterbildung* und die *MAID-Replikation* hier zuzuordnen.

Keine Verfügbarkeit im Fehlerfall In diese Gruppe gehören alle Verfahren, die rein pessimistisch arbeiten, so die klassischen Verfahren (*ROWA*, *Write-all-available*, *Primary*-Verfahren, *Voting*), und die sich dem Fehlerfall anpassenden Verfahren, wie *Missing Writes* und *Primary Missing Writes*.
Die Bewertung anhand von Konsistenzsicherung und Verfügbarkeit hat bereits gezeigt, daß sich im Fehlerfall beide Forderungen gegenseitig ausschließen. Diese Aussage gilt bereits für ein festnetzbasierten VDBMS. Beim Einsatz in mobilen Szenarien kommt noch die Kostenminimierung für die Mobilnetze hinzu.

8.2.3 Kommunikationskosten

Die Kommunikationskostenminimierung ist besonders für den mobilen Nutzer wichtig. Die Dauer und Häufigkeit einer Verbindung wird von dem verwendeten Replikationsverfahren und der Arbeitsweise des mobilen Nutzers bestimmt. Als Zusatzannahme sei hier vereinbart, daß die Zeitdauer von der Anforderung einer Verbindung bis zur tatsächlichen Bereitstellung zwar gering ist, aber im allgemeinen eine zu hohe Verzögerung von Transaktionen über diese Verbindung mit sich bringt. So ist ein Auf- und Abbau der nötigen Verbindungen um eine einzelne Transaktion herum nur selten praktikabel.

Im Gegensatz zu den bereits genannten Kriterien lassen sich die Kosten nicht ohne weiteres als Maßstab verwenden. Um einen Vergleich von Replikationsverfahren in diesem Punkt zu ermöglichen, soll im folgenden zwischen den folgenden Aspekten unterschieden werden :

- Gesamtumfang der entstehenden Kommunikationskosten (Abschätzung)

- Umfang der erfolgten Kostenminimierung innerhalb der für das Verfahren bzw. die Verfahrensklasse sinnvollen Grenzen (Abschätzung)

Bei den *Gesamtkosten* handelt es sich um eine relative Größe, die dem Vergleich der Verfahren untereinander dient. Die *Kostenminimierung* beinhaltet eine Abschätzung der erfolgten Minimierung der Kommunikationskosten. Als Maximum der Kosten kann eine stets vorhandene und verfügbare Verbindung angenommen werden. Das Minimum kann einerseits eine stets abgeschaltete Verbindung darstellen, welche andererseits keinen praktischen Sinn hat, da eine Reintegration über die Verbindungseinrichtung in jedem Fall stattfinden muß. Das Minimum für die pessimistischen Verfahren besteht in einer Verbindung, die nur dann Kosten verursacht, wenn sie benutzt wird. Dagegen ist das Minimum für die optimistischen Verfahren eine Verbindung, die nur zur Reintegration genutzt wird. Das Ergebnis der Kostenminimierung kann grob in den folgenden Stufen eingeteilt werden:

Keine Kostenminimierung Diese Kosten werden von einer permanenten Verbindung mit den anderen Replikaten verursacht. Dieser Zustand ähnelt bis auf die geringere Bandbreite einem festnetzbasierten VDBMS. Replikationsverfahren, die diese Art von Verbindung fordern, sind pessimistischer Natur wie die klassischen Verfahren (*ROWA*, *Write-all-available*, *Primary-* und *Voting*-Verfahren) und neue Enwicklungen, wie *Missing Writes* und *Primary Missing Writes*.

Teilweise Kostenminimierung Um die Verbindung nicht permanent nutzen zu müssen, wird z. B. bei der *Virtual Primary Copy* - Methode der **Home Base Node** (HBN) als Transaktionscache genutzt. Diese Methode funktioniert nur bei Read-only-Transaktionen auf dem Fragment des Mobilrechners zuverlässig. Wenn zu verarbeitende Transaktionen auch schreiben wollen, müssen Cachevalidierungsverfahren eingesetzt werden, wie im **Coda**-Filesystem ([SKB+93]).

Vollständige Kostenminimierung Eine umfangreiche Verringerung der Notwendigkeit einer permanenten Verbindung kann durch den Einsatz von kombiniert pessimistischen und optimistischen Replikationsverfahren erreicht werden. Dazu gehören z.B. eine Kombination aus *Write-all-available* und *Data patch*, die *Virtual Primary Copy*-Methode (ohne Transaktionscache) und die Verfahren mit abgeschwächter Konsistenz, wie die *Clusterbildung* und die *MAID-Replikation*. Dadurch werden Verbindungen nur dann genutzt, wenn es zur Reintegration der getrennten Replikate kommt. Die Ergebnisse der Einordnung der Kritieren werden im folgenden in einer Tabelle kurz zusammengefaßt.

8.3 Bewertung der Verfahren - Zusammenfassung

Um einen Maßstab für die Erfüllung der Kriterien zu geben, sollen folgende Intervalle gelten:

- Konsistenzsicherung : 0 ... 4 (keine Inkonsistenzen - viele Inkonsistenzen)
- Verfügbarkeit : 0 ... 2 (keine Verfügbarkeit - volle Verfügbarkeit)
- Kosten gesamt : 1 ... 10 (geringe Kosten - hohe Kosten)
- Kostenminimierung : 0 ... 2 (keine Minimierung - maximal mögliche Minimierung)

Anhand der Kriterien für das Replikationsverfahren im mobilen Szenarium werden die Verfahren in der folgenden Tabelle 8.1 eingeordnet. Sie berücksichtigt, wie im Maßstab schon erkenntlich, die Aufteilung nach Gesamt- und minimierten Kosten.

Verfahren	Konsistenzsicherung	Verfügbarkeit	Kosten	Kostenminimierung
ROWA	4+	0	5	0
Write-all-available	3	1	5	0
Primary site/ Primary copy	3	1	3	0
Voting	4	1	10	$0/2^1$
Missing Writes	4	1	$5/10^2$	$0/2^2$
Primary Missing Writes	3	1	3	0
ADR	3	1	3	2+
Hybridverfahren[3]	0-3	2	0-2	0-2
VPC	2	2	3	1-2
Clusterbildung	2	2	?0	1-2
MAID-Replikation	2	2	?	1-2
Transformation log	0	2	0	1-2
Data patch	0	2	0	1-2

[a] Die einfachen Voting-Verfahren bieten keine Minimierung, die Struktur- und Weighted-Voting-Verfahren können an den Einsatzweck angepaßt werden und die Kosten minimal halten

[b] Fehlerfrei/Fehlerhaft

[c] Damit sind pessimistische Verfahren gemeint, die zur Erhöhung der Verfügbarkeit optimistische Verfahren für getrennt arbeitende Replikate nutzen, wie die bereits erwähnte Kombination aus *write-all-available* und *data patch*

Tabelle 8.1: Die Replikationsverfahren im Vergleich

Wie erkennbar ist, existiert kein Verfahren, daß alle 3 Kriterien gleichzeitig erfüllt. Die Erfüllung der Konsistenzsicherung und der Kostenminimierung stehen im direkten Widerspruch zu einer hohen Verfügbarkeit der Replikate. Um diesen Konflikt teilsweise lösen zu können, ist es notwendig, Kompromisse in Bezug auf die Kriterien einzugehen. Dazu soll im folgenden ein Verfahren ausgewählt werden, welches bereits gute Voraussetzungen für den Einsatz im beschriebenen mobilen Szenarium besitzt. Dieses Replikationsverfahren wird dann teilweise erweitert werden.

Teil III

Vorstellung und Bewertung des erweiterten Replikationsverfahrens

Kapitel 9

Das gewählte Verfahren und seine Erweiterungen

In den vorangegangen Kapiteln wurde deutlich, daß es quasi kein Replikationsverfahren geben kann, das gleichzeitig die Forderungen nach **Konsistenzsicherung**, **hoher Verfügbarkeit** und **Minimierung der Kommunikationskosten** erfüllen kann. Um nun eines der vorgestellten Verfahren an die Anforderungen der mobilen Arbeit anzupassen, sind Kompromisse in wenigstens einer der 3 Forderungen nötig. Die sich daraus ergebenden Konsequenzen sollen kurz im folgenden Abschnitt dargelegt werden.

9.1 Konsequenzen aus Kapitel 8

Die Möglichkeiten und Erweiterungen von Replikationsverfahren bei der Abschwächung der genannten 3 Forderungen werden im folgenden zusammengefaßt. Die am leichtesten zu realisierende Abschwächung ist die der Verfügbarkeit.

9.1.1 Verringerung der Verfügbarkeit

Dieser Ansatz wird von den pessimistischen Verfahren gewählt. Sobald ein Replikat nicht mehr in der Lage ist, die Änderungen in den anderen Replikaten seines Fragmentes nachzuvollziehen, führt das im günstigsten Fall dazu, daß alle anderen Replikate weiterarbeiten und weitere Transaktionen nicht mehr an das ausgefallene Replikat weitergeleitet werden. Weiterhin ist es nötig, die Änderungen, die das ausgefallene Replikat nicht mehr übernehmen konnte, aufzuzeichnen, um sie zum Reintegrationszeitpunkt dem dann veralteten Replikat mitzuteilen. Außerdem muß der Knotenrechner, der das Replikat hält, entsprechende Transaktionen auf dem Replikat verbieten bzw. einschränken, wenn er dazu in der Lage ist. Wenn das nicht der Fall sein sollte, kann davon ausgegangen werden, daß keine Zugriffe auf das Replikat mehr möglich sind.

Dieser Ansatz ist relativ einfach und im Hinblick auf die Konsistenzsicherung sicher zu realisieren (keine Reintegrationskonflikte). Da aber die Verfügbarkeit ein entscheidendes Maß für die Nutzbarkeit mobiler VDBMS darstellt, stellt eine Verringerung der Verfügbarkeit eine verhältnismäßig große Einschränkung der Arbeit des Nutzers dar.

9.1.2 Höhere Kommunikationskosten

Die stärkere Nutzung der Funknetze hinsichtlich der **Dauer** und **Häufigkeit** der Verbindungen erlaubt es, die Konsistenzsicherung und die Verfügbarkeit zu erhöhen. Die Situation in einem festnetzbasierten VDBMS kann nicht erreicht werden, da die Bandbreiten der Funknetze kleiner sind als die von LANs und WANs. Zu unterscheiden sind also :

- **Zeitlich längere Verbindungen**
  Damit kann eine Festnetz-ähnliche Situation geschaffen werden. Während eines Großteiles der Arbeitszeit ist die Funknetzverbindung verfügbar. Somit können besonders pessimistische Verfahren ohne Verzögerungen (bis auf die eingeschränkte Bandbreite) wirkungsvoll arbeiten. Eine Grenze der Verbindungsdauer ist durch die Kosten der Verbindung gegeben. Angepaßt auf die Situation eines mobilen Nutzers würde das im Extremfall bedeuten, daß die Funknetzverbindung während aller Arbeitsperioden des mobilen und der Festnetznutzer verfügbar ist. Im allgemeinen ist diese Lösung zu teuer und ineffizient, da die Verbindung zum Großteil der Zeit nicht ausgelastet ist.

- **Zeitlich häufigere Verbindungen**
  Diese Möglichkeit erfordert relativ hohen Planungsaufwand. Eine Verbindung wird dann eröffnet, wenn ein genügend hohes Datenvolumen ausgetauscht werden soll. Dazu werden Transaktionen auf dem lokalen Replikat und den zugeordneten HBN so lange gesammelt, bis sich die Übertragung lohnt. Der Vorteil besteht in der relativ hohen Auslastung der Verbindung und der geringen Wahrscheinlichkeit von Störungen während der Übertragung. Dagegen sprechen die Verzögerungen von Transaktionen, die die Arbeit des Nutzers in großem Umfang behindern können. Des weiteren wird die Funknetzverbindung bis an die Grenze ihrer Leistungsfähigkeit genutzt, Fehler bei der Übertragung wirken sich stärker aus.

Wenn die **Dauer** der Funkverbindung sehr gering gehalten und die Häufigkeit von Verbindungen vergrößert wird, sind die Kosten etwas höher als im minimal denkbaren Fall, aber die Verbindungen werden effektiver ausgelastet als bei einer Erhöhung der Verbindungsdauer. Das begünstigt die Nutzung von pessimistischen Replikationsverfahren, die dadurch in der Lage sind, die nötigen Updates sowie die Sperrverwaltung zum Teil überhaupt bzw. mit definierbaren Verzögerungen durchzuführen.

9.1.3 Verringerung der Konsistenz

Diese Möglichkeit nutzen die optimistischen Verfahren. Um die Zahl der Inkonsistenzen zu verringern, können die beim Datenbankentwurf entstandenen Informationen genutzt und dort zu den Schemainformationen hinzugefügt werden (siehe **Data patch**). Die Effektivität dieser Verfahren hängt, wie bereits erläutert, stark von der Art und Anzahl der zu erwartenden Inkonsistenzen ab. Deshalb eignen sich die optimistischen Verfahren hauptsächlich dort, wo der Umfang der tatsächlich gleichzeitig zu bearbeitenden Transaktionen gering ist und wenige bzw. einfach aufzulösende Konflikte zu erwarten sind oder wo die Menge des gesammelten Wissens über die zu bearbeitenden Transaktionen ausreichend ist.

9.1.4 Zusammenfassung

Aus den vorangegangen Abschnitten wurden folgende Folgerungen gezogen, wobei unter *externem* Wissen eine Menge von Fakten oder Annahmen verstanden werden soll, die auf irgendeine Art aus der Einsatzumgebung des Replikationsverfahrens gewonnen wurden.

1. Weder optimistische noch pessimistische Verfahren sind in ihrer reinen Form für den Einsatz in mobilen Umgebungen geeignet.

2. Die pessimistischen Replikationsverfahren sind für VDBMS in mobilen Szenarien nur dann geeignet, wenn sie die Arbeit des mobilen Nutzers nur in geringem Maße einschränken. Das ist dann der Fall, wenn externes Wissen über die Arbeitsweise des mobilen Nutzers vorliegt und diese Arbeitsweise so beschaffen ist, daß sie die Nutzung eines ausschließlich pessimisischen Verfahrens rechtfertigt.

3. Die optimistischen Replikationsverfahren sind für allgemein gehaltene Transaktionen dann geeignet, wenn sie so mit externen semantischem Wissen angereichert werden können, daß es nur zu wenigen manuell zu lösenden Konflikten bei der Reintegration kommt. Es ist anzunehmen, daß das nur in wenigen Spezialfällen möglich ist.

4. Die für den Einsatz in fehlerbehafteten festnetzbasierten VDBMS am besten geeigneten Verfahren sind nach [AS97] die Voting-Verfahren. Deren Nachteil ist eine relativ hohe Netzbelastung, welche den Einsatz in mobilen Netzen stark erschwert. Hinzu kommt die durchschnittlich zu geringe Verfügbarkeit von stimmberechtigten mobilen Rechnerknoten.

5. Es scheint kein Verfahren zu existieren, daß ohne externes Wissen auskommt und die Kriterien des mobilen Szenariums (**Konsistenzsicherung**, **Verfügbarkeit** und **Kostenminimierung** erfüllen kann. Im folgenden werden einige Ansätze vorgestellt, mit deren Hilfe Replikationsverfahren an mobile VDBMS angepaßt werden können.

9.2 Lösungsansätze

Ein Versuch, dieses Problem zu umgehen, kann darin bestehen, ein hybrides Verfahren einzusetzen, das mit ausreichendem externen Wissen ausgestattet wird und so sehr speziell an die zu bearbeitende Aufgabe angepaßt ist. Dabei teilt sich das Wissen in zwei Komponenten auf:

9.2.1 Wissen aus dem *mobilen Szenarium*

Es ist dadurch gekennzeichent, daß es relativ konstant bis gering veränderlich ist, weil die Annahmen, unter denen es gewonnen wurde, für den allgemeinen Problemkreis der mobilen Arbeitweise gelten und nicht die Arbeit selbst beschreiben, aber durch die Festlegung eines Szenariums genug Hintergrundinformationen liefern. Dazu ist es notwendig, das Replikationsverfahren mit einer Schnittstelle zur Erfassung von externem Wissen auszustatten. Diese Schnittstelle kann sowohl von dem Nutzer selbst über eine Nutzerschnittstelle als auch dem Anwendungsprogramm, das auf den Daten des Replikates arbeitet, sowie dem Betriebssystem des Mobilrechners benutzt werden. Dabei könnten folgende Daten (auch statistisch) erfaßt werden :

1. Vom Betriebssystem :

   - Zustand der Funkverbindung (verbunden, nicht verbunden, Fehlerhäufigkeit, Datenrate)
   - Zeitdauer ohne Funkverbindung
   - vorraussichtliche Zeit bis zur nächsten Abschaltung

2. Vom Anwendungsprogramm :

- Zeit seit der letzten Aktualisierung des Replikates
- Menge der veralteten Daten (für Update)
- aktuelle Arbeitsweise (Lesen, Schreiben)

3. Vom Anwender (über Präferenzen) :
   - Sofortiges Update bei einem Wiederaufbau der Funkverbindung ?
   - Zeit bis zur nächsten Abschaltung der Verbindung
   - Zeit bis zur nächsten Abschaltung des Mobilrechners
   - Zeit bis zur nächsten Aktivierung der Verbindung
   - Zeit bis zur nächsten Aktivierung des Mobilrechners
   - Vorgehensweise bei kuzzeitiger Verbindungsunterbrechung (optimistisch, pessimistisch)
   - Vorgehensweise bei langfristiger Verbindungsabschaltung (optimistisch, pessimistisch, Verfahrenswechsel)

Die Daten von mobilen Nutzern sind sowohl im Dialog als auch über ein Nutzerprofil ermittel- bzw. abschätzbar. Folgende Vorgänge sind beim Einschalten des Rechners und der Funkverbindung denkbar : Zuerst informiert das Betriebssystem das Replikationsprogramm über das Wiedereinschalten des Mobilrechners bzw. des Funkmodems. Das Anwendungsprogramm informiert das Replikationsverfahren über die Wiederaufnahme der Arbeit auf einem lokalen Replikat. Der Benutzer kann dem Replikationsverfahren mitteilen, ob er Schreibzugriffe tätigen wird bzw. welche Updateintervalle er bei Lesezugriffen auf dem Replikat akzeptiert. Wechselt der Nutzer während einer geöffneten Funkverbindung die Arbeitsweise, kann das dem Replikationsverfahren durch den Nutzer selbst oder das Anwendungsprogramm mitgeteilt werden. Beabsichtigt der Nutzer ein Abschalten der Funkverbindung oder des Mobilrechners, so informiert er davor das Replikationsverfahren, welches ihm im Dialog die weiteren Entscheidungen für die Konsistenzüberwachung überläßt. Eine Alternative, die allerdings spezielle Hardware voraussetzt, besteht in entsprechenden Reaktionen des Mobilrechners auf einen Versuch des mobilen Nutzers, das Gerät auszuschalten. Folgende Optionen stehen dem Benutzer offen :

1. Beenden ohne Konsistenzüberwachung (rein optimistisch, kein Verfahren)
2. Beenden mit optimistischer Konsistenzüberwachung (welches Verfahren ?)
3. Beenden mit pessimistischer Konsistenzüberwachung (welches Verfahren ?)
4. Beenden, Fragment wird auf beiden Seiten der Funkverbindung nur noch als read-only-Kopie verwendet.
5. Beenden, Fragment wird über lokales Replikat als read-write-Kopie verwendet.
6. Beenden, Fragment wird über entfernte Replikate von anderen Nutzern als read-write-Kopie verwendet
7. Zusätze : Welche Transaktionen sind auf den anderen Replikaten erlaubt ?
8. Absprache mit anderen Nutzern desselben Fragmentes

Wie schon in bei der Vorstellung des Verfahrens in [PB95] beschrieben, kann der mobile Nutzer aufgefordert werden, dem Replikationsverfahren mitzuteilen, wann er die Funkverbindung abschalten will. Außerdem können Informationen über die Art der in der Zeit bis zur nächsten Verbindung anfallenden Transaktionen auf dem

lokalen und anderen Replikaten genutzt werden, z.B. ob es sich primär um read-only-Transaktionen handeln wird, auf welchen Replikaten Schreibtransaktionen zu erwarten sind und wie lange die voraussichtliche Unterbrechung dauern wird. Damit kann der mobile Nutzer selbst die Art der Konsistenzsicherung im verbindungslosen Zustand für alle erreichbaren Replikate festlegen. Er kann z.B. die Entscheidung treffen, ob es überhaupt nötig ist, die im verbindungslosen Zustand seines Replikates an den vom Festnetz aus erreichbaren Replikate ankommenden Transaktionen mitzuloggen, weil er selbst bereits weiß, ob es zu Konflikten kommen wird. Diese Entscheidung würde ein optimistisches Verfahren zum Einsatz bringen können. Ebenso kann er entscheiden, auf seiner lokalen Kopie weiterzuarbeiten, ohne Verbindung zum Festnetz und den von dort aus erreichbaren Replikaten zu haben. Das könnte dazu führen, daß auf allen anderen Replikaten vor der Trennung Schreibsperren mit einem Zeitstempel gesetzt werden, so daß nur noch lesende Transaktionen auf diesen Replikaten ablaufen können, denen außerdem noch der letzte aktuelle Zeitpunkt der gelesenen Daten mitgeteilt werden kann. Dazu ist unter Umständen eine Absprache mit anderen Nutzern bzw. deren Präferenzen nötig (siehe oben Punkt 8). Dieses Wissen ist vorrangig semantischer Natur, da es sich aber auf eine relativ konstante Umgebung bezieht (das mobile Szenarium), ist es einfacher zu spezifizieren als problemspezifisches externes Wissen.

9.2.2 *Problemspezifisches* Wissen

Die zweite Art von externem Wissen kann nur aus der konkreten Einsatzsituation des mobilen VDBMS ermittelt werden. Dabei kommt es im Unterschied zum Wissen über die mobile Arbeit darauf an, möglichst genau zu erfassen, welche Transaktionen ablaufen werden, was diese Transaktionen zu bedeuten haben, welche Transaktionen auf demselben Fragment ablaufen können, ohne sich gegenseitig zu beeinflussen, ob und inwieweit es bei der Reintegration zu Konflikten kommen kann und welche Aktionen im Konfliktfall ausgeführt werden können, um die Konflikte automatisch und ohne Zuhilfenahme eines Nutzers beseitigen zu können. Mit Hilfe dieses Wissens kann ein optimistisches Replikationsverfahren in einem mobilen VBDMS eingesetzt werden (**Data patch**), allerdings wird der Wissensumfang nur in sehr wenigen Fällen so umfangreich sein, daß ein so konfiguriertes Verfahren effizient arbeitet. Der Einsatz dieses Wissens legt ein Replikationsverfahren auf ein einziges Einsatzgebiet fest. Wenn sich das Einsatzgebiet ändert, muß sich auch das Replikationsverfahren ändern.

Zusammenfassend ist zu sagen, daß der Einsatz von Replikationsverfahren in mobilen Umgebungen erst ab einer bestimmten Menge an externem Wissen sinnvoll ist. Im folgenden werden zwei Replikationsverfahren vorgeschlagen, die mit unterschiedlichem Wissen entworfen wurden.

9.3 Entwurf für ein Replikationsverfahren für den Einsatz in mobilen Szenarien

Für dieses Replikationverfahren wird hauptsächlich Wissen aus dem Bereich der mobilen Arbeit verwendet. Folgende Annahmen werden gemacht :

- Das Szenarium entspricht dem in Kapitel 2 vorgestellten mit der Einschränkung, daß nur Einzelrechner über die Verbindungseinrichtungen mit dem Festnetz kommunizieren können, keine Zweitnetze.
- Die Replikate werden sowohl von Mobilrechnern als auch von Rechnern im Festnetz gehalten und benutzt.

- Der mobile Nutzer des VDBMS ist kooperativ und informiert das Programm über seine Arbeitsweise und zu erwartende Änderungen im Systemzustand.

Das gewählte Verfahren besteht aus einer Kombination des *Primary site*- Verfahrens mit mehreren optimistischen Replikationsverfahren. Dazu werden die Primärkopien aller Replikate auf einem oder mehreren Rechnern des Festnetzes gehalten, wodurch ihre Verfügbarkeit gegenüber den mobilen Replikaten ausreichend hoch ist.
Folgende Situationen werden unterschieden:

Verbindungsaktiver Zustand Dieser Zustand ist durch eine vorhandene und aktivierte Verbindungseinrichtung zwischen dem Mobilrechner und der Primärkopie im Festnetz gekennzeichnet. Im einfachsten Fall ist dieser Zustand mit einem Festnetz-VDBMS gleichzusetzen. Zusätzlich kann auf beiden Seiten gleichzeitig jeweils ein Transaktionscache arbeiten, der abhängig von der vorhandenen Bandbreite, dem Transaktionsvolumen und einer Wissensbasis für die Arbeit des mobilen Nutzer Transaktionen auf den Replikaten sammelt und bündelt. Ist eine bestimmte Zeit verstrichen oder das Datenvolumen ausreichend groß, wird die Verbindung geöffnet. Die Daten werden übertragen und die Transaktionen ausgeführt. Dann wird die Verbindung wieder geschlossen.

Übergang in den verbindungslosen Zustand - Trennung vom Festnetz Wenn eine Trennung des Mobilrechners vom Festnetz bevorsteht, wird das Replikationsverfahren benachrichtigt. Das kann durch das Betriebssystem oder den Nutzer des Mobilrechners geschehen. Abhängig von den Einflußfaktoren der mobilen Arbeit kann nun die Art der Konsistenzüberwachung für den zu erwartenden verbindungslosen Zustand gewählt werden. Dabei kann der Nutzer bzw. das Anwendungsprogramm zwischen folgenden Modi wählen :

- Keine Konsistenzüberwachung - rein optimistischer Ansatz mit Zeitstempeln
- Konsistenzüberwachung durch *Data-patch* und *Transformation log*
- Aufteilung der möglichen Transaktionen : lokales Replikat lesen, andere Replikate lesen
- Aufteilung der möglichen Transaktionen : lokales Replikat lesen und schreiben, andere Replikate nicht nutzen
- Aufteilung der möglichen Transaktionen : lokales Replikat nicht nutzen, andere Replikate schreiben

Verbindungsloser Zustand Im verbindungslosen Zustand ist keine Verbindung mit dem Festnetz möglich. Je nach den Ergebnissen der Auswertung der bei der Abschaltung gewählten Modi kann auf den Replikaten gelesen bzw. geschrieben werden.

Übergang in den verbindungsaktiven Zustand - Reintegration Wird die Verbindung zum Festnetz wieder aktiviert, so gleichen der Mobilrechner und der *Primary* ihre Replikate ab. Dazu ist unter Umständen das Hinzuziehen des Nutzers erforderlich. Sobald die Reintegration abgeschlossen ist, geht das Replikationsverfahren in den verbindungsaktiven Zustand über.

9.4 Erweiterungen für den Einsatz im Waldbeispiel

Das Replikationsverfahren ist mit dem oben beschriebenen identisch. Zusätzlich werden speziell angepaßte Varianten des **Data patch** und des **Transformation log**-Verfahren verwendet, deren Einsatz sich nach der Art der Informationsverarbeitung durch den mobilen Nutzer richtet, die als externes Wissen vorliegt. Beispielsweise wären getrennte Verfahren für die Routineüberprüfungen, die Erfassung einzelner oder umfassenderer Schäden, die Verfolgung von Schadensentwicklungen und für bestimmte, festzulegende Regionen wie Reviere, Abteilungen, Unterabteilungen und Teilflächen einsetzbar. So können die optimistischen Verfahren mit Informationen über die zu erwartenden Transaktionen erweitert werden. Dazu wird von folgenden Annahmen ausgegangen:

- Das **Walddatenschema** der Datenbank ist an die Arbeitsweise der mobilen Nutzer angepaßt. Es werden die Datensätze repliziert, die mehrere Mobilrechner nutzen (Randgebiete eines Revieres). Die Kerngebiete sind nicht über die Mobilrechner repliziert, da die Daten dieser Flächen für die anderen Reviere nicht wichtig sind. Damit wird schon beim Datenbankentwurf versucht, den Umfang von Konflikten zwischen den Replikaten zu verringern. Hinzu kommt, daß jedes Fragment noch einmal auf der **Primary site** repliziert wird. Somit existieren für die Randgebiete relativ viele Replikate, für die Kernflächen nur 2.

- Die **Struktur des Netzes** wird wie folgt angenommen:

  - Alle Rechnerknoten oberhalb der Revierebene (siehe Abb. 3.1) sind in einem Festnetz zusammengefaßt.
  - Jedem Revier ist ein Rechnerknoten als *HBN* und ein Mobilrechner zugeordnet.
  - Es existiert ein **Primary** im Festnetz.
  - Zwischen den *HBN* und den Rechnern des Festnetzes existiert eine schnelle und sichere Verbindungseinrichtung. Das wird in der Mehrzahl der Fälle kein LAN sein können. Denkbar ist eine Wählmodemverbindung.
  - Es existieren jeweils Funkverbindungen zwischen den *Home Base Nodes* und den Mobilrechnern.

  Eine **alternative Struktur**, die ein umfangreicheres Funknetz erfordert und höhere Kommunikationskosten verursacht, dafür aber eine bessere Konsistenzkontrolle ermöglicht, könnte wie folgt aussehen:

  - Alle Rechnerknoten oberhalb der Revierebene sind wie oben miteinander in einem Festnetz verbunden.
  - Zu jedem Revier gehört nur noch ein Mobilrechner und kein Rechnerknoten.
  - Die *HBN* der Mobilrechner sind Teil des Festnetzes.
  - Es existieren jeweils Weitstreckenfunkverbindungen zwischen den *HBN* und Mobilrechnern.

- Die **Arbeitsweise** kann z.B. so definiert werden:

  1. Der Nutzer entscheidet, was für ein Revier er aufsuchen will.

2. Der Nutzer lädt alle Daten der umliegenden Reviere (optional) und seines zu besuchenden Revieres (wenn nicht schon vorhanden) vom *HBN* auf seinen Mobilrechner. Das kann einerseits über eine kurzzeitige Funknetzverbindung geschehen, die dann voll ausgelastet wird, andererseits kann sich der Nutzer die Daten auch über ein LAN holen, wenn das möglich ist (Ausgangspunkt der Arbeit ist der Ort des **HBN**).
3. Der Nutzer befindet sich im zu erfassenden Revier und vergleicht die alten Daten des Revieres mit dem aktuellen Zustand. Dabei werden die neuen Daten eingegeben oder im Falle der Fortführung eines bereits erfaßten Schadens die alten Daten erweitert. Je nach Wissen über die Arbeit in anderen Revieren und auf dem **HBN** kann es sinnvoll sein, einerseits die neuen Entwicklungen im eigenen Revier an den **HBN** zu schicken, andererseits die Änderungen in den dort gehaltenen Daten auf den Mobilrechner zu übertragen.
4. Zur Erkennung von möglichen Entwicklungen auf den Teilflächen der umliegenden Reviere und um die Ausbreitung der Schäden rechtzeitig untersuchen zu können, kann der mobile Nutzer Daten aus den Teilflächen der angrenzenden Gebiete lesen und eventuell auch schreiben. Da diese Daten parallel durch einen anderen mobilen Nutzer erarbeitet werden, kann es notwendig sein, eine Synchronisation der Replikate über die Funknetzverbindung durchzuführen.

Unter Berücksichtigung der vorgestellten Möglichkeiten und Erweiterungen des **primary site**-Verfahrens soll nun eine Bewertung anhand der Kriterien für das mobile Szenarium vorgenommen werden.

9.5 Bewertung

Um die Vergleichbarkeit zu den bereits vorgestellten Verfahren zu ermöglichen, sollen im folgenden das Replikationsverfahren für das allgemeine mobile Szenarium und das Walddatenszenarium anhand der bereits vorgestellten Kriterien *Konsistenzsicherung*, *Verfügbarkeit* und *Kostenminimierung* ähnlich wie in Kapitel 8 bewertet werden (Tab. 9.1).

Verfahren	Konsistenzsicherung	Verfügbarkeit	Kosten	Kostenminimierung
allgemein-mobil	0-2	2	0-1	1
Walddaten	1-2	2	0-1	1-2

Tabelle 9.1: Die erweiterten Replikationsverfahren im Vergleich

Im Vergleich mit den bereits vorgestellten Hybridverfahren zeigen sich keine umfangreichen Änderungen in Bezug auf die Ausgangsverfahren. So können lediglich die Gesamt-Kommunikationskosten verringert werden; bei dem walddatenspezifischen Entwurf kommt noch eine Minimierung der Kommunikationskosten hinzu. Die sonst stattfindenden negativen Auswirkungen dieser Kostenverringerung auf die Konsistenzsicherung und die Verfügbarkeit werden hier durch die Anwendung von mehreren Annahmen (externes Wissen über die Arbeitsweise des Nutzers) vermindert.

Bewertung des Verfahrens für das allgemeine mobile Szenarium Die Forderung nach hoher Verfügbarkeit der mobilen Replikate könnte vom Verfahren in gewissem Umfang erfüllt werden. Das kann nur durch eine Verringerung der Konsistenz ermöglicht werden. Gleichzeitig wird aber durch die Eigenschaften des vorgestellten mobilen Szenariums und die Annahmen über den Netzaufbau die Konsistenzsicherung verbessert, da keine Netzpartitionierungen auftreten können und die einzigen Reintegrationskonflikte aus der Verbindung eines Mobilrechners mit dem Festnetz heraus entstehen.

Bewertung des Verfahrens für das Walddatenszenarium Durch die potentielle Verringerung der Konsistenzsicherung, die durch die Anwendung von optimistischen Verfahren bedingt ist, könnte die Verfügbarkeit etwas erhöht werden. Die zu erwartende Steigerung richtet sich nach dem Umfang und der Genauigkeit des externen Wissens über die Arbeitsabläufe, die ein mobiler Nutzer ausführt, was Gegenstand von Forschungen im Projekt MoVi ist. Es ist anzunehmen, daß solches Wissen im erforderlichen Umfang nur durch eine sehr aufwendige Analyse der Arbeitsvorgänge ermittelt werden kann. Das schränkt die Brauchbarkeit des Einsatzes dieses Verfahrens in hohem Maße ein.

Um einen Einblick in die Abläufe in einem Replikationsverfahren zu gewinnen und in gewissen Umfang Vergleichsmöglichkeiten für den Einsatz von Replikationsverfahren in mobilen Umgebungen zu erhalten, wurde ein Simulationssystem entworfen und implementiert, das im nächsten Kapitel vorgestellt werden soll.

Teil IV

Die Simulation von Replikationsverfahren

Kapitel 10

Die Simulationsumgebung

Um ein Replikationsverfahren implementieren und testen zu können, wurde eine Simulationsumgebung entworfen, die einige der für das mobile Szenarium wichtigen Parameter enthält. Insbesondere sollen die Auswirkungen der schwankenden Verfügbarkeit von Funkverbindungen und Mobilrechner auf den Ablauf der Transaktionen in VDBMS bestimmt werden.

10.1 Diskrete ereignisorientierte Simulation

Das Ziel der Simulation besteht darin, durch die Modellierung von einigen Aspekten des Originalsystems ein Modellsystem zu schaffen, das wesentliche Eigenschaften des Originals besitzt, um durch die Analyse des Verhaltens des Modellsystems Erkenntnisse über das Verhalten des Originals zu erhalten. Damit das Modell in den für die zu erwartenden Aussagen wichtigen Aspekten für das Original repräsentativ ist, müssen die dafür nötigen Parameter des Originals bestimmt werden. Der wichtigste Parameter ist hier die Zeit, da das Verhalten des Systems in der Zeit betrachtet werden soll. Das Verhalten umfaßt hier hauptsächlich die Erzeugung und Verarbeitung von Lese- und Schreibaktionen auf den replizierten Daten. Dabei beeinflussen die Eigenschaften der zugrundeliegenden Verbindungseinrichtungen und Rechnerknoten den zeitlichen Ablauf der Transaktionen. Die Aufgabe besteht darin, das mobile Szenarium auf ein Modell zu übertragen, in dem diese Einflüsse untersucht werden können. Dazu wird vereinfacht angenommen, daß sich der Systemzustand im Zeitraum zwischen dem Auftreten von Ereignissen, wie dem Empfang und dem Versand von Nachrichten, nicht ändern kann. Deshalb ist die Zeit, die zwischen dem Auftreten solcher Ereignisse verstreicht, für das Verhalten des Systems nicht von Interesse. Dadurch wird die Zeit im Simulationslauf nicht als kontinuierliche, sondern als diskrete Größe realisiert, d.h. die laufende Simulationszeit überspringt die Zeitintervalle ohne Ereignisse. Für die Realisierung einer zeitdiskreten Simulation existieren verschiedene Ansätze, die unterschiedliche Betrachtungsweisen auf das zu simulierende System nutzen. Dazu gehören:

1. der ereignisorientierte Ansatz
2. der prozeßorientierte Ansatz
3. der transaktionsorientierte Ansatz
4. der aktivitätsorientierte Ansatz

Aufgrund folgender Vorzüge wurde der ereignisorientierte Ansatz gewählt:

- Der Ansatz eignet sich besonders für die Simulation von Bedienungs-/Wartesystemen und ist daher für ein VDBMS geeignet.
- Der Ansatz gehört zu den *materialorientierten* Ansätzen; dabei steht der Fluß der Ereignisse bzw. Materialien während der Simulation im Vordergrund.
- Der Zeitpunkt des Auftretens eines Ereignisses ist beliebig wählbar. Dies stimmt mit dem Verhalten von Nachrichten in einem VDBMS überein.
- Die *statischen Komponenten* des Modells bestehen aus den Bedienstationen.
- Die einzigen *dynamischen Komponenten* des Modells sind Ereignisse. Diese werden während des Simulationslaufes von den Bedienstationen ständig neu erzeugt und gelöscht.
- Die Ablaufkontrolle ist eine relativ einfache Routine, die relativ unabhängig von dem Modell realisiert werden kann.
- Die Simulation verläuft meist schneller als bei den anderen Ansätzen.

Eine ereignisorientierte Simulation umfaßt die **Ablaufsteuerung**, eine zeitlich sortierte **Liste** aller zukünftigen Ereignisse, das **Modell** und die **Simulationsuhr**, welche die aktuelle Zeit in der Simulation angibt. Das Modell besteht aus Blöcken, an denen die Ereignisse auftreten. Diese Blöcke stellen Bedienstationen dar, die mehrere Ereignisroutinen umfassen, mit denen sie auf das Auftreten von Ereignissen reagieren, z.B. mit der Erzeugung von weiteren Ereignissen. Die Ablaufsteuerung ist sehr einfach realisierbar, da sie lediglich folgende Aktionen ausführt:

```
Modell->initialisiere();
SimUhr->setze(0);

while (SimUhr.lies() < EndeZeitpunkt) {

        Ereignis e = Ereignisliste->hole_naechstes_Ereignis();
        SimUhr.setze(e->zeitpunkt);
        Block b = e->gib_block();
        b->reagiere_auf_ereignis(e);

}
```

10.2 Das Simulationsgrundsystem

Um eine zeitdiskrete ereignisorientierte Simulation zu ermöglichen, ist ein Grundsystem nötig. Dieses stellt die grundlegenden Funktionen zur Simulation auf der Ereignisebene bereit. Das System soll hieraus eine einfache Ereignisleitung ermöglichen. Diese wird durch die Erzeugung von Aktivitäten erreicht. Es handelt sich dabei um Entitäten, welche eine Folge von Ankunfts- und Abgangsereignissen zusammenfassen und zum Beispiel Daten enthalten können, die von einem Block zu einem anderen transportiert werden können. Das Zeitverhalten einer solchen Ereignisfolge wird von den zu durchlaufendenden Blöcken bestimmt. Die Blöcke sind in einer Liste angeordnet und haben folgende Eigenschaften :

- Jeder Block hat eine Routine für ankommende Ereignisse.
- Jeder Block hat eine Routine für abgehende Ereignisse.

- Das Grundverhalten für abgehende Ereignisse besteht in der Erzeugung eines Ankunftsereignisses am folgenden Block.
- Das Grundverhalten für ankommende Ereignisse besteht in der Erzeugung eines Abgangsereignisses am selben Block.
- Ein Block kann die Weiterleitung eines Ereignisses, also die Erzeugung eines Abgangsereignisses als Reaktion auf das Auftreten eines Ankunftsereignisses, zeitlich verzögern. Das wird erreicht, indem sich der Zeitpunkt des Abgangsereignisses aus der aktuellen Zeit (dem Zeitpunkt des Ankunftsereignisses) und einem Verzögerungsintervall, das von weiteren speziellen Charakteristika des Ankunftsereignisses abhängt, ergibt.
- Die Weiterleitung von abgehenden Ereignissen wird im allgemeinen nicht verzögert. Somit treten Ereignisverzögerungen nur innerhalb von Blöcken auf.
- In einem Block existieren so 0-n Abgangsereignisse, die auf ihre Aktivierung warten.
- Jeder Block hat eine aktuelle **Belegung**, d.h. in ihm existieren 0-n auf ihre Aktivierung wartende (Abgangs-)Ereignisse.
- Jeder Block ist durch eine **Kapazität** gekennzeichnet. Diese gibt an, wieviele Ereignisse maximal in ihm warten können.
- Ist die Belegung eines Blocks so groß wie seine Kapazität, so ist dieser Block belegt.
- Ist ein Block belegt, werden an ihm keine Ankunftsereignisse erzeugt, und die betroffenden Abgangsereignisse warten im vorhergehenden Block.
- Sinkt die Belegung eines Blockes unter seine Kapazität, so werden die im vorangehenden Block wartenden Ereignisse reaktiviert und es wird versucht, sie weiterzuleiten.

Diese Grundeigenschaften von Blöcken werden an speziellen Blockarten (siehe unten) weitergegeben. In Anlehnung an das transaktionsorientierte Simulationssystem GPSS/H werden folgende Blockarten zur Verfügung gestellt :

Generator Der Generator erzeugt in bestimmten Abständen Ereignisse und (in seinen Erweiterungen) die damit verbundenen Aktivitäten. Die Ereignisse sind durch einen **Typ**, einen **Block** und einen **Zeitpunkt** gekennzeichnet. Der Zeitpunkt eines Ereignisses ist der Zeitpunkt seiner Aktivierung im System. Dabei wird durch den Block des Ereignisses festgelegt, an welchem Block im Modell das Ereignis auftreten wird, und durch den Typ, ob es sich um eine Ankunfts- oder Abgangsereignis handelt. Ein Ankunftsereignis stellt das Erreichen einer Aktivität an einen Block dar, z.B. wird ein Job in einem Rechner gestartet. Dabei wird die Aktivität verändert. Als Reaktion auf ein Ankunftsereignis kann ein Abgangsereignis zu dieser Aktivität erzeugt werden. So kennzeichnet z.B. das Verlassen einer Waschstraße von einem Auto das Ende des Waschvorgangs.

Bedieneinheit/Facility Die Bedieneinheit ist einer der zentralen Punkte im gesamten Simulationssystem. Wenn ein Ankunftsereignis auftritt, wird die entsprechende Aktivität beeinflußt. So wird durch die Abarbeitung eines Jobs in einem Rechner dieser Job zeitlich verzögert. Eine Bedieneinhait kann meist nur eine begrenzte Anzahl von Aktivitäten verändern, deshalb ist sie nicht immer für ankommende Ereignisse verfügbar.

Warteschlange/Queue Die Warteschlange kann das Auftreten von Ereignissen bzw. deren Aktivitäten verzögern, indem sie das Warten dieser Ereignissen auf ihre Aktivierung ermöglicht. Das ist wichtig, wenn z.B. eine folgende Bedienstation noch von vorangegangenen Ereignissen bzw. deren Aktivitäten belegt ist.

Terminator Am Terminator werden die Aktivitäten der ankommenden Ereignissen gelöscht. Das bedeutet, daß auf ihre Ankunftsereignisse meist keine weiteren Ereignisse folgen. Zusätzliche Möglichkeiten, wie die Weiterleitung von Ereignissen an mehrere Blöcke oder das Auslösen spezieller Aktionen bei der Terminierung von Ereignissen bzw. deren Aktivitäten, können durch Erweiterung dieser Blöcke hinzugefügt werden. Diese Blöcke bilden die Grundlage für eine einfache Netzwerksimulation.

10.3 Die Netzwerksimulation

Ein einfaches Netzwerk besteht aus Rechnerknoten und Verbindungseinrichtungen. Die Rechnerknoten können untereinander in Verbindung treten, indem sie Nachrichten austauschen, die mit Hilfe der Verbindungseinrichtungen transportiert werden. Rechnerknoten können Nachrichten generieren, bearbeiten und weiterleiten. Deshalb realisiert ein Rechnerknoten eine Ereignisleitung, die auf einem Generator, einer Warteschlange, einer Bedieneinheit und einem Terminator sowie einem umfassenden Knotenblock, der für das Routing von Nachrichten zuständig ist, basiert. Dieser Knotenblock stellt gleichzeitig eine Warteschlange für die Nachrichten dar, die von diesem Rechnerknoten aus an andere Knoten gesendet werden sollen, deren Verbindungseinrichtungen aber zum Zeitpunkt des Eintreffens dieses Abgangsereignisses schon belegt sind. Nachrichten werden durch die mit jeweils einer aufeinanderfolgenden Kette von Ereignissen verbundenen Aktivitäten realisert. Sie sind durch einen Quellknoten, einen Zielknoten, ein Datum und die Länge dieses Datums gekennzeichnet. Die Verbindungseinrichtungen können die Nachrichten nur zwischen 2 Rechnerknoten weiterleiten und realisieren so ein Netzwerk mit Punkt-zu-Punkt-Topologie. Zu diesem Zweck bestehen sie aus zwei Bedienstationen, die die Nachrichten entsprechend ihrer Länge und ihrer Richtung (von A nach B oder von B nach A) verzögern, und einem Terminator, der die ankommenden verzögerten Nachrichten zum nächsten Rechnerknoten weiterleitet. Die Bedienstationen zur Nachrichtenübertragung sind durch jeweils eine Übertragungsgeschwindigkeit gekennzeichnet, die festlegt, in welchem zeitlichen Umfang die zu übertragenden Nachrichten verzögert werden sollen. Somit können schnelle Netzwerkverbindungen (z.B. 10 Mbit/s) und langsame (z.B. 9600 bit/s) simuliert werden. Jede Bedienstation ist für eine Richtung zuständig, somit kann eine Netzwerkverbindung jeweils eine eigene Übertragungsgeschwindigkeit für eine Richtung realisieren.

10.4 Die Rechnerknoten mit DBMS

Ein (V)DBMS-Rechnerknoten besteht aus einem speziellen Generator, der Anfragen und Updates auf Attribute der Datenbank stellt, einer speziellen Bedienstation, die diese Anfragen und Updates in Transaktionen auf dem verteilten Datenbankbestand umformt, und einem speziellen Terminator, der die Ergebnisse dieser Transaktionen ausgibt. Bei den Anfragen und Updates handelt es sich um normale Nachrichten, wie sie schon in der Netzwerkebene eingeführt wurden. Das gilt auch für die Ergebnisse der von der Datenbankbedienstationen erzeugten Transaktionen. Die Transaktionen bestehen der Einfachheit halber nur aus jeweils einer Schreib- oder Leseaktion. Um die Konsistenz auf der verteilten Datenbank zu erhalten, werden vor einem

Schreib- oder Lesezugriff je nach Replikationsverfahren verteilte Sperren gesetzt, die nach der Abwicklung des Zugriffes wieder gelöst werden. Dazu arbeitet eine Datenbankbedienstation mit dem Verteilungsschema (Fragmentation und Allokation) und dem Replikationsverfahren zusammen, d.h. die Bedienstation gibt an, auf welchem Attribut welcher Wert geschrieben oder gelesen werden soll, das Verteilungsschema liefert die Fragmente und die Rechnerknoten, auf denen das Attribut allokiert ist, und das Replikationsverfahren gibt die Menge aller Knoten zurück, auf denen Sperren gesetzt bzw. gelöst und Daten gelesen bzw. geschrieben werden sollen. Das Ergebnis dieser Anfrage an die Verteilungsverwaltung ergibt eine Lese- oder Schreibtransaktion, die von der Datenbankbedienstation ausgeführt wird. Eine Anfrage wird folgende Zustände durchlaufen:

- Zuerst wird eine Anfrage an ein Attribut gestellt. Diese erhält den Nachrichten-Typ **Query**.
- Die Anfrage gelangt zur rechnerknoteneigenen VDBMS-Bedienstation. Dort wird aus der Anfrage eine Transaktion, indem mit Hilfe des Verteilungsschemas die Zielknoten der Transaktion bestimmt werden.
- Die Transaktion startet. Der erste Schritt besteht in der Anforderung von Lesesperren auf dem Attribut auf den Zielknoten.
- Wenn die Lesesperren auf dem Attribut nicht erfolgreich gesetzt werden konnten, bricht die Transaktion ab (ABORT) und es werden alle Lesesperren auf den Zielknoten entfernt. Ansonsten wird im zweiten Schritt die Leseaktion auf dem Zielknoten ausgeführt.
- Wenn die Leseaktion erfolgreich beendet werden konnte, wird das Ergebnis zwischengespeichert und es werden im dritten Schritt alle Lesesperren auf dem Attribut auf den Zielknoten entfernt. Ansonsten bricht die Transaktion ab (ABORT) und versucht, alle Lesesperren wieder zu löschen.
- Wurden alle Sperren erfolgreich gelöscht, wird die Transaktion beendet (COMMIT) und es wird eine Nachricht mit dem Ergebnis und dem gelesenen Attributwert generiert, die an den Terminator geschickt wird.
- Der Terminator erhält die Nachricht vom Ende der Transaktion und gibt das Ergebnis der Anfrage aus sowie den Status der Transaktion (COMMIT / ABORT).

Eine Schreibtransaktion läuft analog ab, mit dem Unterschied, daß Schreibsperren verwendet werden. Für die Verträglichkeit der beiden Sperrarten und der dazugehörenden Schreib-Lese-Aktionen gilt, daß nur bei einer von derselben Transaktion gesetzten Schreibsperre geschrieben und ebenso nur bei einer von derselben Transaktion gesetzten Lesesperre gelesen werden darf.

Die vorgestellten Schichten der Simulations-, Netwerk- und Datenbankebene bilden das gesamte Simulationssystem, wie in Abb. 10.1 illustriert. Diese Schichten stellen eine Grundfunktionalität bereit, die für die Implementation von einigen Replikationsverfahren notwendig ist. Die so implementierten Verfahren sollen im folgenden beschrieben werden.

10.5 Die implementierten Verfahren

Nachdem in den vorangegangenen Abschnitten die Schichten der Simulations-, der Netzwerk- und der Datenbankebene beschrieben wurden, sollen die Replikationsverfahren vorgestellt werden, die im Simulationssystem implementiert wurden. Es

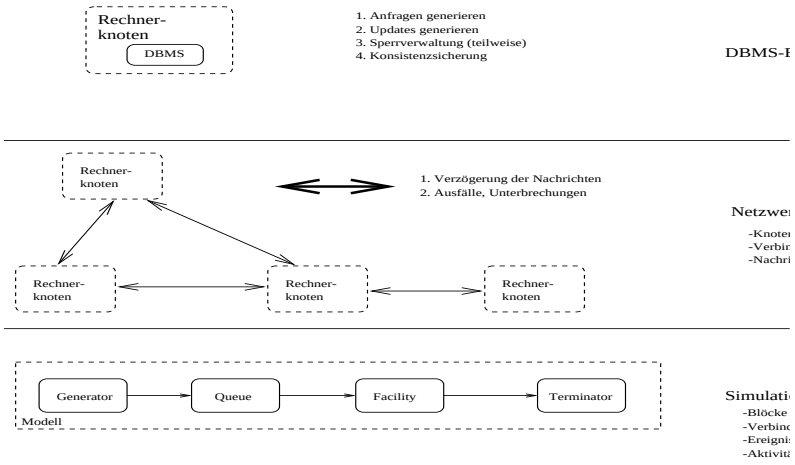

Abbildung 10.1: Aufbau des Simulationssystems

handelt sich dabei um die klassischen Verfahren zur pessimistischen Replikation : **ROWA**, **Write-all-available**, und **Primary site**.

10.5.1 Das implementierte ROWA - Verfahren

Das ROWA-Verfahren wird als pessimistisches Verfahren mit entsprechenden Sperranforderungen ausgeführt. Zum Lesen wird lokal oder am nächsten Replikat eine Lesesperre gesetzt, dort der Inhalt des Attributs aus der Datenbank gelesen und die Sperre wieder entfernt. Bei einer Schreibtransaktion werden zunächst Schreibsperren auf allen Knoten gesetzt, die das Fragment des betroffenen Attributes replizieren. Die Sperranforderung erfolgt seriell; die Knoten werden der Reihe nach ihren Addressen sortiert aufgefordert, die Schreibsperren zu setzen. Dadurch wird zwar eine Großteil der möglichen Parallelität verhindert, zugleich werden Deadlocks vermieden, die sich ansonsten nicht ohne großen Aufwand wieder auflösen liessen. Dann erfolgt ein verteiltes Schreiben, das zur Zeit noch nicht parallel ausgeführt wird. Ebenfalls wird das Löschen der gesetzten Schreibsperren noch seriell ausgeführt. Prinzipiell ist es möglich und empfehlenswert, diese Teile der Schreibtransaktion parallel auszuführen, da sich dadurch eine bessere Ausnutzung der langsameren Netzverbindungen und der Parallelität im VDBMS ergibt.

10.5.2 Das implementierte Write-all-available-Verfahren

Die Implementation ist mit der Beschreibung des Verfahren prinzipiell identisch. Analog zum ROWA-Verfahren laufen die Lese- und Schreibtransaktionen ab. Neu hinzugekommen sind spezielle Rechner im Festnetz, über die sich Mobilrechner mit den anderen Rechnerknoten verbinden können. Diese Rechner (in Anlehnung an das mobile Szenarium *home base node*s (HBN) genannt) fangen die an die vom Festnetz getrennten Mobilrechner zu übertragenden Anforderungen, Sperren zu setzen, Schreibaktionen auszuführen und die Sperren wieder zu lösen, ab. Diese Anforderungen werden vom HBN stellvertretend für die im Moment nicht erreichbaren Mobilrechner beantwortet, ohne sie auszuführen. Dadurch erhält die entsprechende Teiltransaktion ein Commit und kann weiterarbeiten. Gleichzeitig werden von den Anforderungen Kopien angelegt, die in ein Log eingetragen werden, welches vom HBN geführt wird. Sobald ein Mobilrechner sich mit dem Festnetz verbindet, werden die Kopien der bereits bestätigten Anforderungen an ihn weitergeleitet. Konnten alle Anforderungen erfolgreich auf dem Mobilrechner ausgeführt werden, wird

das Log gelöscht. Diese Vorgehensweise setzt voraus, daß der Mobilrechner die Änderungen am Datenbestand in seinen Replikaten stets übernehmen muß, da die entsprechenden Transaktionen bereits erfolgreich abgeschlossen wurden. Anderenfalls würde ein Rollback notwendig, das im Hinblick auf die Transaktionseigenschaften (Atomicity, Consistency, Isolation, Durability) vermieden werden soll. Liegen entsprechende Änderungen in den lokalen Replikaten des Mobilrechners vor, so wäre es unter Umständen möglich, diese getrennt zu sichern, dann den Reintegrationsprozeß auszulösen und schließlich eine Ausgleichstransaktion auszuführen.

10.5.3 Das implementierte Primary-site-Verfahren

Das Lesen bei dieser Art des Primary-Verfahrens erfolgt laut einem Vorschlag aus [AS97] folgendermaßen : Der lesende Knoten setzt eine Lesesperre auf dem Primary, liest lokal den Wert des Attributes aus der Datenbank und löscht die Sperre auf dem Primary. Diese Lesetransaktion ist durch eine einfachere Schreibtransaktion bedingt, die sich in einen Client- und einen Serveranteil (auf dem Primary) aufteilt. Der schreibende Rechnerknoten, kurz als Client bezeichnet, erhalt vom Verteilungsschema eine Transaktion, die nur die Schreibanforderung des Wertes auf das Attribut in der Datenbank enthält. Nach dem Abschicken der Anforderung wartet der Client auf die Bestätigung. Sobald der Primary, hier in der Rolle des Servers, eine solche Anforderung erhält, erzeugt er eine spezielle Transaktion, die die Sperre auf dem entsprechenden Attribut lokal setzt. Dann wird die Bestätigung der Schreibanforderung an den Client abschickt, der von nun an davon ausgehen kann, daß der Wert tatsächlich geschrieben wurde. Auf der Serverseite werden dann alle Replikate mit dem neuen Wert aktualisiert und die lokale Sperre gelöscht. Durch die Aufteilung in Client- und Servertransaktion erscheint das Schreiben vom Client aus gesehen sehr schnell zu sein. Diese Täuschung beruht darauf, daß es sich bei der vom Client ausgeführten Transaktion nur um eine einfache Netzwerknachricht handelt, deren Bestätigung infolge des Setzens der lokalen Sperre auf dem Server bzw. Primary auch relativ rasch erfolgt. Ein Teil der Geschwindigkeit beruht aber auch darauf, daß zum Schreiben auf den Replikaten keine Sperren gesetzt und gelöst werden müssen.

Um die Unterschiede zwischen den einzelnen Verfahren hervorzuheben, wurden auf der Basis eines Grundmodelles für das mobile Szenarium mehrere Simulationsläufe durchgeführt, deren Ergebnisse im nächsten Abschnitt dargelegt werden.

10.6 Die Durchführung und Ergebnisse der Simulation

Das Simulationsmodell besteht aus 5 Rechnerknoten. Davon sind 2 Knoten gleichzeitig HBN und 2 weitere sind Mobilrechner. Die Struktur des Netzes ist in Abb. 10.2 dargelegt. Es handelt sich bei dem simulierten VBDMS um eine vollständig replizierte Datenbank, jeder Knoten hat ein Replikat des Fragmentes, das das Attribut beinhaltet, auf dem alle Anfragen und Updates ausgeführt werden. Die Rechnerknoten arbeiten wie folgt : Jeder Knoten stellt zum Zeitpunkt x eine Update-Anforderung, die von der knoteneigenen Datenbankbedienstation in eine Schreibtransaktion umgesetzt wird. Zum selben Zeitpunkt werden dann Query-Anforderungen generiert (ebenfalls eine pro Knoten), die entsprechend in Lesetransaktionen umgesetzt werden. Die Verteilung der Startzeitpunkte für die Anfragen und Änderungen sind aus Tab. 10.1 entnehmbar. Sowohl die Anfrage- als auch die Änderungsoperationen beziehen sich der Einfachheit halber auf dasselbe Attribut.

Die Mobilrechner trennen sich nach 400 Zeiteinheiten vom Festnetz und verbinden sich nach weiteren 400 Zeiteinheiten wieder mit dem Festnetz, woraufhin

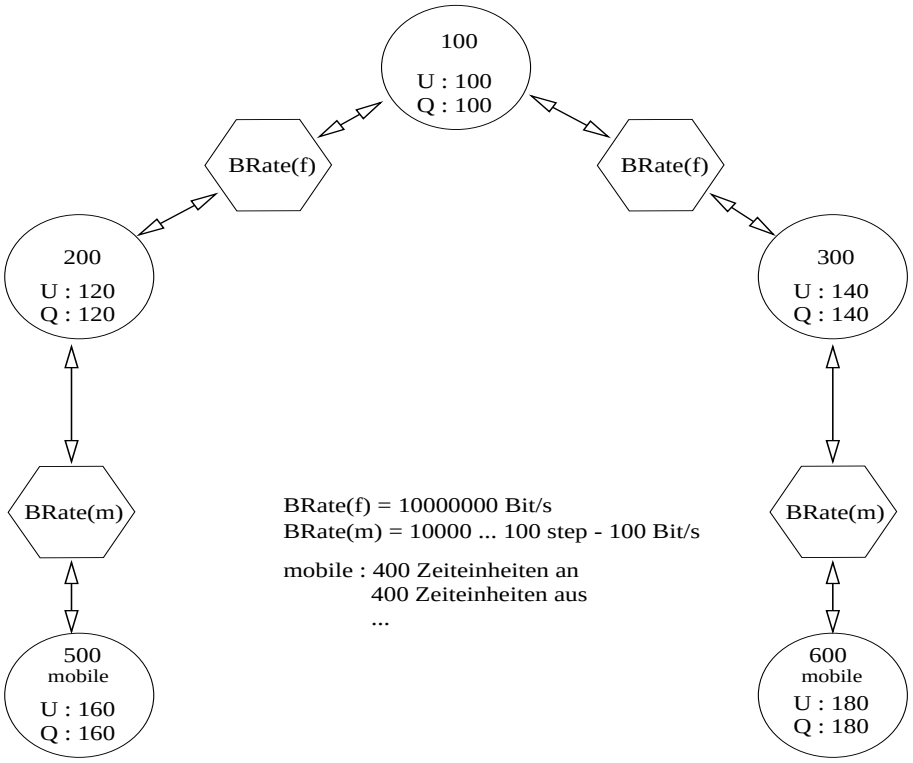

Abbildung 10.2: Aufbau des Simulationssystems

Rechnerknoten-ID	Gateway/-HBN	Mobil-rechner	Änderungs-zeitpunkt	Anfrage-zeitpunkt
100			100	100
200	X		120	120
300	X		140	140
500		X	160	160
600		X	180	180

Tabelle 10.1: Zeitliche Reihenfolge der Anfragen und Updates

die Reintegration beginnt. Die Simulationsläufe unterscheiden sich nur in der vorgegebenen Geschwindigkeiten der Funkverbindungen, die von 10.000 Bit/s auf 100 Bit/s um jeweils 100 Bit/s verringert wird. Die Geschwindigkeit der Festnetzverbindungen beträgt konstant 1.000.000 Bit/s. Die Länge einer einfachen Nachricht, wie das Setzen und Lösen von Sperren und die Bestätigung durchgeführter Operationen (bis auf Lesebestätigungen, die das Ergebnis mit sich tragen), sowie einer Leseanforderung wird mit 1000 Byte angenommen. Eine Lesebestätigung mit Ergebnis umfaßt 20.000 Bytes, eine Schreibanforderung 10.000 Bytes. Die Transaktionen werden ohne Laufzeitbeschränkung (Timeout) benutzt. Die Statistiken teilen sich in die Laufzeiten der Lese- und der Schreibtransaktionen auf. In beiden werden die ungefähren Laufzeiten der von den Rechnerknoten gestarteten Transaktionen angegeben. Die Reintegrationszeit z.B. beim Write-all-available-Verfahren wird dabei nicht berücksichtigt. In den Tabellen wird als Maß für die Nutzbarkeit der Replikationsverfahren der erste Anstieg der Transaktionsdauer angegeben, der (soweit erkennbar) um den Betrag der Zeit steigt, die die Mobilrechner vom Festnetz getrennt sind. Hierbei werden die vorbestimmten 400 Zeiteinheiten angenommen. Damit soll

der Vergleich zwischen den Replikationsverfahren von der eigentlichen Trennungszeit der Mobilrechner vom Festnetz unabhängig gestaltet werden. Wo dieser Anstieg nicht zu beobachten ist, wird angenommen, daß es zu keinen Verzögerungen durch die Trennung der Mobilrechner vom Festnetz kommt.

10.6.1 Die Ergebnisse mit dem ROWA-Verfahren

Leseanforderungen beim ROWA-Verfahren verläufen entsprechend den Erwartungen normal, solange die Funkverbindungen schnell genug sind, alle Nachrichten zu transportieren, bevor es zum Abschalten der Mobilrechner und der Funkverbindungen kommt. Danach steigt die Transaktionsdauer stark an. Der Zeitpunkt für diesen Anstieg ist für die einzelnen Transaktionen unterschiedlich und hängt von dem Startzeitpunkt der Transaktion, den bereits gestarteten Transaktionen und u. U. dem nächsten zu erwartenden Trennungsereignis der Mobilrechner vom Festnetz ab.

Wie in Tab. 10.2 zu erkennen ist, erfolgt das Schreiben von Festnetz aus bis zu einer Geschwindigkeit von etwa 8400 Bit/s relativ problemlos. Ab 300 Bit/s kann man annehmen, daß die Arbeit mit dem VDBMS nicht mehr zu bewältigen ist. Das Lesen erfolgt beim ROWA-Verfahren am nächsten Knoten. Da jeder Knoten das zu lesende Attribut repliziert, kann ein Lesezugriff stets lokal erfolgen. Die Zeiten für die Lesetransaktionen variieren deshalb weniger stark im Vergleich zu den Schreibtransaktionen. Hinzu kommt, daß aufgrund der Lokalität der Lesezugriffe deutliche Verzögerungen in der Laufzeit erst unterhalb von ca. 1000 Bit/s zu erkennen sind. Allerdings steigt die Dauer jeder Lesetransaktion mit sinkender Bandbreite der Funknetzverbindungen trotz der lokalen Zugriffe an. Ein möglicher Grund dafür liegt in der sehr hohen Verzögerung der Laufzeiten der Netzwerknachrichten, so daß sich aufgrund der immer weiter sinkenden Bandbreite der Funkverbindungen die Netzwerknachrichten an den Verbindungsendpunkten stauen und die Transaktionen sich gegenseitig blockieren. Hinzu kommen die Abhängigkeiten der Transaktionen von den bereits gesetzten Sperren.

Rechnerknoten-ID	Anstieg der Lesedauer (Bit/s)	Anstieg der Schreibdauer (Bit/s)
100	ca. 500	ca. 900
200	ca. 500	ca. 2000
300	ca. 500	ca. 3000
500	ca. 900	ca. 5200
600	ca. 900	ca. 8300

Tabelle 10.2: Anstieg der Verzögerungszeit bei ROWA (gerundet)

10.6.2 Die Ergebnisse mit dem Write-all-available-Verfahren

Da die Implementierung des Write-all-available-Verfahrens bis auf die von den HBNs zu führenden Transaktionslogs identisch ist, ähneln sich besonders die Diagramme der Lesetransaktionen. Da auch beim Write-all-available-Verfahren in diesem Modell lokal gelesen werden kann, steigen die Zeiten für die Lesetransaktionen ähnlich wie beim ROWA-Verfahren bis zu einer Bandbreite von 1000 Bit/s sehr langsam an. Größere Unterschiede ergeben sich bei den Schreibtransaktionen. Wie aus Tab. 10.3 ersichtlich, verlagert sich der starke Anstieg der Dauer der Schreibtransaktionen, die von den Rechnerknoten im Festnetz ausgehen, mehr in den Bereich der niedrigeren Datenraten.

Rechnerknoten-ID	Anstieg der Lesedauer (Bit/s)	Anstieg der Schreibdauer (Bit/s)
100	ca. 300	ca. 200
200	ca. 300	ca. 200
300	ca. 300	ca. 200
500	ca. 900	ca. 5200
600	ca. 900	ca. 8300

Tabelle 10.3: Anstieg der Verzögerungszeit bei Write-all-available (gerundet)

10.6.3 Die Ergebnisse mit dem Primary-site-Verfahren

Da bei diesem Verfahren zum Schreiben zwei Transaktionen nötig sind, werden diese getrennt betrachtet. Tabelle 10.4 zeigt die Laufzeiten der Transaktionen, die vom jeweiligen schreibenden bzw. lesenden Rechnerknoten Knoten ausgehen. Diese verlaufen naturgemäß sehr schnell. Der Vorteil gegenüber den Verfahren **ROWA** und **Write-all-available** ist gut zu erkennen. Bei den Schreibtransaktionen im Festnetz kann kein Anstieg der Transaktionsdauer um die 400 Zeiteinheiten beobachtet werden, die ein Mobilrechner vom Festnetz getrennt ist, wogegen ein solcher Anstieg der Transaktionsdauer bei den Schreibtransaktionen von den Mobilrechnern schon bei relativ hohen Datenraten (3500 bzw. 3200 Bit/s) zu erkennen ist.

Um den gesamten Schreibvorgang bewerten zu können, sollen nun auch die Laufzeiten der vom Primary durchgeführten Schreibtransaktionen betrachtet werden. Wie zu ersehen ist, dauern diese Transaktionen länger als die einfachen Schreibanforderungen an den Primary. Allerdings sind sie immer noch früher beendet als die Schreibtransaktionen bei den anderen Verfahren, da die Sperre nur auf dem Primary gesetzt werden muß. Ein Anstieg der Dauer ist bei ca. 3300 Bit/s zu beobachten.

Die Lesegeschwindigkeit ist gegenüber den anderen Verfahren geringer; zu jedem Zugriff muß auf dem Primary (hier der Knoten 100) eine Sperre gesetzt werden. Ein Anstieg um ca. 400 Zeiteinheiten ist bei den Mobilrechnern bereits bei 4900 bzw. 3500 Bit/s zu beobachten.

Rechnerknoten-ID	Anstieg der Lesedauer (Bit/s)	Anstieg der Schreibdauer (Bit/s)
100	-	-
200	ca. 300	-
300	ca. 300	-
500	ca. 3500	ca. 3200
600	ca. 4900	ca. 3500

Tabelle 10.4: Anstieg der Verzögerungszeit bei Primary site (gerundet)

Der Nachteil dieses Verfahrens besteht im mobilen Bereich insbesondere in der Notwendigkeit einer Verbindung zum Primary bei jedem Schreib- und Lesezugriff. Dadurch wird eine Leseoperation auf einem vom Festnetz getrennten Replikat verhindert, was aber unter bestimmten Voraussetzungen (externes Wissen) umgangen werden könnte.

Die Einzelheiten zur Implementation des Simulationssystems sind im Anhang dargelegt. Zum Abschluß der Studienarbeit folgt noch ein Ausblick auf weitere Verbesserungen.

10.7 Ausblick

Folgende Verbesserungen sind infolge der Begrenzung der Arbeit noch zu leisten bzw. sind zur weiteren Untersuchung des Einsatzes von Replikationsverfahren in mobilen Umgebungen sinnvoll :

- Einbringen von Verbesserungen bezüglich der Einbeziehung externen Wissens
- Implementation der verbesserten Replikationsverfahren im Simulationssystem
- Implementation der verbesserten Replikationsverfahren in einer Testumgebung
- Tests, Messungen, Vergleiche der Ergebnisse der beiden Implementationen

Weiterführende Themen:

- Bei Anfrageoptimierung : intelligentes Caching, Anfragebearbeitung aus bereits gestellten Anfragen
- Modulares, skalierbares VDBMS für den Einsatz auf unterschiedlichsten Plattformen
- mobile, verteilte OODBMSs

Anhang A

Walddatenschema im Relationenmodell

create table namen
skey : integer,
nkey : integer,
name : varchar,
constraint primary key(skey, nkey)

create table namensbereiche
s_wo : integer,
s_wie : integer,
s_wodurch : integer,
s_forstbeh : integer,
s_forstdir : integer,
s_forstamt : integer,
s_revier : integer,
s_behandlung : integer,
s_ea : integer,
constraint foreign key(s_wo) **references** namen(skey),
constraint foreign key(s_wie) **references** namen(skey),
constraint foreign key(s_wodurch) **references** namen(skey),
constraint foreign key(s_forstbeh) **references** namen(skey),
constraint foreign key(s_forstdir) **references** namen(skey),
constraint foreign key(s_forstamt) **references** namen(skey),
constraint foreign key(s_revier) **references** namen(skey),
constraint foreign key(s_behandlung) **references** namen(skey),
constraint foreign key(s_ea) **references** namen(skey)

create table uberwachungen
key : integer,
bemerkungen : varchar
constraint primary key(key)

create table wo
ukey : integer,
okey : integer
constraint primary key(ukey, okey),
constraint foreign key(ukey) **references** uberwachung(key),
constraint foreign key(okey) **references** orte(key)

create table orte
key : integer,
ea : integer,
flaeche : integer,
forstbeh : integer,
forstdir : integer,
forstamt : integer,
revier : integer,
abteitung : integer,
unterabt : integer,
teilflaeche : integer,
constraint primary key(key),
constraint foreign key(ea) **references** namen(nkey),
constraint foreign key(forstbeh) **references** namen(nkey),
constraint foreign key(forstdir) **references** namen(nkey),
constraint foreign key(forstamt) **references** namen(nkey),
constraint foreign key(revier) **references** namen(nkey)

create table neben
okey : integer,
mkey : integer,
constraint primary key(okey,mkey)
constraint foreign key(okey) **references** orte(key),
constraint foreign key(mkey) **references** orte(key)

create table kontrolle
pkey : integer,
wodurch : integer,
constraint primary key(pkey,wodurch),
constraint foreign key(pkey) **references** pheromonfallen(key),
constraint foreign key(wodurch) **references** namen(nkey)

create table pheromonfallen
key : integer,
ukey : integer,
nr : integer,
constraint primary key(key),
constraint foreign key(ukey) **references** uberwachungen(key)

create table schaden
key : integer,
ukey : integer,
Uhrzeit : time,
Datum : date,
was : integer,
wie : integer,
wodurch : integer,
constraint primary key(key),
constraint foreign key(ukey) **references** uberwachungen(key)

create table fangbaume
key : integer,
skey : integer,
ukey : integer,
bstuck : integer,

bm3 : integer,
voranflugbeh : boolean,
erstdat : date,
entseuchdat : date,
entsuchart : integer,
constraint primary key(key),
constraint foreign key(skey) **references** schaden(key),
constraint foreign key(ukey) **references** uberwachungen(key),
constraint foreign key(entseuchart) **references** namen(nkey)

create table einzelschaeden
key : integer,
skey : integer,
ukey : integer,
intensitat : integer,
umfang : integer,
redschadfl : integer,
constraint primary key(key),
constraint foreign key(skey) **references** schaden(key),
constraint foreign key(ukey) **references** uberwachungen(key)

create table entwicklung
ekey : integer,
akey : integer,
constraint primary key(ekey,akey),
constraint foreign key(ekey) **references** einzelschaden(key),
constraint foreign key(akey) **references** anderungen(key)

create table anderungen
key : integer,
next : integer,
anzahl_anfang : integer,
anzahl_ende : integer,
grosse : integer,
behandlung : integer,
datum : date,
holz_m3 : integer,
constraint primary key(key),
constraint foreign key(behandlung) **references** namen(nkey),
constraint foreign key(next) **references** anderungen(key)

Anhang B

Fragmentation und Allokation des Wald-Relationenschemas

B.1 Fragmentation

$Orte_n$ = **select** *
from orte
where key **not in**
(**select** okey **from** neben)
union
select o1.orte, o1.ea, o1.flaeche, o1.forstbeh, o1.forstdir, o1.forstamt, o1.revier,
o1.abteilung, o1.unterabt, o1.teilflaeche, o1.flaeche
from o1 orte, o2 orte, neben
where o1.key = okey
and mkey = o2.key
and o2.revier = n

wo_n = wo SJ okey=key $Orte_n$

$pheromonfallen_n$ = pheromonfallen SJ ukey=ukey (wo SJ okey=key $Orte_n$)
$uberwachungen_n$ = uberwachungen SJ key=ukey (wo SJ okey=key $Orte_n$)
$schaden_n$ = schaden SJ ukey=ukey (wo SJ okey=key $Orte_n$)
$einzelschaden_n$ = einzelschaden SJ ukey=ukey (wo SJ okey=key $Orte_n$)
$fangbaeume_n$ = fangbaeume SJ ukey=ukey (wo SJ okey=key $Orte_n$)
$entwicklung_n$ = entwicklung SJ ekey=key (einzelschaden SJ ukey=ukey (wo SJ okey=key $Orte_n$))
$anderungen_n$ = anderungen SJ key=akey (entwicklung SJ ekey=key (einzelschaden SJ ukey=ukey (wo SJ oke
$kontrolle_n$ = kontrolle SJ pkey=key (pheromonfallen SJ ukey=ukey (wo SJ okey=key $Orte_n$))

B.2 Allokation

$fragmente_n$ = { $pheromonfallen_n, uberwachungen_n, schaden_n, einzelschaden_n,$
$fangbaeume_n, entwicklung_n, anderungen_n, kontrolle_n$ }
konstanten = { namen, namensbereiche }
$Node_n$ $\longrightarrow fragmente_n \cup konstanten$

Anhang C

Die Implementierung

Um die Entwicklung und die Tests des Simulationssystems zu vereinfachen, wurde als Sprache **C++** gewählt. Die Entwicklungszeit konnte insbesondere durch die einfache Ablaufsteuerung der ereignisorientierten zeitdiskreten Simulation gering gehalten werden. Eine schon existierende Sprache wie GPSS/H, Simplex etc. wurde nicht in Betracht gezogen, weil die Realisierung von Replikationsalgorithmen mit diesen Mitteln sehr umständlich zu sein scheint.

C.1 Das Simulationsgrundsystem

Das Grundsystem besteht aus 14 Klassen (Abb. C.1). Die Klasse **_event** stellt ein Ereignis dar, welches mit einer Aktivität, realisiert durch die Klasse **_activity**, verbunden sein kann. Die Klasse **_activity** ist in der Lage, eine hierarchische Struktur aus anderen Aktivitäten zu bilden, was in Hinblick auf die spätere Erweiterbarkeit zu bestätigten, verteilten seriellen Aktivitäten (Transaktionen) schon in dieser Ebene notwendig ist. Dazu werden von der Klasse **_activity** Unterklassen abgeleitet, welche bereits die Spezialisierungen in allgemein serielle Aktivitäten (**_serial_activity**), allgemeine parallele Aktivitäten (**_concurrent_activity**) und paarweise auftretende Aktivitäten (**_req_res_activity**) darstellen. Die Ereignisse werden von der Klasse **_event_list** verwaltet. Durch die Erweiterung um die Ablaufsteuerung entsteht aus **_event_list** die Klasse **_simulation**. Die Aktivitäten werden in der Klasse **_activity_list** verwaltet, die analog wie **_event_list** für **_event** als Container fungiert. Das Simulationsmodell wird durch die Klasse **_block** bereitgestellt. Diese Klasse repräsentiert sowohl eine Standard-Bedieneinheit als auch ein Modell, da es selbst mehrere Bedieneinheiten in sich verwalten kann. Das geschieht, indem mit der Klasse **_block** ein einfachen Container für weitere Blöcke bereitgestellt und zwischen diesen Blöcken jeweils Verbindungen einrichtet werden, über die die Ereignisse von Block zu Block weitergeleitet werden. Als Ausgangsbasis für die speziellen Aufgaben von Bedieneinheiten wurden aus **_block** die Klassen **_generator**, **_queue**, **_facility** und **_terminator** abgeleitet. Für die Erzeugung von Statistiken ist die Klasse **_output** bereitgestellt worden. Mit ihrer Hilfe können (später auch im Zusammenhang mit Transaktionen) einfache Diagrammdateien für das GNUplot-System erstellt werden.

C.2 Die Netzwerksimulation

In der Netzwerkschicht (Abb. C.2) wird das Grundsystem um die Fähigkeiten zum gezielten Weiterleiten von Nachrichten erweitert. Leider gelang es in der vorliegenden Implementation nicht, die Trennung zwischen Simulations- und Netzwerkschicht

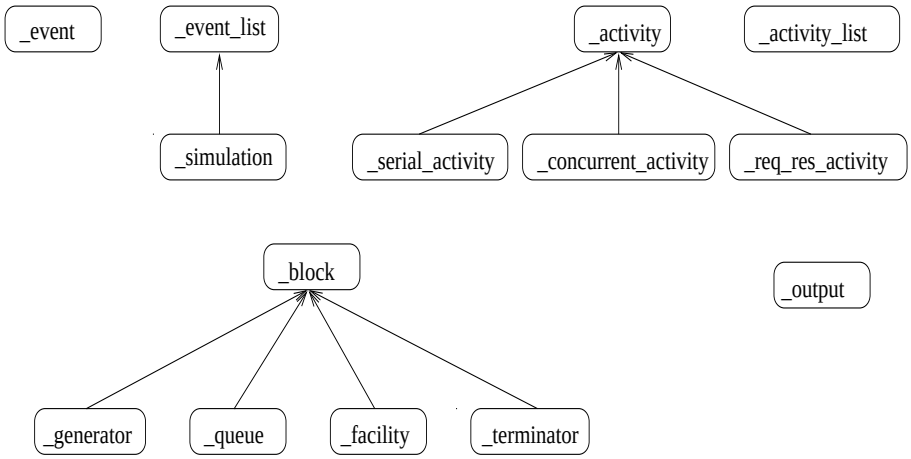

Abbildung C.1: Klassenschema des Simulationsgrundsystems

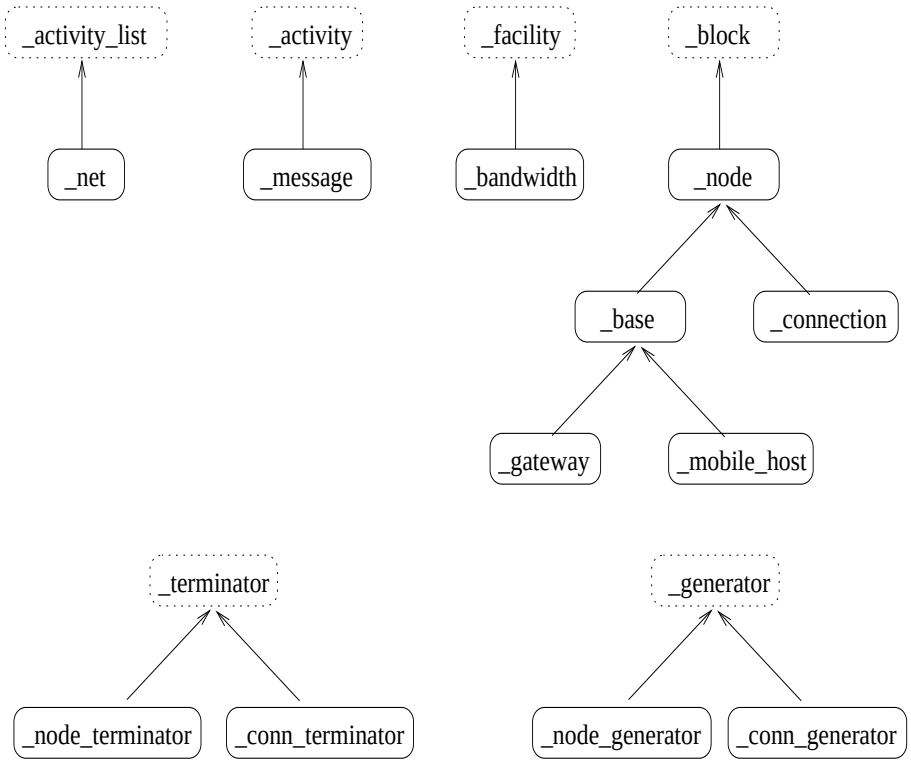

Abbildung C.2: Klassenschema der Netzwerkschicht

vollständig aufrecht zu erhalten. Das liegt in den starken Abhängigkeiten zwischen der Verwaltung von Blöcken durch **_block** und den Erfordernissen von Aktivitäten, realisiert durch **_activity**, begründet. Deshalb umfassen bereits die Aktivitäten in der Simulationsschicht Angaben zum Routing mit der Absenderaddresse. Ebenso haben bereits alle Blöcke in der Simulationsschicht eine Netzwerk- bzw. Blockadresse. Eventuell kann durch eine Nachbearbeitung der Implementierung diese Trennung wiederhergestellt werden. Für die Behandlung und die Weiterleitung von Aktivitäten an anderen Knoten wurde die Basisklasse **_block** erweitert, wodurch die Klasse **_node** entstand. Mit dieser Klasse wird auf der Netzwerkebene der Nachrichtenaustausch zwischen zwei Rechnerknoten, die nur ihre und die Blockaddresse des Zieles kennen, ermöglicht. Fur den Einsatz in jeweils Rechnerknoten und Verbindungseinrichtungen wurden zwei Unterklassen von **_node** abgeleitet, die Klassen **_base** und **_connection**. Der implementierte Routing-Algorithmus ist nicht optimiert, funktioniert aber in den Testphase ausreichend schnell, so daß die Simulationsläufe nicht merklich verzögert werden. Das ist insbesondere auf den geringen Umfang des Modells zurückzuführen.

Die Klasse **_base** stellt einen Rechnerknoten dar, der ankommende Aktivitäten entweder richtig weiterleitet oder, wenn er selbst das Ziel der Aktivität darstellt, in das eigene Untermodell umleitet, welches aus einem Generator, einer Warteschlange, einer Bedieneinheit für die Aktivitäten und einem Terminator besteht. Außerdem werden von diesem Untermodell ausgehende Aktivitäten in einer Warteschlange, die in der Klasse **_base** integriert ist, zwischengespeichert, wenn die für die Weiterleitung nötige Verbindungseinrichtung nicht verfügbar ist.

Die Klasse **_connection** hingegen ist für die Verbindung von jeweils 2 Rechnerknoten zuständig. Sie ist hier als vollduplex modelliert, was in der Realität auf die meisten Verbindungseinrichtungen zutrifft. Sie beinhaltet zwei Simplex-Übertragungseinrichtungen (**_bandwidth** und aus implementationstechnischen Gründen einen speziellen Terminator (**_conn_terminator**). Besonderer Wert wurde auf die Nachbildung der Verzögerung der Datenübertragung in Abhängigkeit von den Übertragungsraten der verwendeten Netzwerkverbindungseinrichtungen gelegt, da diese die Simulation erst sinnvoll machen. Dazu wurde die Klasse **_bandwidth** von der Klasse **_facility** abgeleitet. Diese Klasse findet Verwendung bei der Modellierung einer Verbindungseinrichtung, die eine feste Übertragungsgeschwindigkeit hat. Die oben vorgestellte Klasse **_connection** besteht aus 2 Bedieneinheiten des Types **_bandwidth**, die für jeweils eine Richtung in der Nachrichtenübertragung zuständig sind. Um die Verfügbarkeit einer Verbindungseinrichtung beeinflussen zu können, reagiert die Klasse **_connection** auf Nachrichten eines speziellen Types, die das Ein- und Ausschalten insbesondere der mobilen Funkverbindungen simulieren. Für das zeitgesteuerte An- und Abschalten wurde die Klasse **_conn_generator** als Erweiterung eines Generators implementiert.

Um Netzwerknachrichten erzeugen und richtig leiten zu können, existieren die Klassen **_node_generator** und **_node_terminator**. Die Netzwerknachrichten selbst wurden mit Hilfe der Vererbung von der Klasse **_activity** zur Klasse **_message** realisiert. Die Erweiterungen umfassen Sender- und Empfängeradresse, die Länge der Nachricht und ein Datum. Für die Modellierung eines Netzwerkes ist die Klasse **_net** zuständig. Mit ihrer Hilfe können Rechnerknoten und Verbindungseinrichtungen zu einem Netzwerk zusammengefaßt werden. Für die Bedürfnisse der genaueren Unterschiedung zwischen den Rechnerklassen. die bisher alle in der Klasse **_base** zusammengefaßt sind, wurde zur genaueren Modellierung des mobilen Szenariums die Klasse **_gateway** und die Klasse **_mobile_host** von **_base** abgeleitet. Mit der Klasse **_mobile_host** wird ein Rechner beschrieben, der sich in vorgegebenen Zeitintervallen von Festnetz trennt und wieder verbindet. Die Trennung verläuft in folgenden Schritten :

1. Der mobile Host teilt seinem *home base node*, der durch die Klasse **_gateway** realisiert wird, seinen Trennungswunsch mit.
2. Der HBN vermerkt das und vermittelt keine allgemeinen Nachrichten mehr an den mobilen Host. Diese Nachrichten stauen sich dann im HBN.
3. Der mobile Host empfängt die Bestätigung seines Trennungsgesuches vom HBN.
4. Der mobile Host teilt seiner Verbindungseinrichtung mit, daß sie sich abschalten soll.
5. Der mobile Host empfängt die Bestätigung seines Abschaltungsgesuches von der Verbindungseinrichtung.
6. Die Trennung ist vollzogen.

Die Wiederverbindung verläuft ähnlich:

1. Der mobile Host teilt seiner Verbindungseinrichtung mit, daß sie sich wieder aktivieren soll.
2. Der mobile Host empfängt die Bestätigung seines Anschaltungsgesuches von der Verbindungseinrichtung.
3. Der mobile Host teilt seinem HBN mit, daß er sich wieder mit ihm verbinden möchte.
4. Der mobile Host empfängt die Bestätigung seiner Wiederverbindungsgesuches vom HBN.
5. Der mobile Host ist wieder Teil des Netzwerkes.

Die Klasse **_structure** stellt als Instantiierung von **_net** ein konkretes Netzwerk dar, das vorerst aus 3 Rechnern (ein normaler [_base], zwei HBNs [_gateway]), die über schnelle Festnetzverbindungen, und 2 Rechnern (zwei Mobilrechner [_mobile-_host]), die mit jeweils einem Festnetzrechner (HBN) über eine langsame Netzwerkverbindung verbunden sind, besteht.

C.3 Die Rechnerknoten mit VDBMS

Die Funktionalität der Datenbank-Schicht (C.3) wird durch die Erzeugung von Datenbankanfragen und -updates mittels der erweiterten Generator-Klasse **_db_generator** und der Bearbeitung dieser Anfragen und Updates mit Hilfe der erweiterten Facility-Klasse **_db_facility** bereitgestellt. Die Nachrichtenklasse **_message** aus der Netzwerkschicht wird um weitere Daten zur Beschreibung der Datenbankanfragen und -updates erweitert. Die entstandene Klasse heißt **_action**. Um eine Aktion im Datenbanksinn (Schreiben, Lesen, Sperre setzen, Sperre löschen ...) bestätigen zu lassen und einfacher handhaben zu können, wurden die Klassen **_confirmed_read**, **_confirmed_write**, **_confirmed_lock**, **_confirmed_unlock**, **_confirmed_read_nolock**, **_confirmed_write_nolock** direkt von **_req_res_activity** abgeleitet. Zusätzlich wurde von **_req_res_acitivity** die Klasse **_confirmed_re_action** für die Realisierung der Reintegration und die Klasse **_confirmed_primary_write** als Spezialaktion für die Kommunikation von der Client-Seite zum Server bzw. Primary beim **Primary cite**-Verfahren abgeleitet. Weiterhin wurden die Klassen **_d_c_request** und **_transaction** von der Klasse **_activity** abgeleitet, um mehrere theoretisch parallele bestätigte Aktionen zu einer Aktion zusammenzufassen bzw. um Aktionen nach dem

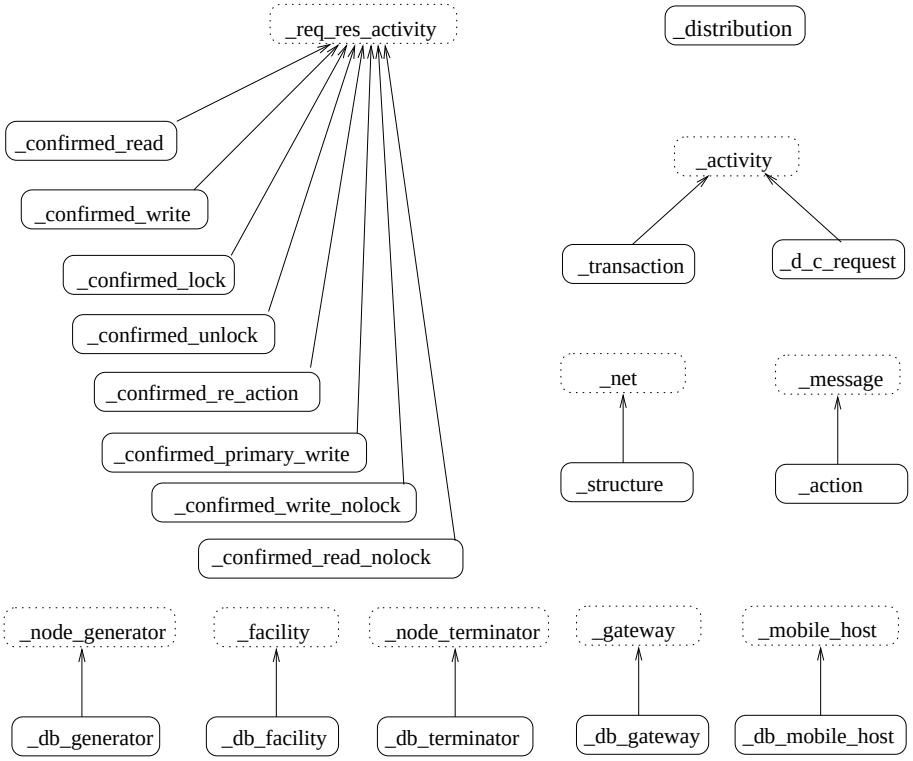

Abbildung C.3: Klassenschema der Datenbankschicht

Schema (Sperren setzen - R/W - Sperren zurücksetzen) ausführen zu können. Der erweiterte Generator erzeugt lediglich Anfragen auf Attributen, die dann von den erweiterten Bedieneinheiten in Lese- und Schreibaktionen auf Fragmente, die sich auf bestimmten Rechnerknoten befinden, umgesetzt werden. Diese umgeformten Aktionen werden dann entsprechend am Zielknoten ausgeführt. Für die Antworten auf eine solche Aktionsanforderung werden dann eigene Transaktionen generiert, die dem Erzeugerknoten der ursprünglichen Transaktion das Ergebnis der Aktion mitteilen. Die Verteilungsinformationen (Fragmentations- und Allokationschema) werden von der Klasse **_distribution** bereitgestellt. Für die Realisierung von einigen Replikationsverfahren wurde es notwendig, den Reintegrationsprozess eines mobilen Rechners so zu erweitern, daß die während dessen Abwesenheit von HBN zu sammelnden Transaktionfragmente als Nachrichten an den mobilen Host weitergeleitet werden können. Dazu wurden die bereits aus der Netzwerkebene vorhandenen Klassen für den HBN (**_gateway**) und den Mobilrechner (**_mobile_host**) um die nötigen Routinen zum Sammeln der Transaktionen auf dem HBN, der vorzeitigen Bestätigung der erfolgreichen Ausführung und dem Nachführen der Transaktionen auf dem Mobilrechner während dessen Reintegrationsphase erweitert. Die entstandenen Klassen sind **_db_mobile_host** für den Mobilrechner mit Datenbank und **_db_gateway** für einen dem Replikationsverfahren angepaßten HBN.

Anhang D

Die Ergebnisse der Simulationsläufe

Die Diagramme zeigen die Dauer einer Transaktion (Ordinate) in Zeiteinheiten in Abhängigkeit von der Datenrate der Netzwerkverbindungen (Abszisse) zwischen den Mobilrechnern 500 bzw. 600 und den *home base nodes* 200 bzw. 300. In der Legende sind die Dateien angegeben, aus denen die Daten für die Transaktionslaufzeiten gewonnen wurden. Der erste Buchstabe gibt an, ob es sich um eine Schreibtransaktion ("**r**") oder eine Lesetransaktion ("**w**") handelt. Die Endung gibt die Knotenaddresse des Rechnerknotens an, der diese Transaktion gestartet hat. Das letzte Diagramm gibt für das implementierte Primary-Verfahren die Laufzeiten der Transaktionen an, die vom Primary (Knotenaddresse 100) als Reaktion auf eine Schreibanforderung erzeugt und gestartet wurden.

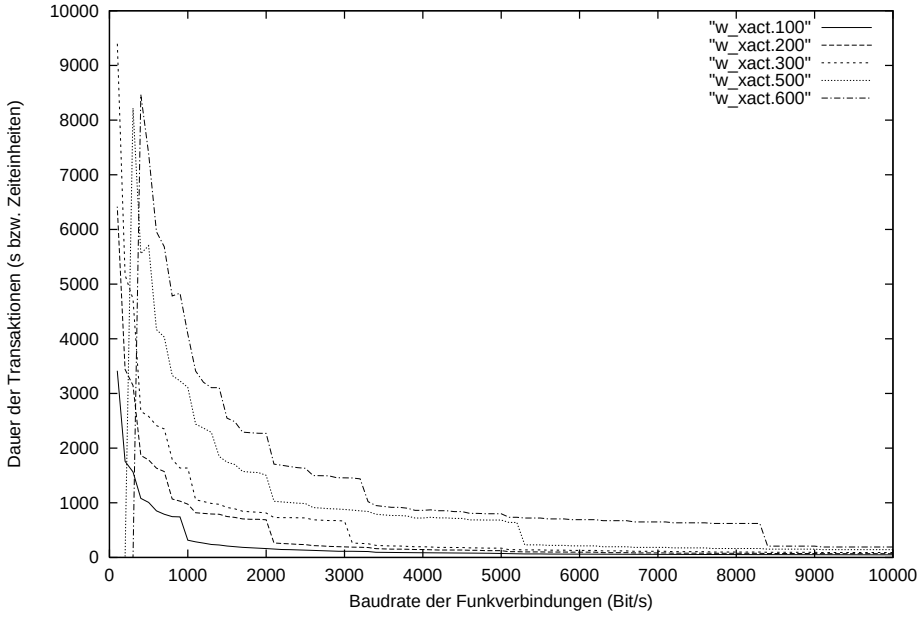

Abbildung D.1: Zeitliche Dauer von Schreibtransaktionen beim ROWA - Verfahren

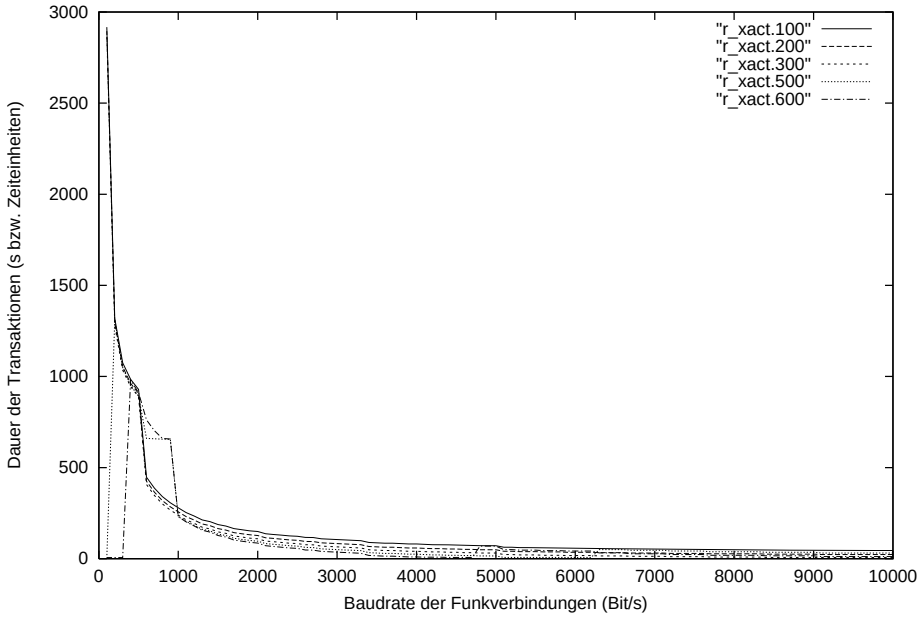

Abbildung D.2: Zeitliche Dauer von Lesetransaktionen beim ROWA - Verfahren

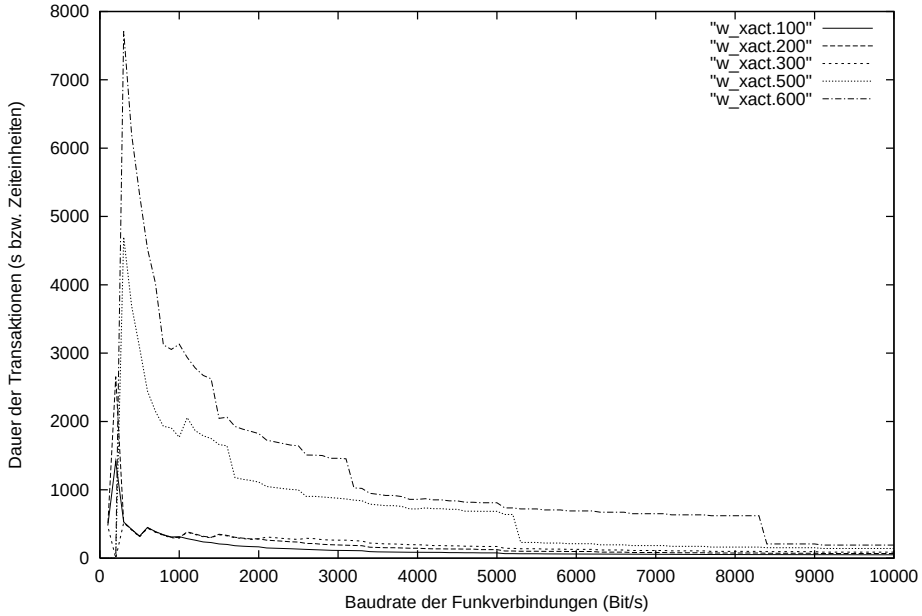

Abbildung D.3: Zeitliche Dauer von Schreibtransaktionen beim WAA - Verfahren

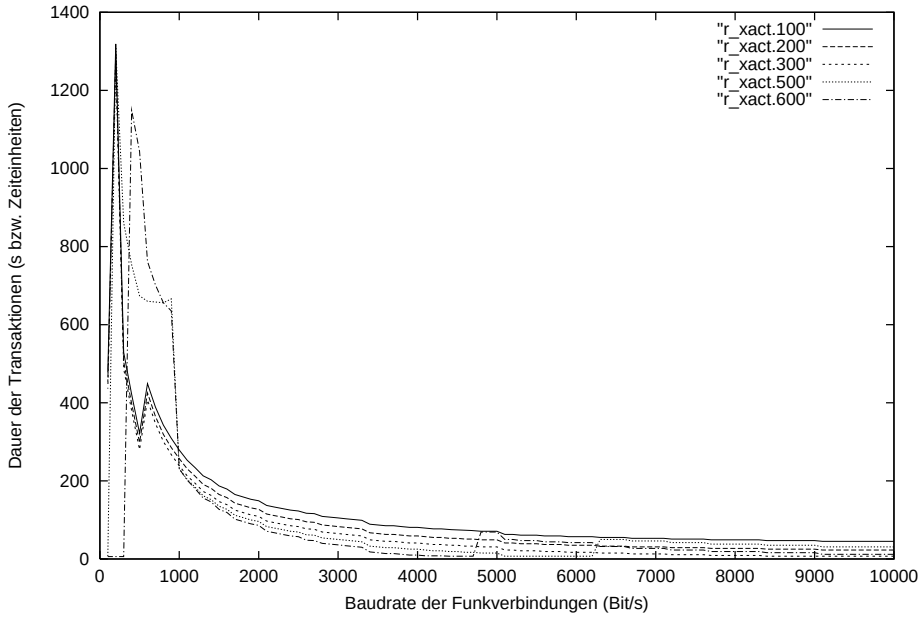

Abbildung D.4: Zeitliche Dauer von Lesetransaktionen beim WAA - Verfahren

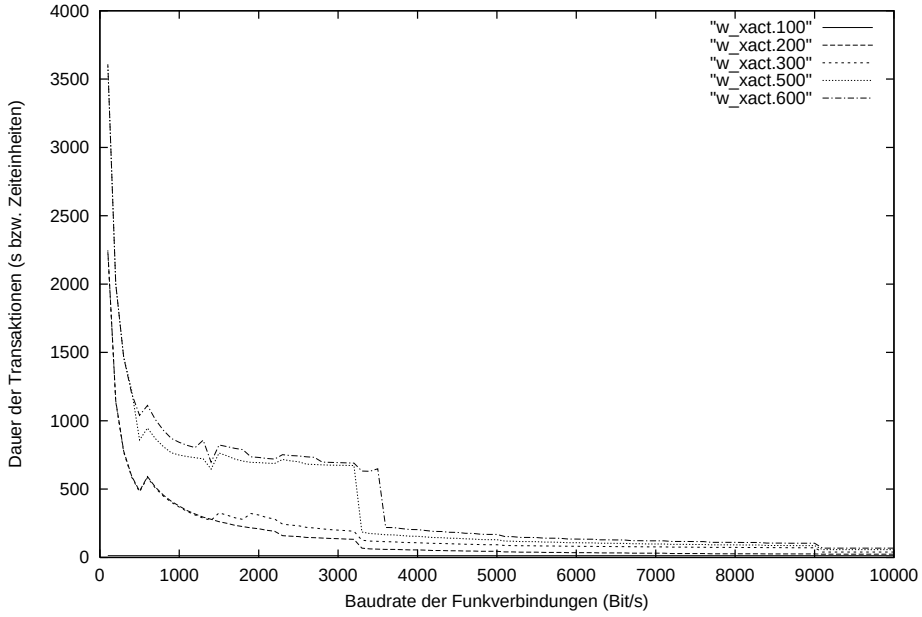

Abbildung D.5: Zeitliche Dauer von Schreibtransaktionen beim Primary - Verfahren

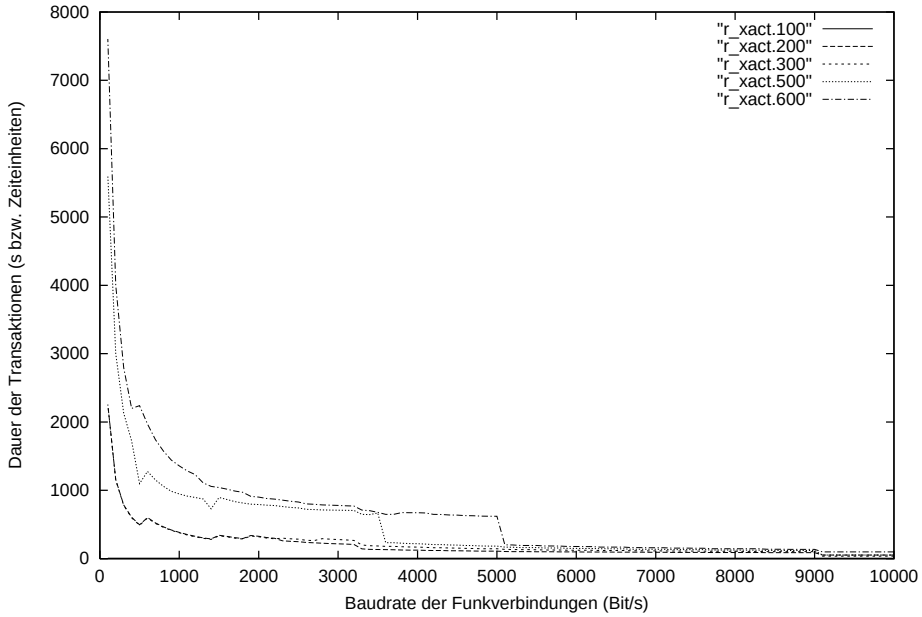

Abbildung D.6: Zeitliche Dauer von Lesetransaktionen beim Primary - Verfahren

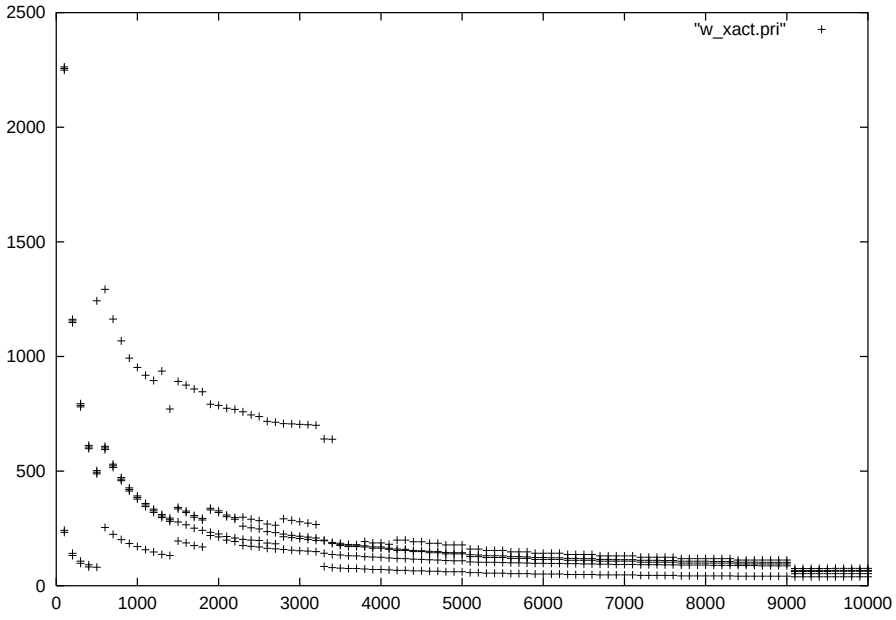

Abbildung D.7: Zeitliche Dauer von Primary-Schreibtransaktionen beim Primary - Verfahren

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AS97] A. Sarodnik. *Evaluierung und Bewertung von Replikationstechniken in verteilten Datenbanksystemen* Diplomarbeit, Universität Rostock, 1997

[BD96] T. Beuter, P. Dadam. *Prinzipien der Replikationskontolle in verteilten Datnebanksystemen* in : Informatik - Forschung und Entwicklung Springer 1996

[BI94] D. Barbara, T. Imielinski. *Sleepers and Workaholics : Caching Strategies in Mobile Computing* in : The VLDB Journal, October 1995, Volume 4, Number 4 Special System-oriented Section : The Best of SIGMOD'94 S. 567-602

[Co97] S. Conrad. *Föderierte Datenbanksysteme - Konzepte der Datenintegration* Springer, 1997

[CP84] S. Ceri, G.Pelagatti. *Distributed Databases - Principles and Systems* McGraw-Hill Book Company, 1984

[FZS95] M. Faiz, A.Zaslavsky, B. Srinivasan. *Revising replication strategies for Mobile Computing environments* in : ECOOP' 95 Workshop on Mobility and Replication Aarhus, Denmark, August 18th

[Hi95] S. G. Hild. *Disconnected Operations for Wireless Nodes - Position Statement* in : ECOOP'95 Workshop on Mobility and Replication Aarhus, Denmark, August 18th

[IB92a] T. Imielinski, B. R. Badrinath. *Querying in highly mobile distributed environments* in : The 18th VLDB-Conference, Vancouver, British Columbia, Canada 1992 S. 1-12

[IB93] T. Imielinski, B. R. Badrinath. *Data management for Mobile Computing* in : SIGMOD record Volume 22, Number 1, March 1993

[IB92b] T. Imielinski, B. R. Badrinath. *Replication and mobility* in : 2nd IEEE Workshop on Management of replicated data '92

[Ki90] J. J. Kistler. *Increasing file system availability through second-class replication*

[MES95] L. B. Mummert, M. R. Ebling, M. Satyanarayanan. *Exploiting weak connectitvity for mobile file access*

[NS96] B. D. Noble, M. Satyanarayanan. *A research status report on adaption for mobile data access* in : SIGMOD Record Volume 24, Number 4, December 1995

[OZ97] O. Zukunft. *Integration mobiler und aktiver Datenbankmechanismen als Basis für die ortsgebundene Vorgangsbearbeitung* in: Dissertation zur Erlangung des Grades des Doktors der Ingenieurwissenschaften am Fachbereich der Carl von Ossietzky Universitt Oldenburg Oldenburg, Dezember 1997 S. 90-122

[ÖV91] M. Tamer Öszu, P. Valduriez. *Principles of Distributed Database Systems* Prentice-Hall, 1991

[Pa91] B. Page. *Diskrete Simulation. Eine Einführung mit Modula-2* Springer, 1991

[PB95] E. Pitoura, B. Bhargava. *Maintaining Consistency of data in mobile distributed environments* in : Proceedings of the 15th International Conference on distributed computing systems Vancouver, British Columbia, Canada, May 30 - June 2, 1995

[Pi96] E. Pitoura. *A replication scheme to support weak connectivity in mobile information systems* in : 7th International Conference on Database and Expert Systems Application (DEXA), September 1996

[Sa96] M. Satyanarayanan. *Fundamental challenges in Mobile Computing* in : Proceedings of the 15. Annual Sympsoium on Principles of Distributed Computing Philadelphia, Pennsylvania, USA, May 23-26, 1996

[SKB+93] M. Satyanarayanan, J. J. Kistler, L. B. Mummert, M. R. Ebling, P. Kumar, Q. Lu *Experience with disconnected operation in a Mobile Computing environment* in : Proceedings of the 1993 USENIX Symposium on Mobile and Location-independent Computing, Cambridge, MA, August 1993

[TP] Terry Pratchett. *Das Licht der Phantasie*

[WJ92] O. Wolfson, S. Jajodia. *Distributed algorithms for dynamic replication of data* in : Proceedings of the 11th ACM SIFACT-SIGMOD-SIGART Symposium on Principles of Database Systems June 2-4, 1992 San Diego, Californien S. 149-163

[WJH97] O. Wolfson, S. Jajodia, Y. Huag. *An Adaptive data replication algorithm* in : ACM Transaction on Database Systems, June 1997, Volume 22, Number 2 S. 255-313

[WP95] Q.R.Wang, J.-F. Paris. *Managing replicated data using referees* in : ECOOP' 95 Workshop on Mobility and Replication Aarhus, Denmark, August 18th S. 1-4