UNIVERSITÄT ROSTOCK

FACHBEREICH INFORMATIK

ADAPTION VON DATENBANKSYSTEMEN AN DEN MOBILEN KONTEXT „ORT UND ZEIT“

Studienarbeit

vorgelegt vom Marcus Mattèrn

geboren am 25.04.1973 in Greifswald

Betreuer: Prof. Dr. Andreas Heuer

Dipl. Ing. Astrid Lubinski

Rostock, den 26.5.1998

Beschreibung der Aufgabe:

Der Mobile Kontext beschreibt die typisch mobilen Eigenschaften einer Verarbeitungsumgebung. Der Anfragebearbeitungsprozeß soll sich automatisch an die jeweils geltenden mobilen Kontexte anpassen. Es gibt verschiedene Auswirkungen auf die Funktionalität des Datenbanksystems, den Anfrageprozeß, das Transaktionsmanagement usw. und die im Datenbanksystem verwalteten Informationen. Eine zentrale Kontextart in mobiler Umgebung stellen Ort und Zeit dar. Von dem Aufenthaltsort und der Zeit des Nutzers hängen die erforderlichen Informationen, Funktionalität und die weiteren Kontexte ab. So läßt sich dieser Kontext nur in Verbindung mit dem Nutzerkontext betrachten, da es um den Aufenthaltsort des Nutzers geht. Für die Anpassung von Informationen und Verarbeitung kann der Ort zu unterschiedlichen Zeiten ebenso wie der Ort in unterschiedlichen Abstraktionsstufen eine Rolle spielen.
Ziel der Studienarbeit ist es, aufbauend auf der Diplomarbeit von J. Waterstraat eine Beziehung innerhalb des Orts-/Zeitkontextes und zwischen ihm und den Wirkungen auf das Datenbanksystem herzustellen. Als Voraussetzung dafür ist zunächst festzustellen, was Ort-, Zeit- und orts- und zeitabhängige Angaben sind und welche Beziehungen zwischen ihnen bestehen (z.B. Hierarchien, Ebenen). Dann sollen die möglichen Wirkungen dieser Kontexte und Kontextbeziehungen auf Datenbanksysteme ermittelt werden. Die Orts-/Zeitangaben und ihre Beziehungen sollen an zwei Anwendungsszenarien überprüft werden und für eines (mobile Walddatenerfassung) Kontextregeln zur Nutzung in einem Kontextserver implementiert werden.

Inhaltsverzeichnis

1 Einleitung

Das rapide zunehmende Informationsbedürfnis der modernen Gesellschaft stellt einen bedeutenden Faktor in der Entwicklung der globalen Vernetzung dar. In der heutigen Zeit ist die Information einer der wichtigsten Bestandteile der täglichen Arbeit. Viele in Firmen neu entstehenden Aufgabenbereiche sind mit einem gewissen Maß an Mobilität verbunden.
Der Begriff der Mobilität wurde in [Wat96] umfangreich erklärt, so daß ich an dieser Stelle darauf verweisen möchte.
Der Einsatz von mobilen Mitarbeitern bringt für viele Firmen eine Effektivitätssteigerung und eine größere Flexibilität in zahlreichen Arbeitsbereichen.
Um diese Ziele zu erreichen, werden die dabei anfallenden Informationsmengen in verteilten Systemen gespeichert. Diese Systeme managen die Bearbeitung und Bereitstellung der Informationen so, daß der Benutzer seine Ergebnisse bekommt, ohne sich mit der Herkunft der gelieferten Daten beschäftigen zu müssen.

Um die vielen in mobilen Umgebungen auftretenden Eigenschaften, (z.B. Ort des Nutzers, Gerät, Vorlieben des Nutzers), vollständig zu erfassen und leicht handhabbar zu machen, wurden Kontexte eingeführt. Die genaue Kontextdefinition kann man wiederum in [Wat96] nachlesen.
Den Mobilen Kontext kann man als Zusammenfassung der mobilen Eigenschaften einer bestimmten Anwendungsumgebung auffassen.

Um Untersuchungen an Kontexten zu unternehmen, ist es notwendig, erst einmal die Grundstruktur von mobilen Umgebungen festzulegen. Der Großteil der sich mit diesem Thema „Mobile Computing“ beschäftigenden Arbeiten hat als Architektur ein Zellulares Netzwerk zu Grunde gelegt z.B. [Imil92/1],[Imil92/2],[RajaBa]. Viele dieser Autoren beschäftigen sich aber hauptsächlich mit Strategien zur Ortsbestimmung von Objekten in solchen Netzen und deren technischer Realisierung.

Ein solches Netzwerk besteht aus mehreren Teilen. Dem Festnetzteil (auch Public Access Network), dem Mobilteil (Funkzellen und darin befindliche Mobile Einheiten) und dem Verbindungsteil (Basisstationen und Ortsserver), welcher die Zuordnung der Mobilen Einheiten zu logischen Orten (Servern) managt.

Der Aufbau eines solchen Netzes sieht folgendermaßen aus:

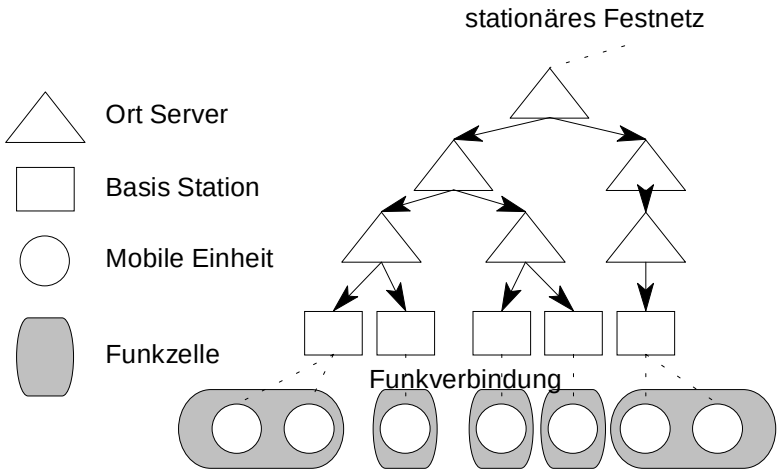

Abbildung 1: Architektur eines zellularen Netzwerkes [Imil92/2]

Der mobile Nutzer ist in diesem Modell also als Mobile Einheit zu verstehen. Er befindet sich mit seinem mobilen Gerät in einer Funkzelle, die von einer Basisstation abgedeckt wird. Die Basisstationen haben immer die genauen Information, welche mobilen Nutzer sich in ihren Zellen aufhalten. Die Ortsserver haben nicht ständig die Information, in welcher Zelle sich ein Nutzer aufhält. Braucht der Server diese, so sendet er eine Art Broadcast über alle in seiner Hierarchie liegenden Basisstationen. Die Daten auf die der Nutzer Zugriff haben möchte, können dabei im Festnetz, im Mobilnetz oder auch verteilt in beiden, gespeichert sein.
Um mit bekannten (Vorlieben, Rechte) und ermittelten Daten (Aufenthaltsort, Systemzeit) eine sinnvolle Anfragebearbeitung zu ermöglichen, werden diese Eigenschaften zu Gruppen zusammengefaßt und als Kontext bezeichnet. Eine Erläuterung und Möglichkeit zur Einteilung von Kontexten wird im anschließenden Abschnitt gegeben.

2 Kontexte

2.1 Eigenschaften von Kontexten

Die wichtigste Eigenschaft von Kontexten, ist ihr Einfluß auf den Anfrage- und Antwortprozeß von Datenbankanwendungen.

Zur Anpassung des Anfrageprozesses an den Kontext bedarf es einiger zusätzlicher Informationen, die für den Prozeß beachtet werden müssen. **[Wat96]**

- **Aktualität:** Um eine korrekte Anpassung einer Anfrage an einen Kontext zu erreichen, muß klar sein, ob die im System vorhandenen Informationen dem aktuellsten Stand entsprechen. Eine Möglichkeit, die Aktualität von Informationen zu gewährleisten, ist die Nutzung von Zeitstempeln.
- **Genauigkeit:** Von der Genauigkeit der Kontextinformation, die zur Anpassung der Anfrage benutzt wird, hängt das Ergebnis unmittelbar ab. Das Ergebnis einer Anfrage, könnte sinnlos sein, wenn durch Ungenauigkeit statt einem Objekt, mehrere Objekte zurück geliefert werden.
- **Herkunft:** Aus der Herkunft von Informationen können andere Daten gewonnen werden. Dazu gehören z.B. Aktualität und Genauigkeit der Information. Zur Ableitung dieser Eigenschaften, benötigt das System Wissen, wie es aus der Herkunft von Daten zum Beispiel auf deren Aktualität schließen kann.

Die vorhandenen Klassifikationsmöglichkeiten für Kontexte sind

- der Zeitpunkt der Einbeziehung,
- die Häufigkeit der Änderung der Kontexte,
- die Beschreibbarkeit,
- die Art und Weise der Beeinflussung.

Die beiden ersten Klassifizierungsmöglichkeiten werden nun kurz erläutert.

2.2 Klassifikation nach Zeitpunkt der Einbeziehung

Ein mögliche Betrachtung von Kontexten ist eine Klassifikation nach dem Zeitpunkt, zu dem das System die Informationen des Kontextes in den Anfrageprozeß einbezieht.

Es gibt hierbei drei Klassen von Kontexten:

Einbeziehung vor Systemnutzung

Damit ein Kontext vor Systemnutzung Einfluß nehmen kann, müssen die Kontextinformationen bekannt sein. Das System trifft aufgrund der Kontextinformationen schon Vorbereitungen bzw. Vorauswahlen von Daten oder Arbeitshilfsmitteln.
Ein Beispiel ist die Einstellung von Sichten, damit der Benutzer nur auf erlaubte bzw. benötigte Daten Zugriff hat.
Eine Änderung des Kontextes während der Systemnutzung wirkt sich auf die Funktionsfähigkeit des Systems aus. Übergibt z.B. ein Mitarbeiter seinem Vorgesetzten das System, so hat dieser im allgemeinen Zugriff auf größere Datenmengen und die vorher erstellten Sichten müssen aktualisiert bzw. neu erstellt werden.

Einbeziehung während der Systemnutzung

Diese Art der Kontextbenutzung ist die flexibelste. Sie wird nach [WAT96] in verschiedene Klassen eingeteilt. Diese Zuordnung geschieht in Abhängigkeit vom Ereignis, das die Kontextbetrachtung auslöst.
Zuerst zu nennen, ist die kontinuierliche Betrachtung des Kontextes, welche zu einer ständigen Belastung des Systems führt, da zu jedem Zeitpunkt die Kontextmerkmale ausgewertet werden.
Eine zweite Methode ist die Kontextbetrachtung zu vordefinierten Zeitpunkten. Dabei kann der Zeitpunktes der Einbeziehung durch den Nutzer oder das System festgelegt werden. Bei dynamischen Kontexten ist die Verwendung dieser Methode nachteilig, weil Änderungen von Kontexten zwischen den vordefinierten Zeitpunkten nicht berücksichtigt werden.

Die beste Möglichkeit der Betrachtung von Kontexten bietet sich durch vordefinierte Aktionen, die als Auslöser für eine Hinzunahme von Kontextdaten dienen.
So wird es möglich, das System so einzustellen, das es auf Systemereignisse und auf Nutzereignisse reagiert. Die aus Datenbanksystemen bekannten ECA Regeln sind ein Hilfsmittel, das hierfür zur Verfügung steht.
Ergänzend zu der eben genannten Möglichkeit bietet sich die Betrachtung des Kontextes nach einer Nutzeranforderung an. Der Nutzer ist somit in der Lage eine sofortige Einbeziehung des Kontextes zu erzwingen.

Einbeziehung zwischen Sitzungen

Die Betrachtung zwischen den Sitzungen bietet die Möglichkeit der Berechnung von Kontextdaten. Während der Nutzung des Systems kann dann auf vorberechneten Daten zugegriffen werden. Probleme entstehen aber bei längerer Nichtnutzung des Systems, weil bereits berechnete Kontexte unter Umständen nicht mehr aktuell sind.

2.3 Klassifikation nach der Häufigkeit der Änderung

Kontexte die sich während ihrer Nutzung durch ein Datenbanksystem nicht ändern, heißen statische Kontexte. Treten dagegen Änderungen im Kontext auf, so spricht man von einem dynamischen Kontext.

Als Beispiel soll hier schon einmal die Mobile Walddatenerfassung dienen, die in Kapitel 3 ausführlich erläutert wird.
Wenn sich ein Waldarbeiter in einem Revier im System anmeldet, so wird seine Applikation gestartet und seine Voreinstellungen werden schon eingestellt. Diese Vorlieben sind im Nutzerkontext gespeichert. Da sie sich nur selten ändern, ist der Nutzerkontext eher ein statischer Kontext.
Im Gegensatz dazu steht der Ortskontext. Durch die Mobilität bedingt, ändert sich der Aufenthaltsort eines Nutzer häufig, somit ändert sich auch der zum Nutzer gehörende Ortskontext. Dieser Kontext ist also dynamisch. Genauso kann auch der Zeitkontext betrachtet werden. Da die Zeitangaben sich in den meisten Anwendungsfällen laufend ändern, zählt auch der Zeitkontext zu den dynamischen Kontexten.

2.4 Klassifikation in mobilen Systemen

Für die Klassifikation von Kontexten in mobilen Umgebungen wird in [Wat96] eine Möglichkeit vorgeschlagen, Kontexte nach ihrer Beeinflussung auf die Arbeitsweise des DBS zu unterscheiden. Die drei Klassifikationsgruppen sind dabei der mobile Mensch, das mobile Benutzergerät (DV-System) und auch die mobile Information. Der Mensch mit all seinen Merkmalen, Vorlieben, Fähigkeiten und Einschränkungen kann als grob als Nutzerkontext zusammengefaßt werden. Alle Eigenschaften des DV-Systems wie z.B. Hardwareangaben, Vorschriften und Restriktionen werden durch den Ressourcenkontext beschrieben. Die Merkmale,

Statistiken und Restriktionen für die Informationen werden im Informationskontext [Boj97] beschrieben.
Nun fehlt noch die Einordnung des Ort-/Zeitkontextes in diese Klassifikation. Eine Einteilung in nur eine Klasse ist dabei nicht möglich, da sich in allen drei Klassen orts- und zeitabhängige Eigenschaften wiederfinden. Beim Menschen ist dies u.a. der Ort, an dem er eine Anfrage an das System stellt, bei der Information der Ort, wo sie zur Verfügung steht und wann bzw. wie lange sie gültig ist.
Der Orts-/Zeitkontext ist also in allen drei Klassen mit mehr oder weniger wichtigen Eigenschaften enthalten.

3 Einführung in die Beispielszenarios

3.1 Die Mobile Walddatenerfassung

Die mobile Walddatenerfassung ist ein Beispiel aus der Forstwirtschaft, an dem in dieser Arbeit Betrachtungen zu Kontexten und deren Einfluß auf Datenbanksysteme angestellt werden.

Grundlage des Beispiels ist die Informationserfassung in der Forstwirtschaft. Sie dient zur Planung der Waldnutzung, deckt Problemzonen auf und gibt Aufschluß, an welchen Stellen Aktionen wie z.B. Aufforstungen oder Pflegemaßnahmen notwendig sind.
Die herkömmliche Datenerfassung geschieht mit einem Fragebogen, den der Forsteinrichter für jede Teilfläche im Wald auszufüllen hat. Ein Mißstand bei diesem Verfahren ist die Verschlüsselung der zu erfassenden Eigenschaften. Zur Hilfe hat jeder Forsteinrichter einen Katalog, der ihm zur Abkürzung auf dem Fragebogen die Eigenschaft genauer beschreibt und die möglichen einzutragenden Werte auflistet.
Zum Beispiel steht im Fragebogen an einer Stelle die Abkürzung HART. Diese steht für die Hiebsart und soll Auskunft darüber geben, wie das Holz gepflegt wird.
Einige mögliche Werte sind 10=Jungwuchspflege, 30=Aushieb, 40=Kahlhieb. In den Fragebögen werden die Schlüsselnummern der entsprechenden Attribute eingetragen.
Wie man sieht, ist diese Methode der Datenerfassung sehr unkomfortabel.

Zur Verbesserung soll deshalb ein System eingesetzt werden, was dem Forsteinrichter erlaubt mit einem mobilen Gerät (z.B. PDA) die Erfassung im Wald vorzunehmen.
Wegen der beschränkten Ressourcen eines PDA’s , ist es notwendig, die Datenmengen auf dem mobilen Gerät so gering wie möglich zu halten. Zur Unterstützung dieser Datenbearbeitung werden Kontexte eingesetzt, die schon auf der Datenbankseite schon eine Verringerung der Datenmenge zulassen.
Eine wichtige Rolle in diesem Szenario spielt der Aufenthaltsort des Forsteinrichters. Das System soll diesen automatisch feststellen können, um ihn dann für weitere Aktionen auswerten zu können.
Weitere Untersuchungen zu Merkmalen des Ortskontextes werden im folgenden Kapitel 4 durchgeführt.

3.2 Die Mobile Möbelvertriebsfirma

Der mobile Möbelvertrieb ist eine flexible Betriebsart, die für den Kunden eine Erleichterung im Einkauf bietet. Diese Erleichterung wird durch die Struktur des Möbelvertriebs erreicht. Zwei Punkte sind als Verbesserung für den Kunden gegenüber dem herkömmlichen Einkauf in einem Möbelhaus herauszuheben. Zum einen das Einkaufen von „zu Hause“ aus und zum zweiten das große Angebot eines überregionalen Möbelvertriebs.
Die Möbelvertriebsfirma setzt sich aus mehreren Filialen zusammen. In diesen Filialen arbeiten die Außendienstmitarbeiter, von denen jeder seinen Kundenstamm betreut.
Weiterhin hat jede Filiale seine Lkw Fahrer, die für die Abholung der Möbel vom Lieferanten und die Auslieferung zum Kunden verantwortlich sind. Auf der obersten Stufe in der Firma befindet sich der Firmenchef, der für das Aussuchen neuer Produkte und die Preisverhandlungen mit dem Lieferanten zuständig ist.
Genauere Erläuterungen zu diesem Beispiel werden in [Boj97] gegeben.
In diesem Beispiel erfolgt im Gegensatz zur Walddatenerfassung eine strikte Rollentrennung der verschiedenen Personen. Die drei Personengruppen Chef, Vertreter und Lkw Fahrer haben unterschiedliche Rechte und Funktionen zur Verfügung.

Welche Rolle Orts- und Zeitkontext für die Arbeit der Personengruppen spielen, wird an späterer Stelle in dieser Arbeit kurz erläutert.

4 Ort-/ Zeit Kontext

Die verschiedenen Merkmale der mobilen Umgebungen lassen sich in Kontexten zusammenfassen. So werden alle nutzerabhängigen Angaben, wie z.B. Vorlieben und Einstellungen als Nutzerkontext behandelt. Alle technischen Daten, wie z.B. Gerätedaten wie Displaygröße, Speicherplatz oder auch Verbindungsdaten wie die Bandbreite der Funkverbindung werden als Informationskontext zusammengefaßt. Die Mobilitätsangaben wie der Ort des Nutzers, der Anfragezeitpunkt bzw. andere Zeitkomponenten und Beziehungen zwischen verschiedenen Orten werden im Orts- / Zeitkontext erfaßt. Um den Orts- /Zeitkontext beschreiben zu können werden jetzt seine Eigenschaften aufgezählt und erläutert.

4.1 Eigenschaften des Ortskontextes

4.1.1 Aufenthaltsort des Nutzers

Der Aufenthaltsort des Nutzers ist der Ort, an dem sich der mobile Nutzer zum Zeitpunkt der Anfragestellung befindet. Diese Ortsangabe wird als Erweiterung der Anfrage benutzt, um die Menge der Ergebnisobjekte ortsabhängig einzuschränken.
Zum Beispiel könnte Herr Schulz, der in Rostock wohnt, fragen „Welche Filme spielen zur Zeit im Kino?“. Stellt er dieselbe Frage, wenn er zu Besuch bei seinem Bruder in Greifswald ist, bekommt er als Antwort das Kinoprogramm von Greifswald.
Die Angabe des Ortes erfolgt hier durch den Stadtnamen. In der Realität sind dagegen viele verschiedene Arten von Ortsangaben vorhanden. Um dennoch mit Ortsangaben arbeiten zu können, müssen diese entsprechend strukturiert werden. Die Struktur von Ortsangaben wird am Beispiel des Ortskontextes für die mobile Walddatenerfassung im Kapitel 4.2 genauer erläutert.

Da der mobile Nutzer in der Lage ist nach Belieben seinen Ort zu ändern, müssen Möglichkeiten vorhanden sein, mit denen man den Aufenthaltsort eines Nutzer in einer mobilen Umgebung feststellen kann.

Ermittlung des Nutzerortes durch direkte Funkverbindungen

Die Genauigkeit der Ortsinformation ist von entscheidender Bedeutung. Die in der Einführung erläuterte Struktur von mobilen Netzten ist Grundlage einer Möglichkeit zur Ortsbestimmung. Für jeden mobilen Nutzer, der sich in einem mobilen Netz befindet, ist die Funkzelle bekannt,

in der er sich befindet. Im allgemeinen wird diese durch eine eindeutige Identifikationsnummer für jede Funkzelle angegeben. Die Größe der Funkzellen beträgt in den heutigen Anwendungsfällen im Durchmesser 1-5 km. Ortsdaten dieser Granularität lassen sich nur in großflächigen strukturierten Ortshierarchien anwenden. Als Beispiel kann man sich die Struktur des Mobilfunknetzes E-Plus in Rostock anschauen. Die Stadt wird von ca. 15 Basisstationen abgedeckt, die zum Bereich Rostock zusammengefaßt werden können. Die Funkzellen der Basisstationen können den Stadtteilen zugeordnet werden.
Die genaueste Ortsangabe ist dann der Stadtteilname. Dabei kommt es an Stadtteilgrenzen zwangsläufig zu Überschneidungen zwischen den Sende- und Empfangsbereichen der Funkzellen. In diesen Fällen wird das Funktelefon bei der Station angemeldet, die das Sendesignal vom Funktelefon am stärksten empfängt. Eine genauere Angabe des Ortes, wie z.B. der Name der Straße, in der man sich befindet ist mit diesem System nicht zu erreichen.
Die Genauigkeit dieser Ortsangaben ist meist unzureichend für konkrete Anwendungsbeispiele deshalb wird größtenteils das GPS System zur Positionsbestimmung eingesetzt.

Das Satellitennavigationssystem GPS (Global Positioning System)

Es gibt zur Zeit zwei Satellitennavigationsysteme die beide militärischen Ursprungs sind.
Dies sind das amerikanische Global Positioning System GPS und das russische Global Orbiting Navigating Satellite System GLONASS.
Das Grundgerüst des GPS bilden 24 Satelliten, die die Erde in etwa 20000 km Höhe umkreisen. Zu jedem Zeitpunkt sind an jedem Ort der Erde mindestens vier Satelliten sichtbar.
Die Satelliten senden fortwährend Signale zur Erde aus, die dann von GPS Empfängern ausgewertet werden können. Die Signale enthalten eindeutigen Code für jeden Satelliten, seinen Standort und eine Zeitmarke. Aus der Laufzeit der Signale und der Lichtgeschwindigkeit wird die Entfernung der Empfänger zum Satelliten bestimmt. Aus den Entfernungen und Richtungen zu mindestens vier Satelliten wird die geographische Position des Empfängers berechnet.
Die systembedingte Genauigkeit beträgt etwa ±20 m. Das GPS befindet sich aber im Besitz des amerikanischen Verteidigungsministeriums. Dieses läßt zwar die zivile Nutzung zu, verfälscht aber durch nachträgliche spezielle Modulationen die Signale so, daß nur noch Genauigkeiten von ±100 m zu erreichen sind.

Ortskontext und GPS

Im PDA des mobilen Nutzers ist ein GPS - Empfänger integriert, der nach dem zuvor beschriebenen Verfahren die „exakte“ geographische Position des Nutzers berechnet. Die ermittelten Daten liegen als Werte des Gradnetzes der Erde vor und sind somit für konkrete Anwendungen noch unbrauchbar.
Die Umwandlung der geographischen Daten in Ortsangaben, die dem Datenmodell der Anwendung entsprechen, werden durch Einbeziehung des Ortskontextes erreicht. Dazu werden Zuordnungstabellen benötigt, die vom Ortskontext bereitgestellt werden müssen.

Der Aufbau und die Größe der Zuordnungstabellen ist von dem zu Grunde liegenden Ortsdatenmodell abhängig. Für ein Modell mit hoher Granularität der Ortsangaben ist die Zuordnungstabelle größer als für ein Modell mit geringer Granularität.
Der Aufbau der Tabelle ist dagegen von der Struktur der Ortsdaten im Modell abhängig. Eine Tabelle für eine flache Ortsstruktur ist einfacher strukturiert als eine Tabelle, die hierarchisch strukturierte Ortsdaten verwalten muß.

Beispiel für die Ortsbestimmung bei der Waldatenerfassung
Ein Forsteinrichter befindet sich mit seinem Rechner in einem Forstrevier und möchte für einige Abteilungen die dazugehörigen Walddaten abrufen. Das Forstrevier wird von mehreren Funkzellen abgedeckt. Die für die aktuelle Funkzelle zuständige Basisstation hat die Information, daß sich der Förster in ihr aufhält. Im konkreten Anwendungsfall umfaßt eine Funkzelle mehrere Abteilungen des Reviers.

Im folgenden stellt der Forsteinrichter eine Anfrage auf die Informationen der Abteilung, in der er sich befindet. Damit die Anfrage sinnvoll erweitert werden kann, muß das System die geographische Position des Forsteinrichters zur Verfügung haben.
Diese Aufgabe übernimmt der Ortsserver, dem die Basisstation der aktuellen Funkzelle zugeordnet ist. Über die Funkverbindung zum PDA des Nutzers wird seine vom GPS-Empfänger ermittelte Position abgefragt und im Ortskontext gespeichert.
Diese wird als eine Geographische Position in der Form: geographische Länge und Breite geliefert. Durch Einbeziehung der Zuordnungstabellen des Ortskontextes werden die Angaben in eine Adresse der Form (Forstamt, Revier, Abteilung, Unterabteilung, Teilfläche) umgewandelt, die dann dem Datenbanksystem zur Anfrageerweiterung zur Verfügung steht.

Andere Ansätze zur Ortsbestimmung

Ein weiterer Ansatz zu Lokalisierungsstrategien wird in [LoeMa96] vorgestellt. Dabei werden flächenmäßig kleinere Ortsstrukturen betrachtet.

Grundlage dieses Vorschlags ist ein Gebäudeüberwachungssystem. Die Ortsbestimmung wird durch ein System von Infrarotsensoren und Infrarotsendern realisiert. Die Sensoren sind überall im Gebäude verteilt. In jedem Raum, jedem Flur und auch in jedem Fahrstuhl sind solche Sensoren installiert.

Die Funktionsweise des Systems ist denkbar einfach.

Jede im Gebäude befindliche Person bekommt einen Infrarotsender, z.B. in Form einer Ausweiskarte. Betritt man einen Raum mit diesem Sender, so registriert der in diesem Raum vorhandene Sensor, die Nummer der Ausweiskarte. Somit ist dem System die Raumnummer bekannt, in der sich die Person aufhält.

Die gleiche Architektur wird in [Schill] betrachtet. Hier werden schon mehrere Vorschläge für die Verwendung von Kontexten gemacht. Es wird dabei zwischen der manuellen und automatischen Kontextbenutzung unterschieden. Es werden Beispiele für getriggerte Aktionen gegeben, die z.B. bei Betreten eines Raumes ausgeführt werden sollen.

Die Wahl der technischen Realisierung der Ortsbestimmung wird hauptsächlich durch die erforderliche Genauigkeit der Ortsangaben, sowie die Struktur der zu verarbeitenden Ortsdaten bestimmt.

So läßt sich das Infrarotsystem nur für Systeme anwenden, die eine verhältnismäßig kleine Grundfläche haben, da die Infrarottechnik nur über geringe Entfernungen funktioniert. Ein Vorteil dieser Methode liegt darin, daß die aufwendige Umrechnung der geographischen Position in Datenbankangaben entfällt, da die Struktur des Systems unmittelbar auf die Ortsangaben des Datenbanksystem abgebildet werden kann.

Das GPS System ist eher für großflächige Anwendungsszenarien geeignet, wie z.B. Verkehrsleitsysteme im Straße und Flugverkehr, LKW-Überwachung und auch die Mobile Walddatenerfassung.

4.1.2 Ortsabhängige Verfügbarkeit von Informationen

Die Verfügbarkeit von Informationen, bezieht sich auf Eigenschaften die bestimmten Orten zugeordnet sind. Fragt der Nutzer an einem Ort A nach einer Eigenschaft die A erfüllt, so kann ihm der Wert der Eigenschaft als Ergebnis geliefert werden. Stellt er die gleiche Frage an dem Ort B, kann der als Wert der Eigenschaft einen Nullwert oder ein damit gleichwertiges „nicht vorhanden“ haben.
Damit der Nutzer dennoch einen Nutzen aus der gestellten Anfrage ziehen kann, kann die vorhandene Anfrage durch Veränderung der Ortsangabe so angepaßt werden, daß sie eventuell ein Ergebnis liefert, mit dem der Nutzer mehr anfangen kann. Damit die Anfrage sinnvoll verändert werden kann, benötigt das System Angaben über die Umgebung des Nutzerortes. Von großem Interesse ist dabei die Entfernung des Nutzerortes zu allen anderen Orten.
Der Ortskontext muß also Methoden zur Verfügung stellen, die eine Bestimmung der Entfernung nach relativen Gesichtspunkten ermöglicht.
Der Begriff Entfernung wird in diesem Zusammenhang immer als Entfernung des Nutzerortes zu geographischen Ortsangaben von angefragten Daten gesehen.
Eine weitere Interpretationsmöglichkeit der Entfernung, ist die Betrachtung der Entfernung vom Nutzerort zum physikalischen Ort der Daten, z.B. zum Server auf dem die Datenfragmente liegen. Diese Entfernung wird an dieser Stelle nicht weiter betrachtet, da der transparente Zugriff des Nutzers auf alle Daten voraussetzt wird.

Die Entfernungsrelation (Nahrelation)

Die Entfernungsrelation ermöglicht die Betrachtung der Umgebung des Nutzerortes. Das Ziel der Umgebungsbetrachtung ist eine Erweiterung der Ergebnismenge im Vergleich zur Ergebnismenge der Anfrage mit dem konkreten Nutzerort.
Statt nach dem Nutzerort zu selektieren, wird nach allen Orten, die bezüglich der Entfernungsrelation einen bestimmten Abstand zum Nutzerort nicht überschreiten. Diese Orte liegen „in der Nähe“ der Nutzerortes.
Weiterhin wird auf diesen „nahen Orten“ dann die ursprüngliche Selektion ausgeführt.
Endergebnis der Ortsselektion sind nur noch Orte, die die am Nutzerort nachgefragten Bedingungen erfüllen.

Beispiel:

Herr Schulz befindet sich in Potthagen und stellt die Frage nach dem aktuellen Kinoprogramm. Die um den Nutzerort erweiterte Anfrage lautet dann: „Gib mir das Kinoprogramm, von Potthagen“. Da es in diesem Ort kein Kino gibt, erhält er normalerweise ein leeres Ergebnis bzw. die Mitteilung, daß es dort kein Kino gibt.
Unter Verwendung der Entfernungsrelation wird die Anfrage jetzt verändert. Sie lautet: „Gib mir das Kinoprogramm, von allen Orten, die in der Nähe von Potthagen liegen“.

Wie groß die maximale Entfernung gewählt wird, ist vom Nutzer selbst, den örtlichen Gegebenheiten des Nutzerortes und der „nahen Orte“ abhängig. Diese Angaben werden während der Benutzung des Systems aus den Voreinstellungen im Nutzerkontext gewonnen, so daß sie für jeden Nutzer eine individuelle Einstellung ermöglichen.

Die Erreichbarkeitsrelation
Die Betrachtung der Entfernung der Orte zum Nutzerstandort ist eine Möglichkeit, vom Nutzer bevorzugte Ergebnisobjekte zu erhalten. Um die Voraussetzungen und Eigenschaften eines Nutzers aber besser einbeziehen zu können, ist in einigen Fällen auch die Betrachtung der Erreichbarkeit bestimmter Orte von Nutzen.

Was die Erreichbarkeit als Auswahlbedingung darstellt, hängt im entscheidenden Maße von der Struktur der modellierten Daten als auch von der Mobilität des Nutzers ab.
Erreichbarkeit kann einmal als Erweiterung der Nahrelation angesehen werden. Besitzt der Nutzer ein Auto, ist er natürlich viel mobiler als wenn er nur ein Fahrrad zur Verfügung hätte. Demzufolge ist der Bereich der Erreichbarkeit eines Autofahrers mit z.B. 20-30 km wesentlich größer als der eines Radfahrers ca. 5-10 km. Die Größe des Anfrageergebnisses steigt unmittelbar mit der Größe der Mobilität eines Nutzers.
Außerdem kann die Erreichbarkeit als zeitliche Angabe gesehen werden. Sie gibt an, ob ein Ort in vertretbarer Zeit vom Nutzerort aus zu erreichen ist oder nicht.
Diese Mobilitätsangaben werden ebenfalls aus dem Nutzerkontext gewonnen und in die Anfragebearbeitung mit einbezogen.

4.1.3 Ortsabhängiger Zugriff

Jeder mobile Nutzer ist auf einem sogenannten Heimatserver registriert und hat auf diesem bestimmte Rechte für die Daten in diesem Bereich. Die Mobilität erlaubt dem Nutzer sich zu-

sätzlich zum Heimatserver auch unter einem anderen Ortsserver anzumelden. Die auf dem Heimatserver geltenden Rechte sind an diesem Server nicht von Bedeutung, da der Nutzer auf diesen Servern andere Rechte besitzt. Denkbar ist, daß man sich dann nur als Besucher anmeldet.
Zur Realisierung dieser Rechte kann man die Rollenverteilung nutzen. In heutigen Datenbanksystem ist eine solche schon implementiert. Beim Anmelden auf einem fremden Server muß dieser sich vom Heimatserver des Nutzer dessen Rollenprofile geben lassen und kann in Abhängigkeit davon seine eigenen Rechte für den Nutzer vergeben. In relationalen Datenbanksystemen kann man zusätzlich das Sichtenkonzept benutzen. Dadurch kann dem Nutzer eine logische Sicht auf die Daten gegeben werden, für die er an einem bestimmten Ort authorisiert ist. Die Einstellung dieser Sichten erfolgt also nutzerabhängig und bezüglich bestimmter Orte. Zur Anwendung des Sichtenkonzeptes ist es nötig das Rollenkonzept um die Ortsangaben zu erweitern.
In der herkömmlichen Rollenverteilung, wird jedem Nutzer einer Rolle zugeordnet.

Nutzer	Rolle
Nutzer 1	Administrator
Nutzer 2	Besucher

Tabelle 1 : Herkömmliche Rollenzuordnung

Eine einfache Möglichkeit der Erweiterung der Rollenverwaltung wird in der folgenden Tabelle dargestellt.

Nutzer	Rolle	Ort
Nutzer 1	Administrator	Revier 1
Nutzer 1	Besucher	Revier 2
Nutzer 2	Administrator	Revier 2
Nutzer 2	Besucher	Revier 1

Tabelle 2 : Rollenzuordnung mit Ortsangabe

Die Ortsangaben im Rollenprofil sind von der jeweils vorhandenen Ortsstruktur abhängig. Zu den Rollenprofilen könne jetzt bestimmte Sichten definiert werden, auf denen der Nutzer an dem jeweiligen Ort arbeitet.

Beispiel:
Der Waldarbeiter Herr Holz ist für das Forstrevier Greifswald zuständig. Hier ist er der Hauptförster und erhält uneingeschränkten Zugriff auf alle Daten dieses Reviers. Er demzufolge kann Anfragen sowie Updates auf den Daten dieses Reviers ausführen. Bewegt er sich aber ins Revier Stralsund, so erkennt das System den Ortswechsel und weist ihm seine neue Rolle zu. Gleichzeitig mit der Rollenänderung erhält der Herr Holz dann die Besuchersicht auf die Daten von Stralsund. Als Besucher kann er jetzt zwar bestimmte Anfragen stellen, darf aber keine Daten mehr ändern.

Durch eingeschränkte Rechte hat der Nutzer verschiedene Daten und Funktionen nicht zur Verfügung. Eine mögliche Anwendung des Wissens über die neue Rolle des Nutzers ist die Anpassung der Arbeitsumgebung auf seinem mobilen Gerät. Ein Beispiel ist die Veränderung der Funktionalität der Nutzerapplikation.
Im Fall des Waldarbeiters entfällt zum Beispiel der Menüpunkt „Daten ändern“ und erscheint nicht mehr auf dem PDA. Weiterhin können auch bestimmte Klassen und Objekte für den Nutzer unzugänglich sein.

4.1.4 Weitere ortsabhängige Angaben

Da sich mobile Nutzer im System bewegen können, und die zellularen Netze nur Funkzellen beschränkter Größe haben, ist es sehr wahrscheinlich, daß ein Nutzer häufig zwischen den verschieden Funkzellen wechselt. Wenn sich beim Ortswechsel des Nutzers auch der für ihn zuständige Server ändert, gibt es verschiedene Strategien, wie die Nutzerdaten von Server zu Server übergeben werden können und wie man den Nutzer nach seiner Bewegung lokalisiert. Dazu werden in [Imil92/2] Techniken betrachtet, die Bewegungsprofile der Nutzer als Basis haben. Diese dienen zur schnelleren und kostengünstigen Lokalisierung von Nutzern.
Ein ähnlichen Ansatz kann im Ortskontext verwendet werden um das Performanceverhalten des Systems zu verbessern.
Ein Beispiel dafür ist ein ortsabhängiges Bewegungsprofil eines Nutzer, welches angibt, zu welchen Orten er sich mit hoher Wahrscheinlichkeit als nächstes bewegt. Durch diese Information können Daten über diese Orte schon in einen Zwischenspeicher(Cache) geladen werden, so daß bei einem möglichen Eintritt des Nutzer in diesen Ort, die Netzlast und die Antwortzeit des Systems verringert wird.

Um zu entscheiden welche Orte in Frage kommen, gibt es mehrere Möglichkeiten. Erstens kann für den Nutzer ein konkretes Bewegungsprofil angegeben sein, in dem sein festgelegter Weg steht. Aus Sicht des Datenschutzes ist diese Möglichkeit eher ungeeignet, da es einer Überwachung des Nutzers gleichkommt.
Befindet sich ein Nutzer häufiger an gleichen Orten, so kann ein statistisches Benutzerprofil erstellt werden, mit dem dann anhand von Wahrscheinlichkeiten die nächsten Orte ermittelt werden können.

Beide Arten von Bewegungsprofilen beinhalten bis jetzt nur die Orte, zu denen sich der Nutzer bewegt. Es ist aber keine Information vorhanden, zu welchem Zeitpunkt er sich sicher bzw. wahrscheinlich an einen Ort begibt. Ohne die Zeitangaben ist eine sinnvolle Zwischenspeicherung für Nutzerbewegungen über größere Entfernungen nicht möglich.

Die einfachste Art eine Vorausbetrachtung der Bewegung zu machen, ist die Benutzung des Ortskontextes zur Ermittlung der umliegenden Ort im Sinne der Ortsstruktur. Ist die Anzahl umliegender Orte nicht zu groß, so ist eine Zwischenspeicherung sinnvoll, weil es als wahrscheinlich angesehen werden kann, daß sich der Nutzer in einen angrenzenden Bereich bewegt.

4.2 Struktur des Ortkontextes

4.2.1 Definition des Ortskontextes

Nachdem in den vorangegangenen Kapiteln die Eigenschaften und Funktionen des Ortskontextes betrachtet wurden, soll jetzt eine Definition des Ortskontextes gegeben werden.

Die Definition spezifiziert die einzelnen Komponenten und den Aufbau des Ortskontextes.

Die Grundlage des Ortskontextes bildet der geographische Ort eines Nutzers.
Die Form dieser Ortsangabe ist abhängig vom speziellen Anwendungsfall.
Im hier betrachteten Walddatenbeispiel, sowie in anderen flächenbasierten Anwendungsszenarien wird der geographische Ort durch die Angabe der Längen- und Breitengrade im Gradnetzsystem der Erde repräsentiert. Dies ergibt sich durch die Nutzung des im Kapitel 3 erläuterten GPS Systems.

In welcher Form die Ortsdaten letztendlich zur Verfügung stehen, hängt von dem jeweilig benutzten Ortsbestimmungssystem ab. Im Folgenden wird der Ortskontext durch seine Funktionen definiert.

1) Ermittlung des Nutzerortes im Gradnetzkoordinaten als Grundlage des Ortskontextes.

2) Bereitstellung des Nutzerortes in der im jeweiligen Datenmodell verwendeten Form.
Zur Realisierung dieser Funktion werden Mappingtabellen benutzt, die eine Zuordnung zwischen geographischen Orten und Orten des Datenbestandes ermöglichen.

3) Es werden Möglichkeiten zur Ermittlung der Ortsabhängigkeit von Daten zur Verwendung in der Anfragebearbeitung und Aggregation bereitgestellt.
Die Grundlage hierbei bilden die hierarchischen Ortsdaten, die zur Laufzeit des Systems ausgewertet werden.

4) Erweiterte Funktionalität zur ortsabhängigen Veränderung der Anfrage im Anfragebearbeitungsprozess und zur Beeinflussung der Ergebnisdaten im Aggregationsprozeß.

5) Verarbeitung von Bewegungsprofilen in Form von Ladelisten zur Unterstützung der vorausschauenden Zwischenspeicherung. Hier ist eine Zusammenarbeit mit dem Informationskontext und dem Nutzerkontext notwendig.

Die hier beschriebene Struktur dient als Grundgerüst des Ortskontextes und kann in jedem Ortskontext verwendet werden. Die konkrete Umsetzung der Ortskontextes ist aber stark von dem Anwendungsfall abhängig, in dem er benutzt werden soll.

Der Ortskontext ist im Bezug auf die im Abschnitt 2.2 gegebene Klassifikationen bezüglich der Einbeziehung von Kontexten in alle Klassen einzuordnen.
Vor der Systemnutzung wird der Ortskontext benutzt um Voreinstellungen für die folgende Sitzung vorzunehmen. Dazu gehört die Verarbeitung von Bewegungsprofilen für die Zwischenspeicherung aber auch die Einstellung von Applikationsfunktionalität, die am Nutzerort vorhanden ist. Außerdem kann durch das Arbeiten mit ortsabhängigen Sichten, schon vor Ausführung der Anfrage an das System, eine logische Verringerung der Datenmenge erreicht

werden, indem Sichten auf bestimmte Hierarchieebenen beschränkt werden. Wenn sich ein Waldarbeiter in einem bestimmten Revier befindet und mit seinem Fragenkatalog nur auf Daten dieses Reviers arbeitet, kann eine Sicht auf die Daten des Reviers benutzt werden. Diese Sicht ist nichts weiter als eine vorgezogene Selektionen nach dem bestimmten Revier.
Während der Systemnutzung wird der Kontext in die Anfragebearbeitung und den Aggregationsprozess eingebunden. Und zwischen den Sitzungen wird wieder die Bewegungsprofilverarbeitung ausgeführt.

Bezüglich der Klassifizierung nach der Häufigkeit der Datenänderung ist der Ortskontext ein dynamischer Kontext, was in der Mobilität des Nutzers begründet ist.

4.2.2 Modell des Ortskontextes (allgemeiner Teil)

Um die in den vorangegangenen Kapiteln erläuterten Ortsangaben für Datenbanksystems nutzbar zu machen, mußte eine geeignete Form der Darstellung für den Ortskontext gefunden werden. In Kapitel 4.5 wird die Kontextarchitektur und die Integration des Ortskontextes in die Personenkontextumgebung ausführlicher erläutert.
An dieser Stelle wird der Ortskontext mittels der CORBA IDL definiert und somit als Interface für den Kontextserver und die Anwendungen bereitgestellt.

Wie bei der Definition festgestellt wurde, ist der Nutzerort als Grundlage des Ortskontextes zu sehen. Um diesen anwendungsgerecht beschreiben zu können, wird die Klasse `GeographischePosition` eingeführt. Sie dient als Grundklasse eines jeden GPS-basierten Ortskontextes.

```
interface GeographischePosition : {
      GeographischeZahl Laenge_und_Breite;
      void        aktualisiere_Position(in GeographischeZahl
                  Laenge_und_Breite);
      boolean     in_der_Nähe (in float Radius,in GeographischeZahl
                  Laenge_und_Breite)
      float       entfernung (in GeographischeZahl Laenge_und_Breite)
}
```

Der Standort eines Objektes wird durch das Attribut `Laenge_und_Breite` bestimmt. Der Typ `GeographischeZahl` ist eine Struktur in der die üblichen Gradnetzkoordinaten ange-

geben werden können. Diese Struktur setzt sich aus der Längen- und Breitenangabe, sowie den dazugehörigen Minuten- und Sekundenangaben zusammen.

Die Methode `in_der_Nähe` liefert genau dann den Wert wahr, wenn die Entfernung zwischen dem übergebenen Standortparameter und dem eigenen Standort nicht größer als der übergebene Radius ist.
Die Methode `entfernung` bestimmt die Entfernung in Metern zwischen der aktuellen Position des Objektes und einer anderen, die als Parameter übergeben wird.
Weil sich bei mobilen Nutzern der Ort häufig ändert, gibt es noch die Methode `aktualisiere_Position`.
Die Aufgabe und Arbeitsweise dieser Methode wird kurz beschrieben. Ändert der mobile Nutzer seine Position, so erkennt sein GPS Empfänger am mobilen Gerät, daß die Position sich geändert hat. Das Datenbanksystem muß auch davon in Kenntnis gesetzt werden, damit es darauf gegebenenfalls reagieren kann. Wenn sich also die Position des Nutzers ändert, wird automatisch durch den GPS Empfänger diese Methode mit den neuen Gradnetzkoordinaten aufgerufen. Dieser Methodenaufruf geschieht nicht direkt, sondern wird in der spezielleren Ortskontextklasse „Ortkontext_Wald“ (wird im nächsten Kapitel vorgestellt) benutzt.

Bei der Definition des Ortskontextes wurde festgestellt, daß der Kontext die Möglichkeit der Bestimmung der Kontextabhängigkeit einer Anfrage bzw. eines Ergebnisses unterstützen muß. Dazu sind Informationen zum konkreten Datenmodell notwendig. Für den Ortskontext ist es wichtig zu wissen, welche Attribute des Datenmodells Ortsbezug haben. An dieser Stelle wird der Ortsbezug noch vereinfacht, indem folgende Aussage als Definition genommen wird.
Def.: Ein Attribut ist dann ortsabhängig, wenn es ein Attribut der Ortshierarchie ist.
Somit ist die Menge der ortsabhängigen Attribute relativ klein und läßt sich in einer eigenen Klasse als eine Art Metadaten ablegen.
Zu diesem Zweck wird die neue Klasse Hierarchieebene eingeführt, die eine Liste von Hierarchieattributen verwaltet.

```
struct Hierarchieattribut {
      attribute string Name;
      attribute string Vater;
      attribute string Sohn;
}
```

```
interface Hierarchieebene {
      attribute sequence<Hierarchieattribut> Hierarchieliste;
      void add(in Hierarchieattribut Attribut);
      void remove(in Hierarchieattribut Attribut);
      boolean is_Hierarchieattribut(in string Name);
      string get_Vater(in string Name);
      string get_Sohn(in string Name);
}
```

Jede Hierarchieebene ist dabei ein Listeneintrag und hat neben dem eigenen Attributnamen jeweils die Attribute Vater und Sohn. Die Methoden add und remove fügen eine Hierarchieebene hinzu bzw. entfernen eine Ebene aus der Liste.
Wie man sieht, müssen die Hierarchiedaten in Baumstruktur vorliegen, damit sie in dieses Schema passen. Weil dies in den meisten Anwendungsfällen auch der Fall ist, wird hier vereinfacht angenommen, daß Ortsdaten für den Ortskontext immer in einer Baumstruktur vorliegen.

Mit der Methode `is_Hierarchieattribut`, kann festgestellt werden, ob ein Attribut ortsabhängig nach der zuvor gemachten Definition ist oder nicht.
Die Methoden `get_Vater` und `get_Sohn`, dienen der verbesserten Anfragebearbeitung die in Kapitel 5.1 behandelt wird.

4.2.3 Ortskontextmodell für die Mobile Walddatenerfassung

Im Beispiel der Mobilen Walddatenerfassung ist die Ortsstruktur wie folgt gegliedert

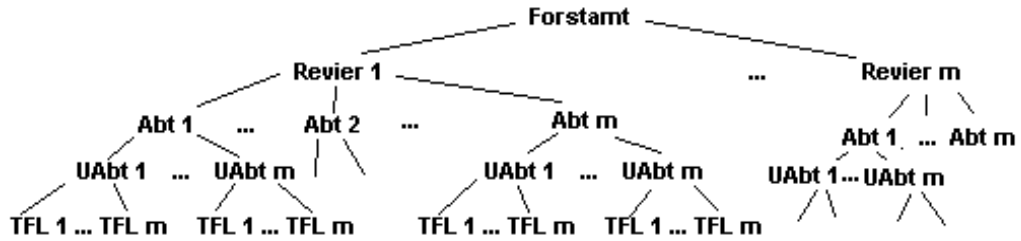

Abbildung 2: Ortshierarchie im Walddatenbeispiel

In der Abbildung sind die einzelnen Ebenen der Hierarchie zu sehen. Die Hierarchieliste für das Walddatenbeispiel sieht also folgendermaßen aus:

Attributname	Vater	Sohn
Forstamt	-	Revier
Revier	Forstamt	Abt
Abt	Revier	Uabt
Uabt	Abt	TFL
TFL	UAbt	-

Tabelle 2 : Hierarchieebenen in Tabellenform

Eine Anfrage eines Forstarbeiters mit der konkreten geographischen Position ist für das Datenbanksystem wenig sinnvoll, da es mit einer solchen Ortsangabe nichts anfangen kann.
Damit die geographische Position des Forstarbeiters in eine Ortsangabe der zuvor genannten Struktur umgewandelt werden kann, benötigt man Mappingtabellen.
Über diese erfolgt die Zuordnung von einer geographischer Position in eine Position der Form Forstamt, Revier, Abteilung, Unterabteilung, Teilfläche.

Der Aufbau der Mappingtabellen erfolgt nach der Struktur der Ortshierarchie. Für jede Hierarchieebene gibt es eine Tabelle, die für alle Elemente dieser Ebene eine Polygonbeschreibung enthält. Die Eckpunkte werden dabei als Geographische Zahlen wie zuvor definiert angegeben. Für die oberste Ebene der Forstämter heißt dies also, daß jedes Forstamt durch ein Polygon beschrieben ist.
Da die Flächen der Forstämter paarweise disjunkt sind, kann zu einer Position eine eindeutige Zuordnung zu einen Forstamt gemacht werden.
Im Walddatenbeispiel liegen die geographischen Daten der Forstämter in einem HPGL ähnlichen Daten Format vor. Die Struktur der Daten sieht wie folgt aus.

```
KartenDaten
PPO -746 -2202
PPV 4516 6015
PPO -3091 -1009
PPV 4514 6014
REV Torfbruecke
PU2530 3550;
PD2343 2266;
PD2076 2060;
...
PD2530 3550;
ABT 102
PU-770 616;
PD-798 689;
...
PD-770 616;
```

Dieses Datenformat ist sehr einfach gehalten und erfordert eine lineare Suche durch die gesamten Daten. Eine erweiterte Datenstruktur für Kartendaten, die besser die Hierarchiestruktur der Daten darstellt, wird in [Boern] vorgestellt.
Zur einfachen Integration der Mappingtabellen ist es sinnvoll, diese als CORBA Objekte zu modellieren und somit über ein mit CORBA IDL definiertes Interface ansprechbar zu machen.

Nachdem die Grundlagen für den Ortskontext jetzt vorhanden sind, wird nun der Ortskontext für das Walddatenbeispiel beschrieben. Aus der Hierarchie der Walddaten ergibt sich die folgende Struktur, mit der eine Teilfläche als Blatt im Hierarchiebaum adressiert werden kann.

```
struct Adresse{
string Forstamt;
string Revier;
string Abteilung;
string Unterabteilung;
string Teilflaeche;
}
```

Die Beschreibung des Ortskontextes erfolgt in CORBA IDL. Ein Ortskontext ist also ein CORBA Objekt, auf das über die definierten Funktionen zugegriffen werden kann.
Die Erläuterung des Kontextservers, der die Kontextobjekte bereitstellt, erfolgt in Abschnitt 5.2.1.

```
interface Ortskontext_Wald : GeographischePosition {
      erzeuge_Adresse();
      Adresse liefere_Adresse();
      float Flaechenentfernung (in Adresse Wald_adresse);
      boolean nahe_Teilflaeche(in key schluessel);
      boolean nahe_Abteilung(in Adresse Wald_adresse);
      sequence <Adresse> werteBenutzerprofil();
}
```

Die Beschreibung des Ortskontextes enthält an dieser Stelle nur die Methoden, die ein Ortskontextobjekt bereitstellt. In der Implementierung des Objektes werden zusätzlich die 3 folgenden Attribute definiert.

```
Adresse Wald_Adresse
```

```
Integer schluessel;
```

Die `Wald_Adresse` ist vom Typ Adresse ist eine Struktur, die eine Teilfläche durch Angabe des ihres Suchpfades im Hierarchiebaum der Ortsdaten beschreibt. Weil im vorhandenen Datenmodell für die mobile Walddatenerfassung jeweils eine Schlüsselnummer pro Teilfläche vergeben wurde, ist die unter `schluessel` gespeicherte Nummer gleichwertig zum Attribut `Wald_Adresse`. Um die Methoden der Hierarchieebenen benutzen zu können, muß der Ortskontext zusätzlich die Referenz auf ein CORBA Hierarchieebenenobjekt ermitteln.

Die Methode `erzeuge_Adresse()` erzeugt aus den Angaben zur geographischen Position eine Waldadresse. Dabei werden die zuvor beschriebenen Mappingtabellen benutzt, in der die Struktur der Walddaten als Menge von Polygonen dargestellt ist.
Die Methode `liefere_Adresse()` gibt die `Wald_Adresse` des aktuellen Kontextobjektes zurück.

Zur Bestimmung der Entfernung zwischen der aktuellen Position des Nutzer und einer anderen Fläche bietet der Ortskontext die Methode `Flaechenentfernung`. Die `Flaechenentfernung` basiert auf der Methode Entfernung der Klasse `Geographische_Position`. Um mit dieser Methode Flächen der verschiedenen Hierarchiestufen betrachten zu können, wurde als Parameter bewußt die Adreßstruktur gewählt und nicht der Schlüssel, der eine Teilfläche kennzeichnet.
Durch diese Wahl der Parameter kann jetzt z.B. die Entfernung einer bestimmten Abteilung zum aktuellen Standpunkt des Nutzer genauso ermittelt werden, wie die einer Teilfläche.
Die Implementierung unterscheidet dabei in Abhängigkeit von der übergebenen Adresse, auf welcher Ebene die Entfernungsberechnung erfolgt.
Die Berechnung der Entfernung einer Teilfläche zum aktuellen Standpunkt hätte als Adressenparameter eine Adreßstruktur, in der jedes Hierarchieattribut einen nichtleeren Wert hätte.
Für die Entfernungsberechnung einer Abteilung dagegen, erhalten die beiden Attribute Unterabteilung und Teilfläche einen Leerstring als Wert.
Zur Entfernungsberechnung wird letztendlich die aus der `Geographische_Position` geerbte Methode `entfernung` benutzt.

Wie bei der Klasse der Geographischen Position gibt es auch hier die Methoden, die zur Bestimmung der „Nähe“ zweier Orte dienen.

Es gibt verschiedene Möglichkeiten die Funktion „in der Nähe“ zu implementieren.
Die einfachste Möglichkeit zu entscheiden, ob eine Teilfläche in der Nähe einer anderen liegt, läuft wie im allgemeinen geographischen Fall auf Entfernungsmessung hinaus. Man bildet für jede zu untersuchende Teilfläche den Polygonschwerpunkt und berechnet die Entfernung zum aktuellen Polygonschwerpunkt. Ist die Entfernung innerhalb der vorgegebenen Größe, so wird als Ergebnis der Funktion wahr geliefert.
Eine weitere Möglichkeit die Funktion „in der Nähe“ zu implementieren ergibt sich aus der Struktur der Walddaten. So kann man alle an eine Teilfläche grenzenden Teilflächen als „nah“ ansehen.
Um in diesem Fall eine Entscheidung zu treffen, wann zwei Teilflächen unmittelbar nebeneinander liegen, muß man sie mit geeigneten Algorithmen auf gemeinsame Kanten untersuchen. Das ist möglich, weil die Flächen als Polygone angegeben sind.

Ein großer Nachteile dieser vorgeschlagenen Implementation der „Nähe“ ist die mathematische Komplexität, die bei der Berechnung von gemeinsamen Kanten auf großen Datenmengen anfällt.

Um den Ortskontext effektiver zu gestalten, ist eine Erweiterung des Modells sinnvoll. Ziel dieser Erweiterung ist, für jede Teilfläche eine Beziehung „nah“ ins Datenmodell aufzunehmen.
Im vorhandenen Datenmodell des Walddatenbeispiels, zu finden im Anhang von [Wendt97], repräsentiert die Klasse `Adress` die Menge aller Teilflächen im System.
Die neu einzufügende Beziehung ist eine Beziehung der Klasse `Adress` auf sich selbst.
Jede Teilfläche kann mit n anderen Teilflächen in der „nah“ Beziehung stehen.
Das Anlegen der Beziehungen muß nur einmal vor Systemstart durchgeführt werden und kann entweder algorithmisch oder manuell geschehen. Zur automatischen Erstellung der Beziehungen kann der Algorithmus zur Berechnung gemeinsamer Kanten benutzt werden.

Diese Modellerweiterung läßt jetzt auch auf anderen Hierarchieebenen eine Betrachtung angrenzender Flächen zu. Das geschieht z.B. so:

Die Unterabteilung A grenzt an die Unterabteilung B genau dann, wenn es in A eine Teilfläche gibt, die an eine Teilfläche aus B grenzt.

Die Methode `werteBenutzerprofil(...)` soll als Grundlage für andere Dienste z.B. für die Zwischenspeicherung, die Orte liefern, die der Nutzer als nächstes benötigt.

4.2.4 Ortskontextmodell für die Möbelvertriebsfirma

Grundlage des Ortskontextes für das Möbelvertriebsbeispiel ist, da keine eigene andere existier, die geographische und verwaltungsrechtliche Struktur der Bundesrepublik Deutschland. Es gibt auch hier eine Hierarchie der Ortsdaten. Auf der obersten Stufe stehen die Bundesländer, dann folgenden die Land- und Stadtkreise, schließlich als kleinste Ortsangabe die Städte und Gemeinden. In Großstädten wie Berlin oder Hamburg ist eine weitere Verfeinerung der Ortshierarchie bis auf Stadtteilebene denkbar.

```
struct adresse {
  string Bundesland
  string Kreis
  string Ort
  string Stadtteil
}
```

Der Aufbau des Ortskontextmodells sieht dem Modell für die Walddatenerfassung sehr ähnlich. Ausgangspunkt ist die Klasse `GeographischePosition`, von der auch dieser Kontext abgeleitet wird. Wie im vorangegangenen Beispiel werden Mappingtabellen benötigt, die eine Umrechnung der geographischen Position eines Objektes in datenbanktaugliche Ortsangaben ermöglichen.
Die Methoden des Ortskontextes sind auch hier ähnlich dem vorangegangenen. Die Relation „in der Nähe“ muß dann für die einzelnen Stufen der Ortshierarchie definiert werden.

Im Unterschied zur Walddatenerfassung gibt es in der Möbelvertriebsfirma unterschiedliche Personen, die verschiedene Aufgaben zu erledigen haben. Wenn sich eine Person im System anmeldet, wird ihr die entsprechende Rolle zugewiesen.

Die jeweilige Rolle eines Nutzers hat Einfluß auf die Nutzung des Ortskontextes durch das System. Für einen LKW-Fahrer kann man sich den Ortskontext in Kopplung mit einem Verkehrsleitsystem vorstellen. Durch das System wird dem Fahrer aufgrund aktueller Verkehrsinformationen, wie Stau oder Straßenarbeiten, der nächste anzufahrende Punkt in Abhängigkeit von seiner derzeitigen Position geliefert. Nach erfolgter Auslieferung ermittelt das System automatisch sein nächstes Ziel und gibt ihm den Weg dorthin bekannt.

Der für den Vertreter gültige Ortskontext hat neben dem Aufenthaltsort auch die in_der_Nähe Funktion. So kann der Vertreter zum Beispiel die Frage nach allen Kunden stellen, die in der Umgebung seines aktuellen Aufenthaltsortes wohnen. Dazu benötigt der Ortskontext aber Informationen aus dem Nutzerkontext, welche Entfernung für den jeweiligen Vertreter als nah anzusehen ist. Interessant ist für den Vertreter vielleicht auch die geographische Entfernung von seinem Standpunkt zu den Kunden. Dazu wird wieder ein Mapping benötigt, welches umgekehrt zum ursprünglichen Mapping, einer Adresse aus der Datenbank eine Geographische Position zuordnet, die für eine Entfernungsberechnung benutzt werden kann. So könnte sich der Vertreter alle Kunden, die nicht weiter als 10 km von ihm entfernt sind, heraussuchen lassen und durch eine spätere Umordnung im Aggregationsschritt der Entfernung nach sortieren lassen.
Bei der im Beispiel zu Grunde gelegten Ortshierarchie ist es aber wahrscheinlicher, daß der Vertreter die Anfrage wie folgt formuliert: „Suche alle Kunden, die in der aktuellen Stadt wohnen“. Die aktuelle Stadt des Vertreters wird durch den Ortskontext ermittelt und die Anfrage damit erweitert.

Zusammenfassend kann man sagen, daß in der Möbelvertriebsfirma der Ortskontext nur in beschränktem Maße zum Einsatz kommt. Die Granularität der Ortshierarchie müßte wie bei einem Verkehrsleitsystem bis auf Straßenebene heraufgesetzt werden um genaue Angaben über den Nutzer und Kunden machen zu können.
Der Einsatz des Ortskontextes in der Möbelfirma beschränkt sich auf eine Standortbestimmung des Nutzers sowie die Entfernungsmessung zu anderen Kunden oder Lieferanten. Für die Nutzbarkeit des Kontextes ist im Gegensatz zum Walddatenbeispiel ein wesentlich höherer Aufwand bei der Erstellung der Mappingtabellen nötig.

Vergleicht man den Aufwand der Bereitstellung des Ortskontextes mit seinem Nutzen, so ergibt sich der Schluß, daß eine automatische Ermittlung des Ortes sich für diesen Anwendungsfall nicht lohnt.

4.3 Eigenschaften des Zeitkontextes

Wie im Abschnitt 2.4. kurz angesprochen sind Informationen mit Zeitbezug im gesamten System zu finden. Der Nutzer selbst hat eine Verbindung zur Zeit, da er eine Anfrage zu einem bestimmten Zeitpunkt stellt. Außer dem Nutzer können natürlich auch die Daten, auf denen er arbeitet, eine Zeitkomponente enthalten.

Grundsätzlich kann man zwei Gruppen von zeitabhängigen Daten unterscheiden. Zum einen sind es die in der Datenbank gespeicherten statischen Daten mit Zeitbezug. Als Beispiel sei hier der Bahnfahrplan genannt. Die Änderungshäufigkeit dieser Daten ist so gering, daß man sie vernachlässigen kann.
Der Fahrplan der Deutschen Bahn wird z.B. nur halbjährlich gewechselt, so daß die Daten über eine relativ langen Zeitraum konstant sind.
Diese Gruppe Daten enthält lediglich feste Zeitangaben, in unserem Fall die Abfahrts- bzw. Ankunftszeiten von Zügen. Bei Anfrage zu unterschiedlichen Zeiten werden die Zeitattribute der Daten zu Selektionskriterien und der Nutzer bekommt voneinander verschiedene Ergebnisse.

Die zweiten Gruppe sind Daten, die periodischen oder ereignisgesteuerten Änderungen unterworfen sind. Beispiele hierfür sind der Wetterbericht und auch die Börsenkurse. Bei Aktien an der Börse, gibt es bestimmte Zeiten, zu denen eine Aktualisierung ihres Wertes durchgeführt wird. Das Gültigkeitsdauer der in der Datenbank bestehenden Werte ist also von vornherein auf einen kurzen Zeitraum begrenzt. Diese Daten können als zeitdynamische Daten bezeichnet werden.
Um diese Art von Daten im Zeitkontext zu berücksichtigen, ist das Wissen über den Zeitpunkt der Änderung nötig.

Die wichtigsten Zeiteigenschaften werden in den folgenden Abschnitten betrachtet.

4.3.1 Zeitpunkt der Anfrage

Der Zeitpunkt zu dem ein Nutzer seine Anfrage stellt, ist ein wichtiger Faktor des Zeitkontextes. Wenn sich in mobilen Datenbanken die Daten häufig in zeitlich kurzen Abständen ändern, ist es wichtig den Zeitpunkt der Anfragestellung zu wissen. Als Beispiel kann man eine Fahr-

planauskunft der Deutschen Bahn betrachten. Angenommen der Nutzer befindet sich in Rostock und stellt die Anfrage „Wann fährt der nächste Zug nach Berlin“.
Zuerst fällt auf, das die Anfrage nicht vollständig ist, da zu einer Zugverbindung immer ein Abfahrts- und ein Ankunftsort gehören. Der Abfahrtsort wird durch die Einbeziehung des Ortskontextes gewonnen. Um ein befriedigendes Ergebnis für den Nutzer zu liefern, muß die Anfrage aber außerdem noch die aktuelle Zeit und das Datum beachten. Die Forderung nach der nächsten Verbindung kann in OQL folgendermaßen realisiert werden.

```
define mögliche_Verbindungen as
select  x from x in Verbindung
where x.Abfahrtzeit>aktuelle_Zeit and
      x.Abfahrtort=aktueller_Ort and
      x.Ankunftsort=Zielort and
      x.datum = Reisedatum;
```

Dies ist eine Selektion, die alle Verbindungen am gewünschten Tag vom Abfahrts- zum Zielort ermittelt, die nach der aktuellen Anfragezeit abfahren.
Nun wird die Verbindung aus den möglichen_Verbindungen selektiert, deren Abfahrtszeit kleiner ist, als die der anderen.

```
select x from x in mögliche_Verbindungen
where x.Abfahrtszeit = min(select y.Abfahrtzeit from y in mögli-
che_Verbindungen)
```

Das Ergebnis der Anfrage ist die erste Verbindung, die nach dem Zeitpunkt der Anfragezeit abfährt.
In diesem Beispiel hat der Zeitpunkt der Anfrage eine direkte Wirkung auf das Anfrageergebnis, da die Anfrage sich auf Daten mit Zeitbezug (Abfahrtszeit) bezog. Bei anderen Anfragen kann der Zeitpunkt der Anfrage dagegen völlig uninteressant sein, da er keinen Einfluß auf das Ergebnis hätte.

4.3.2 Zeitliche Dauer der Anfrage

Die vom Zeitpunkt der Anfrage bis zur Übermittlung der Ergebnisse vergehende Zeit ist als Dauer der Anfrage zu sehen. Betrachtet man Daten die keinen Zeitbezug haben, so fällt ein längere Antwortzeit des Datenbanksystems meist nicht ins Gewicht. Schließlich ist es dem

Forstarbeiter egal, ob seine Anfrage nach den Baumarten 5 Sekunden oder 2 Minuten dauert. Er bekommt in beiden Fällen das gleiche Ergebnis.
Bei zeitabhängigen Daten hingegen, ist die Dauer einer Anfrage wichtig. Stellt man zum Beispiel die oben genannte Frage nach dem Zug von Rostock nach Berlin, so kann bei einer langen Antwortzeit, die z.B. durch Netzbelastung oder Verbindungsausfall zustande kommt, das Ergebnis mit einer schon vergangenen Abfahrtzeit geliefert werden.
Ein besseres Beispiel für stark zeitdynamische Daten sind die Börsenkurse. Diese werden mitunter jede Minute, oder häufiger aktualisiert, so daß eine langsame Antwortzeit zu einem veralteten Ergebnis führen würde, mit dem der Nutzer nichts anfangen kann.
Man stelle sich einen Börsenmakler vor, der im Zug sitzt und sich per Funktelefon die Börsenkurse holen will, um Entscheidungen über Kauf oder Verkauf zu treffen. Während die Anfrage ausgeführt wird, gerät er in ein Funkloch und erst nach einigen Minuten steht die Verbindung wieder. Er entscheidet sich nun auf Grund der erhaltenden Ergebnisse zum Kauf, hat aber ohne es zu wissen mit veralteten Daten gearbeitet.

Damit so etwas nicht passiert, bietet sich das Zeitstempelverfahren aus der Transaktionsverarbeitung an. Jedes Ergebnis bekommt dabei noch einen Zusatz, in dem vermerkt ist, ab welchem Zeitpunkt es ungültig ist.
In unserem Zugbeispiel wäre als Zeitstempel die Abfahrzeit des Zuges möglich. Nutzbringend ist dies aber nicht, da ein Reisender nichts davon hat, wenn er einen Zug als Ergebnis geliefert bekommt, der in 2 Minuten fährt, er sich aber noch 2 Kilometer entfernt vom Bahnhof befindet.
An dieser Stelle braucht man die Verbindung zum Ortskontext und Nutzerkontext. Mit der Einbeziehung der Entfernungsdaten aus dem Ortskontext und den Nutzerdaten über Fortbewegungsart und Geschwindigkeit aus dem Nutzerkontext kann man das Verfallsdatum so berechnen, das für den Nutzer ein brauchbares Ergebnis geliefert wird.

4.3.3 Zusätzliche Eigenschaften

Bei zeitrelevanten Daten ist es wichtig, daß zwischen den einzelnen Komponenten des mobilen Systems eine Synchronisation der Systemzeit erfolgt. Wenn auf dem mobilen Gerät die Zeit nicht mit der vom benutzten Server übereinstimmt, kann es Probleme bei der Auswertung gelieferter Ergebnisse geben. Die Nutzerapplikation kann mit einer falschen Systemzeit arbeiten

und interpretiert dadurch das Verfallsdatum von Ergebnissen falsch. Zur Vermeidung solcher Fehler muß das mobile Gerät seine Systemzeit auf die des Servers stellen.
Ebenfalls in diesem Zusammenhang sind eventuell vorhandene Zeitzonenunterschiede zwischen Nutzergerät und Server zu betrachten.

4.4 Struktur des Zeitkontextes

4.4.1 Definition des Zeitkontextes

Nach der Betrachtung einiger Zeitangaben in mobilen Systemen, wird nun eine mögliche Definition des Zeitkontextes gegeben.
Im Gegensatz zum Ortskontext bietet der Zeitkontext weniger Funktionalität. Das Hauptmerkmal des Zeitkontextes ist der Anfragezeitpunkt des Nutzer.

Der Zeitkontext wird nun durch Beschreibung seiner Funktionen und Aufgaben definiert.

1) Bereitstellung des Anfragezeitpunktes auf Basis der systemweit synchronisierten Zeit.

2) Möglichkeiten zur Ermittlung der Zeitabhängigkeit von Daten, zur Verwendung in der Anfragebearbeitung und Aggregation werden bereitgestellt.

3) Funktionalität zur zeitabhängigen Veränderung der Anfrage im Anfragebearbeitungsprozeß und zur Beeinflussung der Ergebnisdaten im Aggregationsprozeß.

4) Bereitstellung von Mechanismen zur Vergabe von Gültigkeitsintervallen bzw. Verfallsdaten für Anfrageergebnisse.

Diese Funktionen können als Grundlage für die Umsetzung des Zeitkontextes für ein konkretes Anwendungsszenario genutzt werden.

Bezüglich der Klassifikationen ist der Zeitkontext genauso wie der Ortskontext einzuordnen.
Er wird vor und während der Systemnutzung einbezogen und ist dynamisch, was durch die Eigenschaft der Zeit gegeben ist.

4.4.2 Modell des Zeitkontextes

Zur Verwendung in einem Kontextserver wird der Zeitkontext ebenso wie der Ortskontext als IDL Interface definiert. Das Interface umfaßt die Basisfunktionalität, die in der vorhergehenden Funktionenbeschreibung gemacht wurde.

```
interface zeitkontext {
      void aktualisiere_Zeit();
      boolean zeitabhängiges_Attribut( in string attribut);
      boolean zeitdynamisches_Attribut( in string attribut);
      time hole_Anfragezeit();
      time hole_Gueltigkeitsdatum( in string zeitattribut);
      time hole_Intervallgrenze_von ( in string zeitattribut);
      time hole_Intervallgrenze_bis ( in string zeitattribut);
}
```

Die erste Funktion `aktualisiere_Zeit()` wird vom Konstruktor des Zeitkontextes selbst aufgerufen, sobald ein neues Zeitkontextobjekt erzeugt wird. Die Implementation bezieht dabei die Zeit von der serverseitigen Systemzeit und speichert sie in einer dafür vorgesehene Membervariable.

Die folgenden Funktionen dienen zur Ermittlung der Zeitabhängigkeit von Attributen.
Die Verwendung des Zeitkontextes setzt voraus, daß zeitbezogene Daten bekannt sind. Die zusätzlichen Informationen könnten durch Erweiterung der Metadaten im Datenmodell mit untergebracht werden. Weil diese Methode aber Änderungen an der Metadatensyntax erfordert, ist sie recht aufwendig zu realisieren.
Eine zweite Möglichkeit besteht in der Erweiterung des jeweiligen Datenmodells um neue Tabellen (INGRES) oder neue Klassen (O2) , in denen die nötigen Informationen bereitgestellt werden. Die neuen Tabellen heißen dann `Zeitabhängig` und `Zeitdynamisch`.
Die Ausgangstabelle für Züge kann folgendes Aussehen haben.

`Tabelle Zuege`

Zugnr	Abfahrtsort	Zielort	Abfahrtszeit	Ankunftszeit	Bemerkungen
0815	Rostock	Berlin	12:31:00	15:04:00	IR
0747	Rostock	Berlin	14:31:00	17:58:00	D
...	...	...	...	...	...

Für eine Zug müßten also die Attribute Abfahrt- und Ankunftszeit als zeitabhängig erfaßt.

Tabelle Zeitabhaengig

Attributname
Abfahrtszeit
Ankunftszeit
...

Die Anfragebearbeitung kann nun mit der Funktion `zeitabhängiges_Attribut(in string attribut)` für ein Anfrageattribut prüfen, ob es sich um eine zeitabhängiges Attribut handelt, indem in der Tabelle nach dem Attribut gesucht wird.

Für Börsenkurse sieht die Ausgangstabelle wie folgt aus:

Tabelle Boersenkurse

Name	Wert	Min	Max
Kurs 1	67,50	63,30	68,30
Kurs 2	45,00	34,00	51,00
...	...	...	...

Bei zeitdynamischen Daten enthält die Tabelle neben dem Attributnamen, noch die Angabe, in welchen Abständen der Kurs aktualisiert wird und eine Bezugszeit, von der aus die Aktualisierung gerechnet wird.

Tabelle Zeitdynamisch

Attributname	Bezugszeit	Aktualisierung
Kurs 1	00:00:00	00:01:00
Kurs 2	02:30:00	00:25:00
...	...	...

Das erste Tupel der Tabelle gibt an, daß Kurs 1 jede Minute aktualisiert wird. Ausgehend von der Bezugszeit heißt dies, daß zu jeder vollen Minute eine Aktualisierung stattfindet. Kurs 2 wird bezogen auf 2:30 Uhr alle 25 Minuten, also um 2:55, 3:20 usw. aktualisiert.
Die Funktion `zeitdynamisches_Attribut(in string attribut)` prüft in der `Tabelle Zeitdynamisch`, ob das übergebene Attribut enthalten ist.

Nachdem bekannt ist, welche Attribute zeitabhängig sind, werden Methoden benötigt, die z.B. eine Veränderung der Anfrage ermöglichen.

Die einfachste Funktion `time hole_Anfragezeit()`, liefert den Zeitpunkt der Anfrage (entspricht dem Zeitpunkt der Erzeugung des Zeitkontextobjektes). Diese Angabe kann zur Erweiterung der Anfrage benutzt werden. Ein Beispiel dafür wurde bei der Beschreibung der Eigenschaft "Zeitpunkt der Anfrage" gegeben.

Zusätzlich zum Anfragezeitpunkt, ist es aber manchmal interessant auch Zeitintervalle zu betrachten.
Ein solches Intervall besitzt eine untere und eine obere Grenze.

Mit den beiden Funktionen `hole_Intervallgrenze_von(...)` und `hole_Intervallgrenze_bis (...)` liefert der Zeitkontext die Intervallgrenzen für ein zeitabhängiges Attribut. Diese können dann zur Anfrageerweiterung oder Ergebnisbeeinflussung bei der Aggregation benutzt werden. Die Funktion `hole_Intervallgrenze_von(...)` könnte als Rückgabewert z.B. den Anfragezeitpunkt liefern, es kann aber auch eine Einstellung aus dem Nutzerkontext abgefragt werden. Da die Ermittlung der Intervallgrenzen attributabhängig ist, müssen bei manchen Attributen neben den Zeitangaben auch Eigenschaften aus dem Nutzerkontext betrachtet werden. Diese Einstellungen werden für das jeweilige Anwendungsszenario vorgenommen und beinhalten z.B. daß für das Attribut "Abfahrtszeiten" ein 2 Stunden Intervall eingestellt ist oder für andere Attribute eine "Standard Zeitangabe" vorgegeben wird, die bei einer Anfrage benutzt werden soll.

Die Funktion `hole_Gueltigkeitsdatum(...)` berechnet das Gültigkeitsdatum für ein übergebenes zeitabhängiges bzw. zeitdynamisches Attribut. Die Berechnung erfolgt in Abhängigkeit von dem jeweils betrachteten Attribut und abermals von Einstellungen im Nutzerkontext.

Die Gültigkeit eines Anfrageergebnisses kann in zwei Arten unterschieden werden.

- für den Nutzer ab einem Zeitpunkt nicht mehr brauchbare Informationen bezüglich der Semantik der Anfrage
- ab einem Zeitpunkt veraltete Informationen, durch geänderte Daten im Datenbestand.

Für Attribute, die durch Intervallgrenzen in der Anfrage eingeschränkt werden, ist eine Vergabe von Gültigkeitsdaten nur für diejenigen möglich, deren Intervallgrenzen bezüglich dem Anfragedatum in der Zukunft liegen. In diesem Fall kann das Gültigkeitsdatum z.B. gleichsetzen mit dem Zeitpunkt der oberen Intervallgrenze.

Die zweite Art von Gültigkeit kann bei den zeitdynamischen Attributen verwendet werden. Unter Verwendung der weiter oben definierten `Tabelle Zeitdynamisch` können die Änderungszeitpunkte der Werte von solchen Attributen berechnet werden und damit als Gültigkeitsdatum gesetzt werden.

Neben den hier beschriebenen Funktionen gehört auch die Wertung des Benutzerprofils mit in den Zeitkontext. Wie im Ortskontext beschrieben, wirkt die Benutzerprofilwertung nicht auf den eigentlichen Anfragebeeinflussungs- bzw. Aggregationsprozeß , sondern wird vom System verwendet um das Laufzeitverhalten bei Ortswechseln durch Zwischenspeicherung zu verbessern.
Das Aussehen solcher Bewegungsprofiltabelle ist in [Boj97] beschrieben und wird hier nicht weiter ausgeführt.

4.4.3 Zeitkontext für die Walddatenerfassung

Im unserem Walddatenbeispiel gibt es keine zeitbezogenen Informationen. Die einzigen im Modell enthaltenen Zeitangaben sind Jahresangaben z.B. für das Aufforstungsjahr und das Hiebsjahr.
Alle anderen Daten in diesem Anwendungsszenario können als statisch betrachtet werden. Sie ändern sich nur dann, wenn eine Aufforstung oder eine Abholzung durchgeführt wird, oder ein Waldarbeiter im Wald vor Ort den Datenbestand um eventuelle Schäden aktualisiert.
Der Vorgang der Bestandsaktualisierung findet allerdings nur ungefähr alle 10 Jahre statt, und erfordert keine Betrachtung von Gültigkeitszeiträumen o.ä. Zeitdaten.
Der Einsatz des Zeitkontextes ist also für das Walddatenbeispiel nicht nutzbringend. Einzig die Auswertung von Benutzerprofilen bietet sich in diesem Beispiel an.

4.4.4 Zeitkontext für den Möbelvertrieb

Im Möbelvertrieb gibt es zeitabhängige Daten, für Verträge zwischen der Vertriebsfirma und den Lieferanten. Da ein Vertrag immer für einen Zeitraum ausgehandelt wird, kann man für Anfragen nach diesen Daten einen im voraus berechneten Gültigkeitszeitraum benutzen. Mit dem Gültigkeitszeitraum kann dann problemlos die Aktualität der Daten überprüft werden. Normalerweise werden solche Verträge aber eher relativ langfristig abgeschlossen, so daß die Dynamik der Daten nicht sehr groß ist.
Bei kurzfristigen Preisänderungen für bestimme Möbel könnte der Zeitkontext benutzt werden. Da solche Preisänderungen aber nicht zu vorhersehbaren Zeitpunkten geschehen, müßten Zeitintervalle (z.B. 5 Minuten) festgelegt werden, für die ein Zeitkontextobjekt als gültig angesehen wird. Nach Ablauf solch eines Intervalls muß der Zeitkontext einen Abgleich mit der Datenbank machen und auf Änderungen durch Verwerfen des Ergebnisses oder Nachfrage beim Nutzer reagieren.

Betrachtet man aber die üblichen Vertragslaufzeiten in diesem Beispiel, so bringt der Einsatz des Zeitkontextes auch im Möbelvertrieb keinen erkennbaren Nutzen.

5 Wirkung des Orts-/ Zeitkontextes auf Datenbanksysteme

5.1 Datenbanksystemfunktionalität

Zur Benutzung von Kontexten in Datenbanksystemen muß die Architektur der DBS dahingehend erweitert werden, daß sie die Kontextdaten verwalten und verarbeiten kann.

Dazu gehört die Integration des Kontextservers, der letztendlich für die Verwaltung und Speicherung der Kontextobjekte zuständig ist.

Der Einsatz von Kontexten in einer verteilten Datenbankumgebung erfordert ein Kontextmodell, das von jedem eingesetzten Datenbanksystem verstanden wird. Da sich die eingesetzten Datenbanksystem aber hinsichtlich ihrer Modellparadigmen (objektorientiert, relational) unterscheiden, muß eine Transformation des Kontextmodells in das jeweilige Datenbankmodell erfolgen.

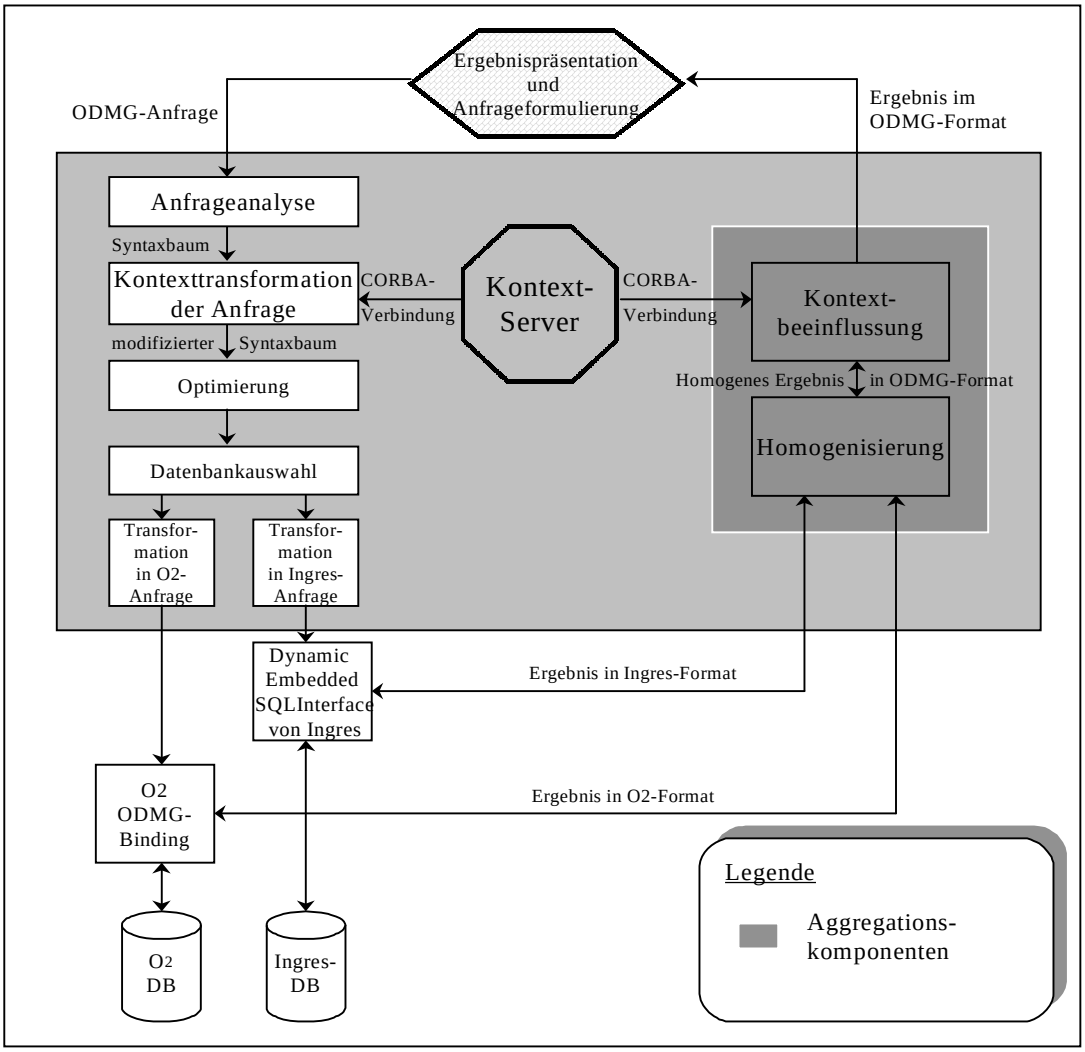

Abbildung 3 : Architektur der Anfragebearbeitung für die Mobile Walddatenerfassung

Die Architektur zur Einbindung von Kontexten in den Prozeß der Anfragebearbeitung ist in der Abbildung 3, die aus [Wen97] entnommen wurde, dargestellt.

Wie in der Abbildung 3 zu sehen ist, übernimmt der Kontextserver die Bereitstellung der Kontexte für die Komponenten des Datenbanksystems. Der Einfluß der Kontextbetrachtung ist in der Abbildung durch die Verbindungen des Kontextservers zur Kontexttransformation der Anfrage und zur Kontextbeeinflussung des Ergebnisses dargestellt. In diesen beiden Teilprozessen der Anfragebearbeitung wird also die Kontextbeeinflussung durchgeführt.

Im folgenden Abschnitt wird nun der Aufbau des Kontextservers und die Einordnung des Ortskontextes, in diesen, beschrieben.

5.1.1 Architektur der Kontextumgebung - Der Kontextserver

Um die in den vorangegangenen Kapiteln erläuterten Ortsangaben für Datenbanksysteme nutzbar zu machen, muß eine geeignete Form der Darstellung für den Ortskontext gefunden werden.
Diese Realisierung erfolgt in einem Kontextserver, der die verschiedenen Kontexte verwaltet und den einzelnen Prozessen die Möglichkeit gibt, mit diesen Kontexten zu arbeiten.

Die Kontexte liegen als Objekte auf dem Kontextserver und können über CORBA benutzt werden. Der CORBA Standard wurde in Absprache mit den Autoren von [Wen97] und [Boj97] als Kommunikationsplattform aus mehreren Gründen vorgeschlagen. Der Hauptgrund ist, daß er plattformübergreifend in Form von verschieden CORBA Implementierungen zur Verfügung steht (z.B. VisiBroker, Orbix, DEC ObjectBroker). Außerdem bietet er große Funktionalität für die Verwaltung und das Arbeiten mit verteilten Objekten, wobei der Programmieraufwand für die Kommunikation bei Client/Server Anwendung sehr gering ist, da dies alls in CORBA integriert ist.
Da sich die CORBA Implementierungen an den IIOP Standard halten, ist die Kommunikation zwischen verschieden ORB´s möglich.

Für die Client Anwendung, die auf ein Kontextobjekt zugreifen will, ist es durch CORBA völlig transparent, wo sich das Kontextobjekt befindet und auf welchen System es implementiert ist.

Einen weiteren Vorteil bietet CORBA bei der Implementierung der Kontextobjekte.
Die Objekte werden in der IDL von CORBA definiert. Danach übernimmt CORBA die Generierung der konkreten Implementationsklassen aus der IDL Definition.
Weil die CORBA Implentierungen für mehrere Programmiersprachen erhältlich sind, können Client und Server sogar in verschiedenen Sprachen implementiert werden. Die Generierung macht dann aus der IDL Definition die Umsetzung der Klassen in die jeweilige Programmiersprache.

Die folgende Abbildung soll die Funktionsweise der Client / Server Architektur mit CORBA für ein Ortskontextobjekt verdeutlichen.

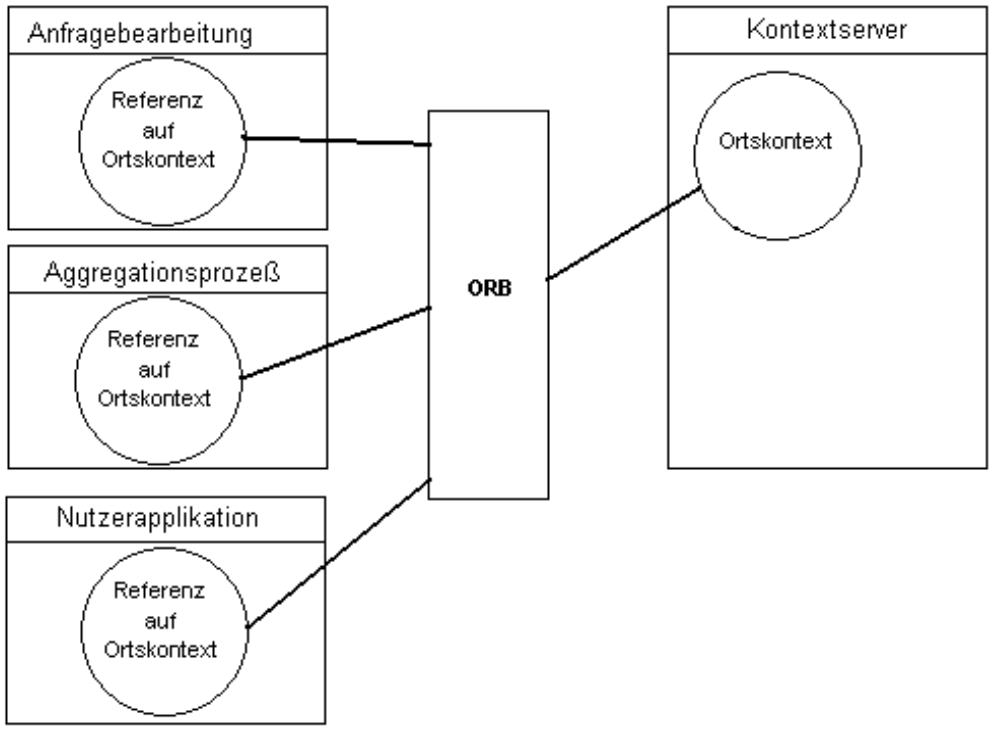

Abbildung 4 : Zugriff über CORBA auf den Ortskontext

An dieser Stelle will ich kurz das Prinzip des Objektzugriffs der Clientanwendungen erläutern. Der Server stellt über einen sogenannten Agenten (vergleichbar mit einem Dämonenprozeß) die Kontextobjekte zur Verfügung.

Der Client fordert beim ORB eine OID für ein Objekt, das unter dem Namen „Ortskontext“ beim Agenten registriert wurde. Ist ein solches Objekt vorhanden, bekommt der Client einen Zeiger auf das Serverobjekt geliefert. Man bezeichnet den Zeiger auf der Clientseite auch als Proxyobjekt. Mit dem so gebundenen Objekt kann der Client alle durch dieses Objekt zur Verfügung gestellten Attribute und Methoden nutzen, als läge es lokal auf seinem Rechner.
Der ORB im Hintergrund übernimmt das Delegieren der Aufrufe des Clients an das Objekt zur Serverseite. Alle Methodenaufrufen, die der Client aufruft, werden also in Wirklichkeit auf dem Serverobjekt ausgeführt und der Client erhält dann nur die Ergebnisse mitgeteilt.

Das Ortskontextobjekt wurde an dieser Stelle nur zur Veranschaulichung gewählt. In der Realität kann der Ortskontext nicht alleinstehend betrachtet werden, weil er an vielen Stellen das Zusammenspiel der einzelnen in [Wat96] betrachteten Kontexte benötigt.

Aus diesem Grund wird in Zusammenarbeit mit den Autoren von [Wen97] und [Boj97] eine Klasse „Personenkontext“ eingeführt, die als Komponentenobjekte die verschiedenen Kontexte enthält.
Der Personenkontext enthält alle Kontextdaten für einen Nutzer und wird einem Nutzer zugeordnet. Diese Zuordnung geschieht bei der Anmeldung an das System und endet mit der Abmeldung aus dem System. Die Persistenzfragen werden durch das zugrundeliegende Datenbanksystem geregelt. Es müssen also keine expliziten Mechanismen gefunden werden um veränderte Kontextdaten persistent zu machen, damit sie bei der nächsten Sitzung des Nutzers wiederverwendet werden können.

Der Personenkontext enthält im Einzelnen die Kontexte : ORT, ZEIT, RESSOURCE(Information) und NUTZERPROFIL.

Die IDL Deklaration des Personenkontextes sieht folgendermaßen aus:

```
interface Personenenkontext
{       Ortskontext  Hole_Ort()
        Zeitkontext Hole_Zeit()
        Ressource   Hole_Ressource()
        Profil      Hole_Profil()
        Aendere_Ort( parameterliste)
        Aendere_Zeit( parameterliste)
        Aendere_Ressource( parameterliste)
        Aendere_Profil( parameterliste)
}
```

Über die "Hole" Funktionen kann auf die jeweiligen Kontextobjekte für den Nutzer zugegriffen werden. Für den Orts- /Zeitkontext bedeutet das, daß mit `Hole_Ort()` bzw. `Hole_Zeit()` jeweils eine Referenz auf den aktuellen Ortskontext bzw. auf den aktuellen Zeitkontext geliefert wird.
Mit diesen Objektreferenzen ist dann eine Zugriff auf die Methoden des Orts- / Zeitkontextes möglich.

Die Erläuterung des Informationskontextes und eine Spezifikation des Nutzerprofils erfolgen in [Boj97].

5.2 Wirkung des Ort-/Zeitkontextes auf Anfragebearbeitung

Bei der Betrachtung der Architektur der Anfragebearbeitung wurden die beiden Einflußbereiche der Kontexte festgestellt. Es sind die Kontexttransformation der Anfrage und die Kontextbeeinflussung der Ergebnisse. Damit diese beiden Komponenten mit Kontexten arbeiten können, müssen sie die vom Kontextserver bereitgestellten Kontextobjekte nutzen. Die Kommunikation mit dem Kontextserver erfolgt über die im vorherigen Abschnitt beschriebene CORBA - Umgebung.

Für die Nutzung des Orts- / Zeitkontext heißt das, daß sich eine Bearbeitungskomponente erst einen Personenkontext vom Kontextserver binden muß. Über diesen kann dann auf den Orts- und Zeitkontext zugegriffen werden.

Für die folgende Betrachtung der Kontextbeeinflussung bei der Transformation der Anfrage und der Beeinflussung der Ergebnisse werden einige wiederkehrende Schritte vorausgesetzt.
Zuerst bindet sich die Bearbeitungskomponente mit

```
Personenkontext *pk = Personenkontext::_bind();
```

eine Referenz auf einen Personenkontextobjekt. Das Binden wird durch einen CORBA Namensdienst erledigt.
Um den dazugehörigen Orts- / Zeitkontext zu erhalten, erfolgt ein Aufruf der Form

```
Ortskontext *ok = pk->Hole_Ort();
Zeitkontext *zk = pk->Hole_Zeit();
```

Jetzt kann die Komponenten auf die Methoden der Kontexte zugreifen.
Der Zugriff auf die Attribute des Nutzerortes geschieht mit der folgenden Anweisung.

```
Adresse nutzer = ok->liefere_Adresse();
```

In den folgenden Abschnitten wird zur Verkürzung der Schreibweise nur noch auf die Struktur `nutzer` zugegriffen.

5.2.1 Kontexttransformation der Anfrage

Die Anfrage des Nutzers wird dem Datenbanksystem in Form einer ODMG Anfrage übergeben.
Diese ODMG Anfrage wurde von der mobilen Anwendung des Nutzers konstruiert und dann über das mobile Netz zum Datenbanksystem transportiert.

Unmittelbar nach dem Eintreffen einer Anfrage im System, wird vom Datenbanksystem die Anfrageanalyse ausgeführt. Diese Analyse besteht aus einer herkömmlichen Syntax- und Semantikanalyse. In diesem Schritt werden Kontexte noch nicht betrachtet. Bei der Syntaxanalyse wird die Anfrage in einen entsprechenden Syntaxbaum transformiert.
Dieser Syntaxbaum ist dann der Ausgangspunkt für die folgende Semantikanalyse.

Nach der abgeschlossenen Standardanalyse erfolgt nun die Einbeziehung der Kontexte zur kontextabhängigen Bearbeitung der Anfrage.

Um für eine Anfrage zu entscheiden, ob sie kontextbezogen ist oder nicht, muß eine erweiterte Anfrageanalyse vorgenommen werden.
An dieser Stelle erfolgt die erste Prüfung der Anfrage hinsichtlich ihrer Kontextabhängigkeit. Das wird durch eine syntaktische Untersuchung der Anfrage auf vorhandene kontextabhängige Attribute erreicht.
Damit die erweiterte Analyse solche Attribute erkennen kann, muß sie Zugriff auf die Attributbeschreibung in den Metadaten und auf die speziellen Hierarchiedaten der Kontexte haben. Für den Ortskontext ist dies unter anderem die Klasse Hierarchieebenen, die bei der Einführung des Ortskontextmodells vorgestellt wurde. Durch die Analyse der Anfrage wird ihre Kontextabhängigkeit bestimmt und mit dem Ergebnis läßt sich dann die Transformation der Anfrage durchführen.

Bezüglich des Ortskontextes kann man die Anfragen in mehrere Gruppen einteilen.

Anhand einiger Beispielanfragen, soll die Funktionsweise der semantischen Analyse bezüglich des Ortskontextes vorgestellt werden.

Gruppe 1: Die Ausgangsanfrage beinhaltet keinerlei Ortsangaben

Beispiel.:

```
select Baumart, Produktionsziel from x in Zustandszeile_fest, y in Waldnut-
zungsplanung
where x.schluessel=y.schluessel;
```

So gestellt ist die Anfrage syntaktisch und semantisch richtig, aber trotzdem ist das Ergebnis dieser Anfrage für den Nutzer höchstwahrscheinlich unbrauchbar, da er alle in der Datenbank vorhandenen Baumarten mit dem jeweils dazugehörigen Produktionsziel erhalten würde.

Zur Verbesserung trifft das System die Annahme, daß der Nutzer eine ortsbezogene Anfrage gestellt hat und bezieht den aktuellen Ort des Nutzers in die Anfrage mit ein.
Die Ermittlung des Nutzerortes in Form der im Datenmodell vorhandenen Attribute erfolgt durch die Benutzung der Funktionalität des Ortskontextes, der zu Beginn des Kontexttransformationsprozesses vom Kontextserver geholt wurde und somit für die Transformation der Anfrage zur Verfügung steht.

Als Ortsangabe wird dabei die gesamte Struktur des Nutzerortes genommen. Somit wird die Anfrage auf die kleinste mögliche Ortsangabe eingeschränkt.
Die Erweiterung der **from** und **where** Bedingungen werden wenn nötig und möglich ebenfalls vorgenommen.

nach der Erweiterung sieht die Anfrage so aus:

```
select Baumart, Produktionsziel from x in Zustandszeile_fest, y in Waldnut-
zungsplanung, z in Adressen
where x.schluessel=y.schluessel and x.schluessel=z.schluessel and
      z.revier=nutzer.revier and ... and z.teilflaeche=nutzer.teilfäche;
```

Das Ergebnis der modifizierten Anfrage sind jetzt alle Baumarten und das dazugehörige Produktionsziel bezüglich der Teilfläche in der sich der Nutzer zum Zeitpunkt der Anfrage befunden hat. Das Ergebnis allein betrachtet hat keinerlei Ortsbezug in seinen Attributen.

Wirkung des Kontextes :

➔ **Erweiterung der Anfrage mit dem bereitgestellten Nutzerort**

Gruppe 2: Ein Hierarchieattribut in der select Anweisung

Beispiel.:

```
select Einrichtungsjahr, Teilfäche  from x in Zustandsteilfläche, y in
Adresse
where x.schluessel=y.schluessel;
```

In diesem Fall hat der select-Teil der Anfrage einen Ortsbezug durch das Hierarchieattribut Teilfläche.
Der Nutzer möchte alle Teilflächen und deren Einrichtungsjahr. Ohne Kontexteinbeziehung liefert die Anfrage alle Paare von Teilflächen und Einrichtungsjahren aus der Datenbank.
Eine Erweiterung der Anfrage mit der Teilfläche des Nutzers ist in diesem Fall wenig sinnvoll, weil dann nur Daten dieser Teilfläche als Ergebnis geliefert werden. Vielmehr kämen die in der Hierarchieebene höher gelegenen Attribute zur Erweiterung in Frage.
Die veränderte Anfrage sieht wie folgt aus.:

```
select Einrichtungsjahr, Teilfäche  from x in Zustandsteilfläche, y in
Adresse
where x.schluessel=y.schluessel and
      z.revier=nutzer.revier and
      z.abteilung=nutzer.abteilung;
```

Das Ergebnis der Anfrage sind dann alle gewünschten Daten aus der Abteilung in der sich der Nutzer befindet. Welche Hierarchieebene zur Erweiterung benutzt wird, hängt von den Einstellungen ab, die im Nutzerkontext gespeichert sind.

Wirkung des Kontextes :

➔ **Ermitteln der nächsthöheren Hierarchieebene**
Erweiterung der Anfrage um Nutzerort, aber nur bis zur ermittelten Hierarchieebene

Fall 3: Anfrage hat Hierarchieattribute aber festen Ort in der where Klausel

Beispiel:

```
select Einrichtungsjahr, Teilfäche  from x in Zustandsteilfläche, y in
Adresse
```

```
where x.schluessel=y.schluessel and
      z.revier="Born" and
      z.abteilung="183";
```

In diesem Fall wird die Anfrage durch die Analyse nicht verändert, da der Nutzer einen Ort spezifiziert hat. Das Ergebnis der Anfrage sind ähnlich wie zuvor, alle Teilflächen der Abteilung 183 mit ihrem Einrichtungsjahr.

5.2.2 Aggregation (Kontextbeeinflussung) der Ergebnisse

Bevor die Ergebnisse einer Anfrage dem Nutzer gesendet werden bzw. entsprechend für ihn aufbereitet werden können, durchlaufen sie eine Reihe von Prozessen. Die Gesamtheit dieser Prozesse ist die Aggregation.
Die Aggregation kann man in die Homogenisierung und die Kontextbeeinflussung einteilen. Die Homogenisierung ist dabei für die Transformation der Teilergebnisse in ein globales ODMG - konformes Schema verantwortlich. Dieser Schritt ist ziemlich komplex, da die Teilergebnisse von den verschiedenartigen Komponenten des mobilen Systems kommen und unterschiedlich strukturiert sind.
Im Waldbeispiel sind zum Beispiel die Ergebnistupel der Ingres Teilanfrage mit den Ergebnisobjekten der O2 Anfrage so zu kombinieren, das zum Schluß ein Ergebnis im ODMG Format vorliegt.
Die zweite Aufgabe der Aggregation ist die Kontextbeeinflussung auf der Ergebnisebene. Dabei wird die Anwendbarkeit der verfügbaren Kontexte für ein Ergebnis untersucht. Sind ein oder mehrere Kontexte auf ein Anfrageergebnis anwendbar, so führt die Aggregation die Beeinflussung durch.
Das Ergebnis der Aggregation ist ein Anfrageergebnis, daß durch alle anwendbaren Kontexteigenschaften beeinflußt (z.B. Attributpriorisierung, Ortsangaben) wurde.

Eine wesentlicher Punkt des Aggregationsvorgangs ist die Ergebnisbeeinflussung. In [Wen97] werden mehrere Arten von Beeinflussungsmöglichkeiten vorgestellt. Die folgende Abbildung zeigt die Struktur der Beeinflussungsarten.

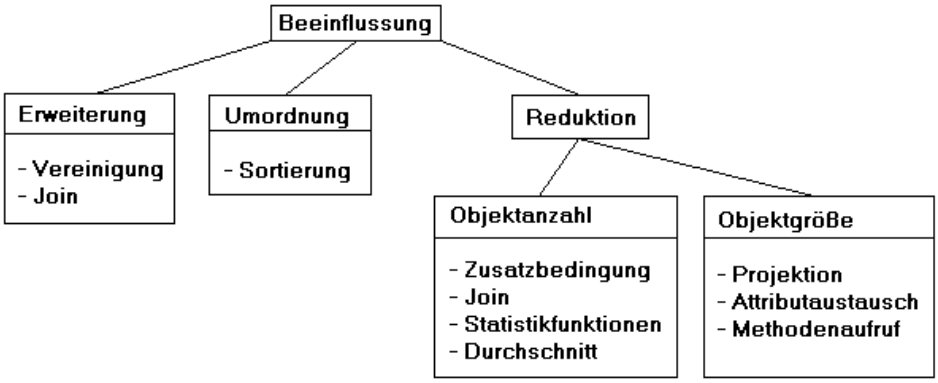

Abbildung 5 : Überblick der Ergebnisbeeinflussungsarten aus [Wen97]

Im folgenden soll der Orts- /Zeitkontext bezüglich seiner Wirkung auf die einzelnen Möglichkeiten der Ergebnisbeeinflussung untersucht werden.

5.2.2.1 Reduktion der Ergebnismenge durch Zusatzbedingung

Bei dieser Art der Reduktion wird die Ergebnismenge durch Hinzunahme von zusätzlichen Auswahlkriterien verkleinert, indem nur die Objekte in die Zielmenge übernommen werden, die die Bedingungen erfüllen. Die Reduktion kann natürlich nur mit Bedingungen vorgenommen werden, die semantisch zu den Daten des Zielergebnisses passen.
Der Orts- /Zeitkontext kann in dieser Kategorie der Reduktion angewendet werden, wenn die zu bearbeitende Daten Orts- oder Zeitbezug haben. Im Walddatenbeispiel kann man sich eine Menge von Teilflächen vorstellen, die als Ergebnis einer Anfrage geliefert wurde. Zur Reduktion des bisherigen Ergebnisses kann die zusätzliche Bedingung formuliert werden, daß nur die Teilflächen, die in der aktuellen durch den Ortskontext bestimmten Abteilung liegen, in die neue Ergebnismenge übernommen werden.
Die zusätzliche Klausel in der select Anweisung sieht dann so aus:

```
select X from X in Ergebnismenge_alt where X.Abteilung = nutzer.Abteilung;
```

X repräsentiert dabei alle Teilflächen der Ergebnismenge und mit nutzer.Abteilung wird auf die aktuelle Abteilung des Ortskontextes zugegriffen. In diesem Bereich der Aggregation kann auch der Zeitkontext angewendet werden.
Durch eine Zusatzbedingung vom Typ Zeit, kann das bisherige Anfrageergebnis reduziert werden. Das bisherige Ergebnis muß aber Zeitangaben enthalten, da die Zusatzbedingung sonst nicht angewendet werden kann. Zur Überprüfung der Zeitabhängigkeit, kann man die Attribute aus dem Ergebnistyp mit den vom Zeitkontext bereitgestellten Funktionen untersuchen.

5.2.2.2 Reduktion der Ergebnismenge durch Verbundoperation (Join)

Die Reduktion durch Verbundoperation hat als Ausgangspunkt zwei Objektmengen, von denen eine das ursprüngliche Ergebnis darstellt und die andere eine bestimmte Bedingung erfüllt.
Die Verbundoperation hat zur Folge, daß in die neue Ergebnismenge tatsächlich nur die Objekte übernommen werden, die in beiden Mengen vorhanden sind. Das Endergebnis enthält dann alle die Objekte, die neben der Anfrage auch die zusätzliche Bedingung erfüllen.
Auch bei dieser Reduktionsart kann der Ortskontext einbezogen werden. Als Beispiel dient wie im vorangegangenen Fall wieder eine Ausgangsmenge von Teilflächen. Die Methode `nahe_Teilflaechen()` des Ortskontextes liefert als Ergebnis eine Menge von Teilflächenadressen, die bezüglich des aktuellen Nutzerortes in der Nähe liegen.
Nach welchen Kriterien die Entscheidung über die Nähe der Teilflächen getroffen wird, hängt vom Nutzerprofil und von der Implementation der Methode ab. Einige Vorschläge zur Berechnung der Nähe wurden im Abschnitt zum Ortskontext gemacht.
Im diesem Fall wird die Verbundoperation über allen Attributen der Struktur Adresse ausgeführt.
Natürlich lassen sich Teilflächen schneller durch ihren Schlüssel ansprechen, da aber die Nähe auch für andere Hierarchieebenen definiert ist, wurde als Rückgabe die Struktur Adresse gewählt.
Sucht man z.B. alle Abteilungen in der Nähe, so kann als Ergebnis der Methode `nahe_Abteilungen()` eine Menge von Adressen geliefert werden, deren Teilflächen- und Unterabteilungsattribute leer sind. Der Verbund mit der ursprünglichen Ergebnismenge wird dann über die Attribute von Adresse, außer Unterabteilung und Teilfläche gemacht.

5.2.2.3 Reduktion der Ergebnismenge durch Statistikzusammenfassungen

In bestimmten Situationen sind nicht alle genauen Werte eines Datenbestandes interessant, sondern der Nutzer möchte eventuell nur einen Überblick über vorhandene Attributgruppen und deren Werte bekommen.
Zu diesem Zweck können die Aggregatfunktionen von SQL zur Bearbeitung von Attributgruppen eingesetzt werden.
Folgendes Beispiel wird in [Wen97] für die mögliche Anwendung gegeben.
Mit dem Teilausdruck

```
"...group x in $1k by
(baumart:x.baumart,\saatgutherkunft:x.saatgutherkunft)
with (jaerlicher_zuwachs:\ avg(select x_1.jaehrlicher_zuwachs from x_1 in
partition))...",
```

wird der zuvor auf die drei Attribute Baumart, Saatgutherkunft und Zuwachs reduzierte Datenbestand in Gruppen gleicher Baumart und gleicher Saatgutherkunft eingeteilt.
Für jede dieser neu entstandenen Gruppen wird dann über allen Gruppenmitgliedern die Durchschnittsbildung des Zuwachswertes durchgeführt.
Abschließend kann der Nutzer dann übersichtlich für die einzelnen Baumarten erkennen, bei welcher Saatgutherkunft der beste Zuwachs zu erfolgt ist.
Für die Aggregationen der Ergebnisse durch Statistikzusammenfassungen kann der Ortskontext Gruppierungsattribute bereitstellen. Ein einfacher Anwendungsfall wäre z.B. die Gruppierung der Ergebnisse nach Abteilungen.

5.2.2.4 Reduktion auf Attributebene

Bei der Ergebnisbeeinflussung auf der Attributebene, wird im Gegensatz zur vorherigen Methode nicht die Menge der Ergebnisobjekte reduziert, sondern die Größe der Objekte bzw. die Reihenfolge der Objektattribute.
Alle Arten der Reduzierung auf Attributebene müssen Zugriff auf Informationen aus den Metadaten des Datenbestandes haben. Auf diesem Gebiet hat der Orts- /Zeitkontext keine Wirkung auf die Aggregation. Die wichtigste Aufgabe kommt in diesem Zusammenhang dem Informationskontext [Boj97] zu, da er Methoden zur Bearbeitung und Wichtung der Attribute bereitstellt.

Die drei in der Abbildung der Beeinflussungsarten enthaltenen Möglichkeiten zur Attributänderung werden nun kurz beschreiben.
Die erste Art ist die Projektion auf eine Untermenge der Ausgangsattributmenge. Dabei werden alle nicht benötigten Attribute ausgeblendet und somit der Typ des Ergebnisses geändert. Die Entscheidung der Attributpriorisierung kann über eine Wichtungstabelle getroffen werden, in der die Attribute einen Wert zugewiesen bekommen. Je kleiner dieser Wert für ein Attribut ist, desto eher wird dieses Attribut bei der Reduktion durch Projektion aus dem Ergebnis herausfallen.
Die Reduzierung der Größe von Attributen ist eine weitere Möglichkeit der Beeinflussung.

Um dies für eine bestimmtes Attribut durchführen zu können, muß ein Attribut mit der gleichen Information aber geringerer Größe zur Verfügung stehen. Ein Beispiel ist das Ersetzen eines Bildattributes durch eine dazugehörige Bildbeschreibung. Eine zweite Möglichkeit ist die Reduzierung der Informationsmenge eines Attributes durch Anwendung einer vom Attributtyp bereitgestellten Methode. Denkbar wäre z.B. für ein Bildattribut die Verringerung der Bildauflösung.

5.2.2.5 Umordnung der Ergebnismenge

Die Änderung der Reihenfolge der Ergebnismenge erfolgt durch Sortieren der Objekte nach den Werten von einem oder mehreren Attributen. Auch für diesen Zweck wird wieder eine Bewertungsgrundlagen in Form eine Wichtungstabelle benötigt, nach welchen Attributen vorrangig sortiert werden soll. Diese Aufgabe fällt wieder dem Informationskontext zu. Der Ortskontext stellt zwar keine Sortiermöglichkeit zur Verfügung, zumindest aber eine Möglichkeit, Entfernungsattribute für das Ergebnis zu erzeugen. Mit der Methode `entfernung` des Ortskontextes kann die Entfernung einer Teilfläche bzw. Abteilung usw. zum aktuellen Standpunkt ermittelt werden und als Wert in das Ergebnis übernommen werden.
Durch Sortierung dieser Entfernungsspalte kann man nun das Ergebnis ortsabhängig sortieren, so daß die am dichtesten am Standort gelegenen Objekte zuerst übermittelt werden können.

5.2.3 Kontextabhängige Datendarstellung

Unter der Datendarstellung versteht man die Präsentation eines Anfrageergebnisses auf dem Gerät des Nutzers.
Die Datendarstellung ist in entscheidendem Maße vom Gerät des Nutzers abhängig. Es ist ja ein Unterschied, ob er die Daten in seinem Büro auf einem Pentium Rechner mit 15 Zoll Farbmonitor dargestellt haben möchte, oder ob er mobil mit seinem PDA mit Graustufendisplay unterwegs ist.
Damit eine situationsgerechte Darstellung erfolgen kann, muß auch die mobile Applikation die in den verschiedenen Kontexten gespeicherten Informationen berücksichtigen.
Die wichtigsten Eigenschaften werden hierbei aus dem Nutzer- und dem Ressourcenkontext genommen. Ein Nutzer läßt sich seine Ergebnisse als einfaches Balkendiagramm anzeigen, ein anderer hingegen bevorzugt die einfache Zahlendarstellung.

Die Darstellung der Ergebnisse wird aber auch durch die Funktionalität der Anwendung selbst bestimmt. Diese wird bei Systemstart unter Auswertung der Kontextinformationen festgelegt.

Die größte Rolle spielt dabei der Nutzerkontext. Zur Anwendung kommen dabei Eigenschaften wie Zugriffsrechte, Nutzervorlieben aber auch Eigenschaften des mobilen Gerätes.

Der Orts-/Zeitkontext spielt bei der Darstellung der Daten keine große Rolle. Seine Aufgaben liegen in der Beeinflussung eines Anfrageergebnisses.

5.3 Wirkung auf das Transaktionsaktionsmanagement

Transaktionen in mobilen System laufen unter erschwerten Bedingungen gegenüber normalen Transaktionen in verteilten Systemen ab. Die Umsetzung der Transaktionen in den verschieden Datenbanksystemen gestaltet sich recht kompliziert. Immerhin muß die auf Kontextebene angestoßene Transaktion so aufbereitet werden, das sie in den einzelnen zu Grunde liegenden Datenbanksystemen verwendet werden kann. Für dieses Problem ist eine Betrachtung des Transaktionsmanagements in verteilten Systemen nötig, welche mit dem Orts- /Zeitkontext nicht direkt zu tun hat. Die Untersuchungen von Transaktionen und deren Einsatz in verschieden Systemen kann z.B. in [Wei88, Voss93] nachgelesen werden.

Die Mobilität des Nutzers ermöglicht es, daß sich ein Nutzer während einer Anfrage bewegt. Bewegt sich der Nutzer während der Bearbeitung von einem Ortsserverbereich in einen anderen, so muß das System den neuen Ort des Nutzer finden, damit ihm gegebenenfalls die angefragten Daten nachgeschickt werden können.

Bei einer ortsabhängigen Verteilung der Datenbestände sind bei Anfragen an verschiedenen Orten, die jeweiligen Datenbanken in den Bearbeitungsprozeß eingebunden. Meistens ist die mobile Netzstruktur nicht überall gleich ausgebaut, so daß an machen Stellen gute Bandbreiten und Übertragungsgeschwindigkeiten vorhanden sind und dagegen in anderen Bereichen dem Nutzer nur eine geringe Bandbreite mit einer niedrigen Transferleistung zur Verfügung steht. Bei großflächigen Anwendungsszenarien besteht zudem das Problem der möglicherweise schlechten Qualität und Stabilität der Verbindungen.

Das Hauptproblem der Transaktionen in mobilen Systemen ist die Möglichkeit der Verbindungslosigkeit. Dabei wird in [Lub96] zwischen der erwünschten und der systembedingten Verbindungslosigkeit unterschieden.

Durch die unerwünschte Verbindungslosigkeit muß man besondere Verfahren einführen, die die Verwaltung von sogenannten "Langen Sperren" ermöglichen. Diese Sperren treten auf, wenn ein Nutzer eine Anfrage stellt. Das Datenbanksystem sperrt die Daten, auf denen der Nutzer arbeitet. Bei einer Unterbrechung der Verbindung zwischen Nutzer und Datenbank an dieser Stelle befinden sich noch aktive Sperren im System.
In [Imil93] werden einige Vorschläge zu Transaktionsverfahren gemacht. Die Vorschläge reichen von komplett pessimistischen Sperrverfahren, die die Sperren während der gesamten Dauer der Verbindungslosigkeit aufrechterhalten bis zu optimistischen Verfahren, die ohne Sperren arbeiten. Diese arbeiten auf replizierten Daten und brauchen daher keine Sperren. Problem dieser Verfahren ist der unumgängliche Abgleich der lokal geänderten Daten mit den serverseitig gespeicherten Daten.
Als Zwischenlösung werden Sperren mit einer festgelegten Lebensdauer vorgeschlagen.

Der Zeitkontext kann die Sperrverfahren so beeinflussen, daß sie sich an die bestehende Situation anpassen. In der Struktur des Zeitkontextes befindet sich das Verfallsdatum einer Anfrage mit zeitabhängigen Daten. Werden durch die Anfrage Sperren benötigt, so kann die maximale Sperrdauer mit dem Verfallsdatum des Zeitkontextes abgeglichen werden. Damit werden Ressourcen, die noch von ungültigen Anfragen gesperrt sind, freigegeben.

6 Fazit und Ausblick

Ausgehend von der Betrachtung und Analyse der möglichen Eigenschaften des Orts-/Zeitkontextes wurde in dieser Arbeit ein Vorschlag zur Realisierung dieser Kontexte für eine mobile Architektur gegeben.
Als Ergebnis der Untersuchung kann man erkennen, daß Ort und Zeit zwei stark zusammenhängende Komponenten sind. Wobei der Ortskontext mit Ausnahme der Bewegungsprofile durchaus ohne die gleichzeitige Benutzung des Zeitkontextes, für den Einsatz in einem Kontextserver geeignet ist.

Die Definition des Ortskontextes für die beiden Referenzbeispiele hat gezeigt, daß eine Realisierung sehr stark von dem umzusetzenden Szenario abhängt. Die allgemeine Funktionalität eines Ortskontextes ist dagegen von der Architektur des mobilen Systems abhängig. In dieser Arbeit wurden nur GPS basierte Beispiele betrachtet, weitere Einsatzgebiete wie das in [LoeMa96] vorgestellte Konzept zur Anwendung von Ortskontexten in Gebäuden mittels Infrarotsendern sind kurz vorgestellt worden.

Die Definition der Kontexte wurde in CORBA IDL vorgenommen. Somit ist die Unabhängigkeit der Kontextschnittstelle von System und Programmiersprache gewährleistet. Für einige Funktionen des Ortskontexts wurden außerdem Vorschläge zur ihrer Implementierung gemacht.
Die Anwendung des Orts-/Zeitkontextes geschieht vornehmlich in der Anfragebearbeitung und im Aggregationsprozeß, bei der Darstellung dagegen spielt er kaum eine Rolle.

Eine Voraussetzung für den Einsatz von Kontexten ist der Zugriff auf die Metadaten des Modells. Anhand der Hierarchieebenen im Ortskontext wurde eine Möglichkeit gezeigt, wie man ermitteln kann, ob ein Kontext für bestimmte Daten anwendbar ist.

Für komplexere Anwendungen ist meiner Meinung nach die Entscheidung der Kontextabhängigkeit von Anfrage bzw. Ergebnissen eines der wichtigsten Probleme. Um die Vielzahl an Zusatzinformationen bereitzustellen, die Kontexte benötigen, erscheint mir die Erweiterung der Metadaten als geeignet. In der Definition des Datenmodells würden dann neue Schlüsselwörter (z.B. Hierarchie, Aktualisierung) mit dazugehörigen Werten stehen.

Zur Übertragung dieser zusätzlichen Definitionen in die jeweiligen Kontextstrukturen ist der Einsatz eines Präprozessors denkbar.

Eine Konzeption und Implementierung eines Kontextservers, für den die Konzepte aus dieser Arbeit und den Arbeiten [Wen97] und [Boj97] umgesetzt werden, könnte sich als interessante Aufgabe anschließen.

7 Abkürzungsverzeichnis

IIOP	Internet Inter ORB Protocoll
ORB	Object Request Broker
CORBA	Common Object Request Broker Architecture
IDL	Interface Definition Language
PDA	Personal Digital Assistant
HPGL	Hewlett Packard Graphic Language
OQL	Object Query Language

8 Abbildungsverzeichnis

9 Literaturverzeichnis

[Boj97] Bojak, Thoralf:
Adaption von Datenbankensystemen an den mobilen Kontext „Information“, Studienarbeit, Fachbereich Informatik, Universität Rostock, November 1997

[Boern] Börnchen, Christian
Darstellung von hierarchisch gegliederten Daten, Studienarbeit, Fachbereich Informatik, Universität Rostock, 1998

[HeuLu96] Heuer, A. ; Lubinski, A. :
Database Access in Mobile Environments, Fachbereich Informatik, Universität Rostock, 1996

[Imil92/1] Imilinski, T.; Badrinath, B.R.:
Querying in highly mobile distributed environments; Proc. Of the 18th VLDB, 1992, S.41-52.

[Imil92/2] Imilinski, T.; Badrinath, B.R. Virmani, A.:
Locating Strategies for Personal Communication Networks, In Workshop on Networking of Personal Communications Applications, December 1992

[Imil93] Imilinski, T; Badrinath, B.R. :
Data Management for Mobile Computing, SIGMOD RECORD, Vol 22, Nr. 1, 1993

[Imil94] Imilinski, T; Badrinath, B.R. :
Mobile Wireless Computing, Solutions and Challenges in Data Management, Communications of the ACM; Vol.37, No.10, 1994

[KroKo93] Koch H.; Krombholz L.; Theel O.:

A Brief Introduction into the World of „Mobile Computing“, THD-BS-1993-03, Fachbereich Informatik, Universität Darmstadt, 21. Mai 1993.

[LoeMa96] Leonhardt, U.; Magee J.; Dias, P.:
Locations Service in Mobile Computing Environments, Workshop IMC ’96, Dept. Of Computing, Imperial College London, 1996.

[Lub96] Lubinski, A.
Datenbanksystem in mobiler Umgebung, FB Informatik, Uni Rostock, 1996

[RajaBa] Rajagopalan, S. ; Badrinath, B.R. :
An Adaptive Location Management Strategy for Mobile IP, Dept. Of CS, Rutgers University New Brunswick

[Schill] Schillit, B. ; Adams, N. ; Want, R. :
Context-Aware Computing Applications, In Proc. of the IEEE Workshop Mobile Computing Systems ans Applications, Santa Cruz, 1994

[Wat96] Waterstraat, Jörn:
Definition und Nutzung von Kontexten in mobiler, verteilter Datenbankumgebung, Diplomarbeit, Fachbereich Informatik, Universität Rostock, 1996.

[Wei88] Weikum, Gerhard:
Transaktionen in Datenbanksystemen, Addison-Wesley, Bonn, 1988.

[Wen97] Wendt, Karsten:
Kontextabhängige Aggregation heterogener Daten, Studienarbeit, Fachbereich Informatik, Universität Rostock, 1996.

[Voss93] Vossen, Gottfried; Groß-Hardt, Margret:
Grundlagen der Transaktionsverarbeitung, Addison-Wesley, Bonn, 1993.