Published in: Proc. Workshop Mobilität und Informationssysteme,
16.-17. Okt. 2003, Zürich, ETH Zürich Rep. Nr. 422, 2003, pp. 67-76

Implementierungskonzepte und Anwendungsentwicklung kommerzieller mobiler Datenbanksysteme – Ein Vergleich –

Bela Mutschler, Günther Specht

Universität Ulm
Abteilung Datenbanken und Informationssysteme (DBIS)
89069 Ulm
bela@mutschler.info
specht@informatik.uni-ulm.de

Abstract. In diesem Papier werden vier mobile Datenbanksysteme – *IBM DB2 Everyplace*, *Oracle 9i Lite*, *Sybase UltraLite* und *Tamino Mobile* – bezüglich ihrer Architekturen und Implementierungskonzepte sowie ihrer Unterstützung bei der Entwicklung mobiler Anwendungen verglichen. Darüberhinausgehende Aspekte der mobilen Datenbanksysteme *Pointbase Micro* und *eXtremeDB* werden ebenfalls berücksichtigt. Schwerpunkte des Papiers sind, neben den unterschiedlichen Architekturkonzepten, die von den Datenbanksystemen bereitgestellten Verfahren zur Replikation und Synchronisation.

1 Einleitung

Aus den Bedürfnissen einer effizienteren mobilen Datenverarbeitung heraus entstanden in den letzten Jahren eine ganze Reihe kommerzieller mobiler Datenbanksysteme. Dabei konkurrieren in ihrer Funktionalität sehr umfassende Produkte großer Hersteller wie IBM oder Oracle mit eher schlankeren Produkten kleinerer Hersteller wie Pointbase oder Sybase. Für die Auswahl und den Einsatz eines bestimmten Produkts ist es deshalb wichtig, die den einzelnen Produkten zugrunde liegenden Konzepte und Implementierungsdetails zu kennen und einschätzen zu können. Gerade im performanzkritischen Bereich von Replikation und Synchronisation existieren dabei große Unterschiede. Im Folgenden werden die mobilen Datenbanksysteme *IBM DB2 Everyplace*, *Oracle 9i Lite*, *Sybase UltraLite* und *Tamino Mobile* untersucht und verglichen. Schwerpunkte bilden Replikation und Synchronisation, aber auch die Unterstützung bei der Entwicklung mobiler Datenbankanwendungen.

Das Papier gliedert sich wie folgt. Zunächst werden in Kapitel 2 Implementierungskonzepte erläutert und verglichen. Abschnitt 2.1 geht kurz auf die wichtigsten Aspekte der jeweils realisierten Anwendungsarchitekturen ein. Während Abschnitt 2.2 Replikations-

verfahren untersucht und vergleicht, fokussiert Abschnitt 2.3 bereitgestellte Synchronisationsverfahren. Abschließend stellt Kapitel 3 die verfügbaren Tools und Programmierschnittstellen der vier Systeme zur Unterstützung der Nutzer bei der Entwicklung mobiler Datenbankanwendungen gegenüber.

2 Untersuchung und Vergleich von Implementierungskonzepten

2.1 Anwendungsarchitekturen

In diesem Abschnitt werden die von den untersuchten Datenbanksystemen realisierten Anwendungsarchitekturen kurz beschrieben und eingeordnet. Dabei lassen sich zwei Ansätze unterscheiden. Dem klassischen, um spezielle Middleware ergänzten Client/ Server-Ansatz, wie er bei *IBM DB2 Everyplace*, *Oracle 9i Lite* und *Tamino Mobile* realisiert wird (Abbildung 1), steht als Alternative die vollständige Integration der gewünschten Datenbankfunktionalität in eine Anwendung (Abbildung 2) gegenüber. *Sybase UltraLite*, *Pointbase Micro* und *eXtremeDB* sind Vertreter dieser zweiten Kategorie. Während im ersten Fall das DBMS der mobilen Datenbank von den auf sie zugreifenden Anwendungen getrennt bleibt, wird beim letzteren ein in seiner Funktionalität angepasstes Datenbanksystem vollständig in eine Anwendung „einkompiliert". Die Unabhängigkeit des Datenbanksystems von den Anwendungen wird in diesem Fall aufgegeben. Wir geben im Folgenden einen kurzen Überblick über die einzelnen Systeme, bevor wie in den Abschnitten 2.2 und 2.3 auf Konzepte der Replikation und Synchronisation genauer eingehen.

DB2 Everyplace-basierte Anwendungen [3, 4, 6] bestehen aus dem mobilen Datenbanksystem *DB2 Everyplace*, einem Mid-Tier-Server mit zusätzlicher Synchronisationskomponente und einem DB-Server im Backend. Das mobile Kerndatenbanksystem benötigt einen Speicherplatz von 180-220 kB. Für Transaktionen kann Atomarität und Dauerhaftigkeit (durch den *Recovery Manager*) garantiert werden [1]. Lokale Konsistenz auf dem Client ist immer gegeben, da es sich bei der mobilen Datenbank um ein klassisches Einbenutzer-System handelt. Einzelne Tabellen können aus Sicherheitsgründen verschlüsselt werden. Die *SyncServer*-Komponente des Mid-Tier-Servers ist für die Steuerung eines Synchronisationsvorganges zuständig. Über diese Komponente können sowohl Daten als auch Anwendungen zwischen mobilen Clients und Datenquellen im Backend synchronisiert werden. Der *SyncClient* ist die auf den mobilen Clients für die Steuerung der bidirektionalen Synchronisation zuständige Komponente. *DB2 Everyplace* baut auf einer klassischen Client/Server-Architektur auf, die um eine Middleware-Komponente erweitert ist (*3-Tier-Architecture*).

Oracle 9i Lite [7] verwendet genau wie *DB2 Everyplace* eine *3-Tier-Architecture*, gebildet aus mobilen Clients, einem zwischengeschalteten *Mobile-Server* und einem DB-Server im Backend. Alle Architekturkomponenten werden zusammen als *Mobile Server Environment* bezeichnet. Das mobile Kerndatenbanksystem *Oracle Lite* benötigt einen Speicherplatz von 50 kB-1MB. Auf dem mobilen Client werden für Transaktionen die ACID-Eigenschaften garantiert.

Transaktionen werden dabei explizit mit `COMMIT` oder `ROLLBACK` abgeschlossen. Der *Mobile-Server* entspricht in seiner Aufgabenstellung weitestgehend dem *SyncServer* von *DB2 Everyplace*. Einen drahtlosen Zugriff unterschiedlich leistungsstarker mobiler Clients ermöglicht der *Oracle 9i Application Server (Wireless Edition)* – als Teil des *Mobile Servers*. Im Backend können ausschließlich *Oracle*-Datenbanken eingesetzt werden, während *DB2 Everyplace* hier auch andere ODBC/JDBC-Datenbanken zulässt. Neben der Speicherung der Anwendungsdaten wird im Backend auch ein spezieller, für die Replikation relevanter Datenbereich, das *Mobile-Server Repository* (MSR), verwaltet. Um eine korrekte Replikation zwischen Backend und Clients zu ermöglichen, ist es bei Oracle notwendig, sowohl die zu replizierenden Anwendungsdaten, als auch das MSR in der gleichen Datenbank zu speichern.

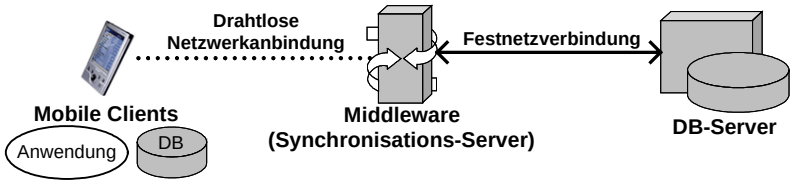

Abbildung 1: *3-Tier-Architecture* von *DB2 Everyplace*, *Oracle 9i Lite* und *Tamino Mobile*.

Auch ***Tamino Mobile*** [9] geht von einer erweiterten Client/Server-Architektur aus. Auf den mobilen Clients kommt als mobiles Kerndatenbanksystem die *Tamino Mobile Database* (TMDB) zum Einsatz. Der Unterschied zu *DB2 Everyplace*, *Oracle 9i Lite* und *Sybase UltraLite*, die alle (objekt-)relationale Datenmodelle implementieren, liegt in der Unterstützung von XML durch TMDB. So erlaubt TMDB das Suchen und Navigieren innerhalb von XML-Daten mit den W_3C-Abfragesprachen *XQuery* und *XPath*. Synchronisationsmechanismen erlauben den Abgleich von Daten zwischen mobilen Clients und einem zentralen *Tamino XML-Server*. Ein spezielles *SyncModule* ist dabei für die Kontrolle des Synchronisationsvorganges mit einem entfernten *Tamino XML-Server* zuständig. Der TMDB ist ein Web-Server vorgeschaltet, der ein browser-basiertes Administrations- und Arbeitsinterface zur Verfügung stellt. Der *Tamino XML-Server* erlaubt die Anbindung von weiteren, auch auf anderen Datenmodellen basierenden DB-Servern.

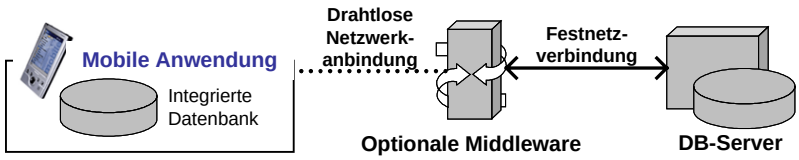

Abbildung 2: *3-Tier-Architecture* von *Sybase UltraLite*, *Pointbase Micro* und *eXtremeDB*.

Sybase UltraLite [8] ermöglicht es, eine Datenbank zusammen mit der zugehörigen Datenverwaltungslogik vollständig in eine Anwendung „einzukompilieren“. Anwendung, Datenbank und Datenverwaltungslogik bilden zusammen eine *UltraLite*-Anwendung. Dies hat jedoch die Aufgabe des Prinzips der Datenunabhängigkeit zur Folge. Eine in eine Anwendung einkompilierte Datenbank ist dadurch nur für die dedizierte Host-Anwendung zugänglich.

Prinzipiell bietet *UltraLite* die Funktionalität und Zuverlässigkeit einer vollständigen SQL-Datenbank. Die von einer Anwendung nicht benötigten Datenbank-Features müssen auf dem mobilen Client jedoch auch nicht verfügbar sein und werden entsprechend auch nicht mit in eine *UltraLite*-Anwendung integriert. Dies spart Speicherplatz. Auf die gleiche Weise gehen auch die mobilen Datenbanksysteme ***Pointbase Micro*** [10] und ***eXtremeDB*** [11] vor. Auch sie integrieren die erforderliche Datenbankverwaltungsfunktionalität und damit das mobile DBMS direkt in die Anwendung.

In ihrer grundlegenden Anwendungsarchitektur unterscheiden sich die untersuchten Datenbanksysteme somit vor allem in Bezug auf die Architektur der mobilen Kerndatenbanksysteme. Während eine Seite eine saubere Trennung zwischen Anwendung und Datenbankmanagementsystem aufrechterhält, wird diese Unabhängigkeit auf der anderen Seite aufgegeben, in dem ein in seiner Funktionalität angepasstes Datenbanksystem in eine Anwendung einkompiliert wird.

2.2 Replikationsverfahren

Um das Arbeiten mit mobilen Clients im unverbundenen Zustand (*Disconnected Mode*) zu unterstützen, ist es notwendig, während einer bestehenden Netzwerkverbindung (*Connected Mode*) Daten eines DB-Servers auf das mobile Datenbanksystem des mobilen Clients zu replizieren. Die Datenbanksysteme *DB2 Everyplace*, *Tamino Mobile* und *Sybase UltraLite* fassen Replikation lediglich als eine Teilphase bidirektionaler Synchronisation vom DB-Server zum mobilen Client auf. *Oracle 9i Lite* definiert dagegen ein dediziertes Replikationsverfahren.

Beim von ***DB2 Everyplace*** realisierten Replikationsverfahren (als Teilphase der Synchronisation) werden Daten eines DB-Servers (*Source Database*) zunächst in eine Austauschrelation (*Change Data Table*) übertragen. Diese befindet sich physisch betrachtet ebenfalls auf dem DB-Server. Von dort aus werden die Daten über eine Spiegeltabelle (*Mirror Table*) einer Spiegeldatenbank (*Mirror Database*) auf dem *Mid-Tier-Server* auf die mobilen Clients repliziert. Replikation wird dabei nicht nur als initiale Übertragung von Daten in eine neue, noch leere mobile Datenbank, sondern auch als Propagierung von auf dem DB-Server durchgeführten Änderungen an die mobilen Clients verstanden. Auch bei ***Sybase UltraLite*** und ***Tamino Mobile*** kann Replikation (wie bei *DB2 Everyplace)* als Teilphase bidirektionaler Synchronisation aufgefasst werden. *UltraLite* nutzt dazu die *MobiLink*-Technik, die ein sitzungsbasiertes, bidirektionales Synchronisationsverfahren implementiert. Replikation bei *Tamino Mobile* erfolgt über eine dedizierte *DataLoader*-Komponente und den Einsatz weiterer Synchronisationskomponenten.

Replikationsquelle bei ***Oracle 9i Lite*** muss eine Oracle-Datenbank sein. Mobile Clients, die den Replikationssenken entsprechen, können keine Replikationsquelle sein. Die Replikation erfolgt dabei unidirektional vom DB-Server zum Client. Das implementierte Replikationsverfahren basiert auf der Verwendung von *Snapshots*. Replikationsserver werden als *Master-Sites*, Clients als *Snapshot-Sites* bezeichnet. Die Replikation kann verbindungsbasiert oder dateibasiert erfolgen. Die Kommunikation zwischen Replikationsserver und Clients erfolgt bei verbindungsbasierter Replikation synchron, bei dateibasierter Replikation asynchron.

Dateibasierte Replikation erfolgt asynchron unter Verwendung von Dateien, die in einem gemeinsam zugreifbaren Speicherbereich abgelegt sind. Der Dateitransfer kann auf verschiedene Arten erfolgen. Eine Möglichkeit besteht darin, Replikationsdateien manuell (beispielsweise über FTP) zu verteilen. Alternativ kann auch ein automatischer Dateitransfer mit Hilfe einer dedizierten Middleware-Komponente erfolgen, dem *Oracle Mobile Agent* (OMA). Als dritte Möglichkeit, einen Dateitransfer zwischen Replikationsquelle und Replikationssenke zu implementieren, kann eine HTTP-Verbindung samt zwischengeschaltetem *Oracle Web Application Server* verwendet werden. Dabei ist der Applikationsserver für die automatische, asynchrone Weitergabe aller Replikate an die Clients zuständig (Standardfall). Snapshots, die den Basiseinheiten des Oracle-Replikationsverfahrens entsprechen, sind materialisierte Sichten nicht-lokaler Datenbestände. Snapshots basieren auf Originaltabellen, die auch als *Mastertabellen* oder *Mastersichten* bezeichnet werden. Zur Replikation werden auf dem *Mobile-Server* so genannte *Publikationen* erzeugt. Publikationen bestehen aus einer Menge von *Publikationsartikeln* (die den Snapshots entsprechen), die über *Subskriptionen* mobilen Clients zugeordnet sind. Publikationsartikel werden mittels SQL-Anweisungen definiert und zu Publikationen zusammengefasst. Subskriptionen können parametrisiert sein.

DBS	Konzepte des implementierten Replikationsverfahrens
DB2 Everyplace	Teilphase des bidirektionalen Synchronisationsprozesses. Über die Spiegeltabelle (*Mirror Table*) einer zwischengeschalteten Spiegeldatenbank (*Mirror Database*) werden Daten repliziert.
Oracle 9i Lite	Replikation kann entweder verbindungsbasiert oder dateibasiert erfolgen. Basiseinheit der Replikation sind Snapshots (Publikationsartikel).
Tamino Mobile	Teilphase des bidirektionalen Synchronisationsprozesses. Zur Replikation wird eine dedizierte DataLoader-Komponente eingesetzt.
Sybase UltraLite	Teilphase des bidirektionalen Synchronisationsprozesses. Zur Replikation wird das sitzungsbasierte MobiLink-Verfahren herangezogen.

Tabelle 1: Vergleich der implementierten Replikationsverfahren.

2.3 Synchronisationsverfahren

Ziel der Synchronisation ist die Aufrechterhaltung bzw. Wiederherstellung eines konsistenten Zustands aller Kopien eines Datenelementes innerhalb einer Replikationsumgebung. Ein Synchronisationsprozess besteht aus zwei verschiedenen Phasen [2]. Die *Reintegration* überträgt auf dem mobilen Client geänderte Daten auf den DB-Server (*Phase 1*). Bei der anschließenden (optionalen) *Rückübertragung* werden zwischenzeitlich auf dem Datenbestand des Servers vorgenommene Änderungen auf die Clients übertragen (*Phase 2*). Alle in diesem Papier untersuchten mobilen Datenbanksysteme bieten Verfahren zur bidirektionalen Synchronisation an.

Der *SyncServer* ist die bei ***DB2 Everyplace*** für die Koordination der Synchronisation zuständige Komponente. Der *SyncClient* ist die korrespondierende Synchronisationskomponente auf den mobilen Clients. Sowohl *SyncClient* (= *SyncEngine* in Abbildung 3) als auch *SyncServer* (= *SyncML Synchronizer using WBXML* in Abbildung 3) basieren auf Synchronisations-Engines, die über optional ansteckbare Adapter die Anbindung unterschiedlichster Synchronisationsquellen ermöglichen.

Die Synchronisations-Engines kommunizieren untereinander über eine *Java Servlet-Engine*, die Bestandteil des *SyncServers* ist. Als Übertragungsformat der Synchronisations-Nachrichten wird *Wireless Application Protocol (WAP) Binary encoded XML* (WBXML) verwendet. Dabei handelt es sich um eine kompakte XML-Repräsentation, die die Übertragungsgröße der Synchronisations-Nachrichten stark reduziert. Verschiedene Kommunikationsadapter stellen unterschiedliche Möglichkeiten eines drahtlosen Zugriffs (auf die *Java Servlet-Engine*) zur Verfügung. Für die Synchronisation werden Nutzer zu Gruppen zusammengefasst. Ein zu synchronisierendes Datenelement wird in einer *Subscription* gekapselt, die wieder zu *Subscription Sets* aggregiert werden. Nutzergruppen können solche *Subscription Sets* zugewiesen werden.

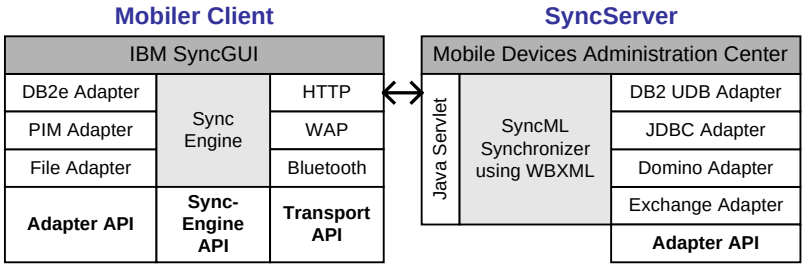

Abbildung 3: Synchronisationsarchitektur von *DB2 Everyplace* [6].

Die Synchronisation vom DB-Server hin zu mobilen Clients kann auch als Replikation aufgefasst werden und wurde bereits in Abschnitt 2.2 erläutert. Die Synchronisation vom mobilen Client hin zum DB-Server verläuft technisch gesehen prinzipiell gleich. Nach der notwendigen Authentifizierung einer Synchronisationsanforderung (*Synchronization Request*) eines mobilen Clients werden geänderte Daten über eine Zwischentabelle (*Staging Table)* in eine Spiegeltabelle (*Mirror Table)* einer Spiegeldatenbank (*Mirror Database*) übertragen. Neu ist, dass an dieser Stelle eventuell auftretende Synchronisationskonflikte durch den *SyncServer* gelöst werden müssen. Dazu greift er auf verschiedene (interne) Konfliktlösungsstrategien zurück, ohne zu garantieren, dass auch tatsächlich alle Konflikte erkannt oder behoben werden. Ausgehend von der Spiegeltabelle werden die Änderungen mit den ursprünglichen Quelldaten des DB-Servers abgeglichen. Synchronisationsvorgänge können manuell oder automatisch unter Verwendung der *IBM SyncEngine API* angestoßen werden.

Oracle 9i Lite definiert Synchronisation als Abgleich von Snapshots. Ein Synchronisationsvorgang kann einerseits automatisch durch einen serverseitigen Datenbankprozess initialisiert werden. Andererseits können für eine individuelle Initialisierung spezielle *Java Replication Classes* bzw. *Replication OLE Controls* verwendet werden. Bei Snapshots unterscheidet *Oracle 9i Lite* zwischen einfachen und komplexen Snapshots. *Einfache Snapshots* basieren auf einer einzelnen Tabelle. Zwischen Originaltabelle und Snapshot muss eine bijektive Abbildung existieren (einfache Sichten). Nur so können auf einem Snapshot durchgeführte Änderungen auf die Originaltabelle übertragen werden. Im Falle eines Fehlers muss das Wiederherstellen eines konsistenten Datenbankzustands ermöglicht werden. Dazu kann bei einfachen Snapshots für die Originaltabelle ein *Snapshot Transaction Log* geführt werden.

Komplexe Snapshots entsprechen komplexen Sichten. Für sie ist in Oracle keine Reintegration auf dem Server möglich, daher darf auf sie auf dem mobilen Client nur lesend zugegriffen werden. Komplexe Snapshots können sich nicht nur auf verschiedene Tabellen, sondern auch auf andere Snapshots beziehen. Einzige Voraussetzung: Datenstrukturen, auf die sie sich beziehen, müssen auf dem gleichen DB-Server verfügbar sein. Das Snapshot-Reintegrationsproblem entspricht damit dem klassischen View-Update-Problem. Snapshots werden mittels der Anweisung `CREATE SNAPSHOT` angelegt. Eine optionale `FOR UPDATE`-Klausel legt fest, ob es sich um einen einfachen oder komplexen Snapshot handelt. Die oben angesprochene bijektive Abbildungsvorschrift einfacher Snapshots wird durch eine Datenbankanfrage, die so genannte *Snapshot-Anfrage*, bestimmt. Abhängig vom Typ eines Snapshots kann zwischen verschiedenen Synchronisationsverfahren unterschieden werden. Im Zuge einer *vollständigen Aktualisierung* (*Full Refresh*) werden alle Tupel, die im Ergebnis einer Snapshot-Anfrage auftauchen, vom Replikationsserver in die Snapshot-Tabelle des mobilen Clients übertragen. Vor der Übertragung werden alle existierenden Tupel der Snapshot-Tabelle gelöscht, so dass immer in eine völlig leere Tabelle geschrieben wird. Bei der *schnellen Aktualisierung* (*Fast Refresh*) werden die in den Snapshot-Logs protokollierten Änderungen an den Originaltabellen für eine effizientere Aktualisierung der Snapshots verwendet. So werden beim *Fast Refresh* ausschließlich die protokollierten Änderungen übertragen. Der Begriff der *schnellen Aktualisierung* kann zu Missverständnissen führen. Zwar wird der Eindruck erweckt, die schnelle Aktualisierung sei die in jedem Fall die effizientere Synchronisationsmethode. Dies ist jedoch nicht immer richtig. Bei kleinen Tabellen, die ein hohes `UPDATE`-Aufkommen vorweisen, ergibt sich eine umgekehrte Situation. In diesem Fall kann die Übertragung aller protokollierten Änderungen wesentlich aufwendiger sein, als eine alternative vollständige Aktualisierung. *Force Refresh* implementiert eine Mischung der beiden zuerst genannten Techniken. Zunächst wird eine schnelle Aktualisierung versucht. Scheitert diese, beispielsweise aufgrund fehlerhafter Snapshot-Logs, wird eine vollständige Aktualisierung durchgeführt. Nach der Initialisierung eines Synchronisationsvorgangs wird eine so genannte *Refresh Group* erstellt. Diese fasst alle Datenbanktabellen respektive Snapshot-Tabellen zusammen, die synchronisiert werden sollen. Durch Methoden der *Java Replication Classes* und der *Replication OLE Control* können der Refresh-Gruppe Snapshots hinzugefügt oder entfernt werden. Nach der Definition einer Refresh-Gruppe beginnt der eigentliche Synchronisationsvorgang. Dieser gliedert sich in zwei Phasen: Im ersten Schritt werden alle Änderungen einfacher Snapshots seit dem letzten Synchronisationsvorgang an den DB-Server übertragen (→ Reintegration). Die Transaktionen der mobilen Clients werden auf der Server-Datenbasis nachvollzogen. Im zweiten Schritt werden an den Originaltabellen durchgeführten Änderungen auf die mobilen Clients übertragen (→ Rückübertragung).

Das mobile Datenbanksystem ***Sybase UltraLite*** implementiert mit *MobiLink* ein sitzungsbasiertes Synchronisationsverfahren, dessen Ablauf durch den *MobiLink*-Server koordiniert wird. Der *MobiLink*-Server kontrolliert für alle Clients sowohl die Ausführung der Synchronisation, als auch die zur Synchronisation benötigten Verwaltungsdaten (zur Ablaufsteuerung werden spezielle SQL- und Java-Skripte genutzt).

Teilaufgaben sind die Koordination der Datenübertragung, die Automatisierung und Überwachung von Synchronisationssitzungen, die Auswahl von Kommunikationsprotokollen und die Erstellung von Reports, die dem DB-Administrator helfen können, Fehler aufzuspüren und zu beseitigen. Der *MobiLink*-Synchronisationsserver garantiert Transaktionsintegrität, d.h. eine Datenbank wird entweder ganz synchronisiert oder gar nicht. Die Synchronisation erfolgt wieder in zwei Phasen. Jeder mobile Client muss protokollierte Änderungsoperationen adäquat aufbereiten, um sie anschließend dem *MobiLink*-Server zu übermitteln. Genauso muss ein mobiler Client in der Lage sein, Daten in seine lokale Datenbank integrieren zu können, die vom *MobiLink*-Server (respektive dem DB-Server) übermittelt wurden.

Tamino Mobile fasst alle Synchronisationskomponenten zum *MobileLogic-Framework* zusammen. Für die Durchführung der Synchronisation ist der *Tamino Mobile Smart Client* (TMSC) zuständig. Der TMSC besteht aus dem *Smart Client* (SC) und dem *Tamino Mobile Synchronization Server* (TMSS). Der SC ist auf der mobilen Einheit installiert und nutzt ein spezielles *SyncModule*, das für die Steuerung eines Synchronisationsvorgangs zuständig ist. Die notwendigen Synchronisationsvorgänge selbst werden vom Entwickler noch während der Implementierung einer mobilen Anwendung definiert. Auf dem TMSS werden Arbeitsvorgangsvorlagen (*Workflows*) definiert, die genau festlegen, wie Daten, die von einer mobilen Einheit geschickt wurden, bearbeitet werden. Andere Workflows definieren Regeln zum Generieren neuer Daten, die an bestimmte mobile Einheiten geschickt werden müssen.

DBS	**Konzepte des implementierten Synchronisationsverfahrens**
DB2 Everyplace	Synchronisation erfolgt über den Umweg einer Spiegeltabelle (*Mirror Table*) des *Mid-Tier-Servers*. Auftretende Synchronisationskonflikte werden durch den *SyncServer* erkannt und aufgelöst.
Oracle 9i Lite	Synchronisation entspricht einer Aktualisierung von Snapshots. Dazu werden verschiedene Verfahren (*Full Refresh*, *Fast Refresh*, *Force Refresh*) angeboten. Man unterscheidet zwischen einfachen (änderbaren) und komplexen (nicht änderbaren) Snapshots.
Tamino Mobile	Synchronisation erfolgt unter Verwendung spezieller Synchronisationskomponenten (*Synchronization Server*, *SyncModule*).
Sybase UltraLite	Unterstützung eines bidirektionalen, sitzungsbasierten Synchronisationsverfahren, genannt *MobiLink*.

Tabelle 2: Vergleich der implementierten Synchronisationsverfahren.

3 Anwendungsentwicklung und Ausblick

Alle vier mobilen Datenbanksysteme realisieren Infrastrukturen für den Einsatz datenbankgestützter mobiler Anwendungen. Genauso wie sich die Systeme in den Konzepten ihrer Implementierung teilweise erheblich unterscheiden, hat eine praktische Evaluierung aller Systeme auch Unterschiede in Bezug auf die Unterstützung der Nutzer bei der Erstellung mobiler Datenbankanwendungen gezeigt.

3.1 Anwendungsentwicklung

Mit ***DB2 Everyplace*** kann IBM wohl die vollständigste und am flexibelsten einsetzbare Lösung vorweisen. Besonders gelungen ist in diesem Zusammenhang der *Mobile Application Builder*, einem Entwicklungstool, mit dem schnell und einfach mobile Datenbankanwendungen erstellt werden können. Als Mitglied der DB2-Produktfamilie ist *DB2 Everyplace* weitestgehend auf die Zusammenarbeit mit anderen IBM-Komponenten abgestimmt, trotzdem können beispielsweise im Backend auch Datenbanksysteme anderer Hersteller als Replikationsquellen herangezogen werden. *DB2 Everyplace* unterstützt die dynamische Ausführung von SQL-Statements. Auf Datenbanktabellen kann über die Standard-APIs ODBC/CLI und JDBC zugegriffen werden.

Auch Oracle bietet mit ***Oracle 9i Lite*** genau wie IBM eine ausgereifte Lösung für die Entwicklung mobiler Datenbankanwendungen an. Alle grundlegenden Anforderungen an mobile Datenbanksysteme werden erfüllt. *Oracle Lite* basiert auf einem Objekt-Kernel, auf den eine zusätzliche SQL-Schicht aufgesetzt ist. Auf der SQL-Schicht setzen die klassischen Anwendungsschnittstellen für den Zugriff auf Datenbanken, ODBC und JDBC, auf. Um die Funktionen des Kernels direkt ansprechen zu können, wird ein proprietäres *Call Level Interface* bereitgestellt, welches ein *Object Kernel Application Interface* (OKAPI) implementiert, das jedoch ausschließlich von C/C++-Anwendungen genutzt werden kann. Eine integrierte Entwicklungsumgebung, vergleichbar mit dem MAB von *DB2 Everyplace*, kann Oracle selbst jedoch nicht vorweisen. Diverse „normale“ Entwicklungsumgebungen, wie beispielsweise der *JDeveloper* von Oracle, unterstützen jedoch die Entwicklung mobiler Anwendungen.

Mit ***Tamino Mobile*** (TMS) wird von der Software AG ein leistungsstarker Framework für die Entwicklung XML- und Java-basierter mobiler Anwendungen bereitgestellt. Mit dem *Tamino Mobile Studio* steht wie bei *DB2 Everyplace* eine Entwicklungsumgebung für die schnelle Erstellung von *Tamino Mobile*-Anwendungen bereit. Die Unterstützung extrem vieler (DeFacto-)Standards, wie Java, J2EE, XHTML, XML, XSL, VoiceXML oder WAP fördert einerseits die Definition und Verfügbarkeit eines offenen Frameworks, andererseits führt dies aber zu einer rasant zunehmenden Komplexität bei der Konzeption und Realisierung mobiler Anwendungen.

Sybase UltraLite weist ein, im Vergleich zu anderen Anbietern, komplementäres Konzept auf, das durch die vollständige Integration bzw. „Einkompilierung“ einer Datenbank samt Datenbanklogik in eine *UltraLite*-Anwendung gekennzeichnet ist. Dadurch kann der benötigte Speicherplatz des Datenbanksystems reduziert werden. Dies erlaubt die Konzipierung und Realisierung von mobilen Anwendungen nicht nur auf PDAs oder Notebooks, sondern auch auf extrem leistungsschwachen mobilen Einheiten, wie Handys oder einfachen Organizern. Im Prozess der Anwendungsentwicklung (es stehen die Programmiersprachen C/C++ und Java zur Verfügung) muss die in ihrer Funktionalität angepasste Datenbank (Anwendungsdatenbank) in eine *UltraLite*-Anwendung integriert werden. Dabei wird auf eine Referenzdatenbank zurückgegriffen, die physisch auf einer *Adaptive Server Anywhere*-Installation eingerichtet sein muss.

Adaptive Server Anywhere ist ein von Sybase angebotenes Hochleistungs-Datenbanksystem. Die Referenzdatenbank wird ausschließlich während der Entwicklungsphase benötigt. Auf ähnliche Weise können auch ***Pointbase Micro*** und ***eXtremeDB*** eingeordnet werden (auch wenn bei diesen beiden Produkten die tatsächliche Anwendungsentwicklung anders erfolgt).

3.2 Ausblick

Aktuelle Tendenzen im Bereich mobiler Datenbanken zeigen, dass die prinzipiellen Problemstellungen weitestgehend gelöst sind, während technische Details durchaus noch Probleme bereiten. Fortschritte sind vor allem in der Unterstützung der Anwendungsentwicklung zu erwarten. Der *Mobile Application Builder* von IBM zeigt diesbezüglich die Richtung auf. Zusätzliche Impulse sind auch durch die Zielsetzung weitestgehender Automatisierung aller Aspekte von Replikation und Synchronisation zu erwarten. So hat IBM das *Automatic Computing* – und das nicht nur im Datenbankbereich – zu einem neuen Entwicklungsschwerpunkt erhoben.

Schwierigkeiten – insbesondere bei der Konzeption und Realisierung mobiler Datenbankanwendungen – macht die große Vielfalt existierender Mobilplattformen (*Palm OS*, *Windows Mobile 2003*, *Symbian OS*, etc.), die die Entwicklung universell einsetzbarer mobiler Anwendungen stark einschränkt. Die Einführung des *Mobile Information Device Profile 2* (MIDP2) stellt aber einen neuen Versuch der Etablierung einer plattformunabhängigen Mobilplattform dar. Alle großen Anbieter von Datenbanktechnologie (IBM, Oracle, Sybase, Software AG) sind im Segment mobiler Datenbanken vertreten. Nischenanbieter wie *Pointbase Micro* oder *eXtremeDB* werden sich dagegen nur im Bereich sehr spezieller Anwendungsdomänen etablieren können.

Literaturverzeichnis

[1] T.Fanghänel, J.S. Karlsson, C.Leung. DB2 Everyplace Database Release 8.1: Architecture and Key Features. Datenbank Spektrum - Zeitschrift für Datenbanktechnologie. Heft 5/2003. Mai 2003. pp 9-15.

[2] C.Gollmick. Nutzerdefinierte Replikation zur Realisierung neuer mobiler Datenbankanwendungen. GI-Reihe Lecture Notes in Informatics. Band P-26. pp 453-462. 2003.

[3] R.Greenwald, R.Stackowiak, J. Stern. Database Management. White Paper. 2002.

[4] N.Mielke. DB2 Everyplace: An Overview. IBM Research, Silicon Valley Lab. 2001.

[5] A.Noether. Extending Enterprise Data to Mobile Devices. IBM White Paper. 2002.

[6] DB2 Everyplace Dokumentation. Juni 2003.

[7] Oracle 9i Lite Dokumentation. Juni 2003.

[8] SQL Anywhere Studio 8 Dokumentation. Juni 2003.

[9] Tamino Mobile Dokumentation. Juni 2003.

[10] Pointbase Micro Dokumentation. Juni 2003.

[11] eXtremeDB Dokumentation. Juni 2003.