Review of Data Management Mechanisms on Mobile Devices

Überblick über Datenmanagementmechanismen auf mobilen Endgeräten

Hagen Höpfner, Essam Mansour, International University in Germany, Bruchsal,
Daniela Nicklas, University of Oldenburg

Summary The design of data management for mobile applications is a non-trivial task. Even with today's much more powerful devices, we must consider limited resources: size and persistence of the local data storage, limited bandwidth and reliability of connection for data transmissions to remote servers, and constraint energy consumption of the used algorithms. In this article we focus on client/server approaches. We classify existing techniques and support the designer of data management functionalities for mobile applications with a set of fundamental characteristics of replication, hoarding and caching solutions.

►►► **Zusammenfassung** Der Entwurf des Datenmanagements mobiler Anwendungen ist eine nichttriviale Aufgabe. Selbst mit den heute gängigen leistungsstärkeren Endgeräten müssen Ressourcenbeschränkungen wie die Größe und Persistenz lokaler Daten, limitierte Bandbreite und Netzverfügbarkeit bei der Kommunikation mit Servern oder der Energiebedarf der Datenspeicherung berücksichtigt werden. In diesem Artikel fokussieren wir auf Client/Server-Verfahren. Wir klassifizieren existierende Techniken und unterstützen den Designer der Datenmanagementfunktionalität mobiler Anwendungen durch eine Zusammenstellung fundamentaler Charakteristika von replizierenden, hortenden und cachenden Lösungen.

KEYWORDS H.2.4 [Information Systems: Database Management: Systems]; H.2.8 [Information Systems: Database Management: Database Applications]; H.3.2 [Information Systems: Information Storage and Retrieval: Information storage]; H.3.4 [Information Systems: Information Storage and Retrieval: Systems and Software]; caching, hoarding, replication, mobile information systems / Caching, Horten, Replikation, mobile Informationssysteme

1 Introduction and Motivation

Data management in mobile environments is affected by the characteristics of used networks and devices. Mobile devices are battery powered and wireless networks are, compared to wired networks, unreliable. Furthermore, wireless data transmission is energy intensive. Extensive communication reduces the operation time of mobile devices [12]. Even client/server systems with mobile clients that use fixed networked components as backbone *must* support redundant data storage and handling. If no servers are involved in the system, like in so called peer-to-peer (P2P) approaches, network stability is even worse and data redundancy becomes essential. However, in this article we focus on client/server approaches. We classify existing techniques and support the designer of data management functionalities for mobile applications with a set of fundamental characteristics of replication, hoarding and caching solutions. Hence, we guide the developer to decide, based on the specifics of the target application, to use a proper data management strategy. Furthermore, even managing data on the mobile device without any communication requires a deeper look into the systems. In many cases only rudimentary data managing methods are provided to the application developer. Therefore, we give also a brief introduction in how to store/manage data on mobile devices.

The remainder of the paper is structured as follows: Section 2 describes the storage of data on mobile devices. Section 3 focuses on approaches for managing data in client/server systems with mobile clients. Section 4 summarizes the paper.

2 Storing Data on Mobile Devices

Mobile devices like PDAs, mobile phones or other programmable lightweight devices are able to store data in various ways, but handling data storage very often strongly depends on the operating system and the supported programming languages. Furthermore, mobile devices differ in used storage media. Some use volatile memory (like older Palm devices), some use non-volatile flash memory, some use harddisks, and some combine these media. Current trends go in the direction of a desktop like filesystem for mobile devices and many mobile devices, e. g., such that use Windows Mobile, do provide such a storage mechanism already. Nevertheless, there are many devices on the market that are not able to store data in files but in proprietary formats. Most current mobile phones support Java's Mobile Information Device Profile (MIDP) [26] and are equipped with non-volatile flash memory. For security purposes J2ME-applications (so called MIDlets) must not save files to the file system directly. However, one can use MIDP's Record Management Systems (RMS) in order to store data permanently on such a device. The data is stored in the form of device dependent binary record stores but the API provides functions for accessing it device independently. An application can manage various record stores. Each of them is a collection of uniquely identified byte arrays. The ID is assigned incrementally starting with 1. IDs of deleted records are not reused. The RMS API allows MIDlets to add and remove records. Furthermore, it is possible to share records within the same application (see Fig. 1). Sharing data between applications is possible with MIDP 2.0.

A good starting point for working with RMS in MIDP 1.0 is the online article [24]. The MIDP 1.0 and MIDP 2.0 specifications are also available online [29; 30]. Besides storing data with the RMS directly, one can also use systems like Pointbase Micro or NanoBase [5] that are built on top of the Record Store Manager but support a higher level of abstraction. Unfortunately, in September 2007 IBM acquired DataMirror, the company that developed Pointbase Micro. The current status is that IBM will stop any support for the product at the end of September 2009 [15]. As NanoBase is new to the community, no release is available at the moment. However, the authors of the cited paper promised on request that they are still finishing the website of the system.

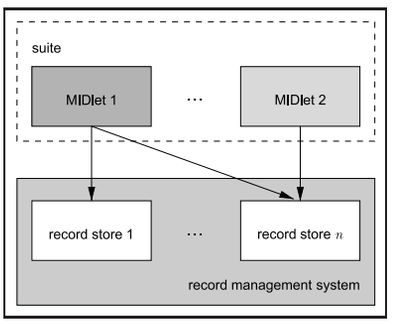

Figure 1 RMS structure.

Some newer mobile devices also support accessing PIM data or handling files through their Java Virtual Machine. However, not all manufacturers implement the respective optional Java Specification Requests (JSR 75) [31]. Unfortunately, there is no manufacturer independent list that summarizes which mobile devices support which JSR, and to focus on a certain manufacturer would be out of scope for this article. Hence, a developer has to carefully check system specifications before implementing certain functions.

3 Client/Server Approaches

Recent mobile information systems work mostly in a client/server-manner where clients are mobile and servers are static, powerful devices. Due to the unreliability of energy intensive wireless data transmission it is not possible to guarantee a permanent connection between server and clients. Therefore, data has to be copied to the mobile devices where it then can be used offline, too. One can find three major classes of approaches for managing such redundant data in a client server manner in the literature: caching, hoarding, and replication. The following questions need to be answered in order to determine the appropriate approach for a given application scenario [19].

- Is the data on the mobile device read only or mutable?
- Who decides what data is stored on the mobile device, the user or the system?
- Is the decision for storing data on the device a dynamic or a static decision?

Figure 2 illustrates, based on [10], various important aspects in which the three classes differ from each other:

The grade of possible data manipulations: Are updates allowed on the mobile device?

The possibility to work offline: Is all required data guaranteed to be on the mobile device?

The potential dynamic of data: How often does the data that is on the mobile device change?

The required resources: Which kind of software is required for a certain technique?

The influence on local data: Is the user able to specify the data needed on the mobile device?

We discuss these aspects based on [11] per technique in more details in the following sub-sections.

3.1 Caching

The easiest way to provide a mobile user with data locally is to cache data once it has been received. So, data is requested implicitly. Web and WAP browsers, for example, automatically cache WML or HTML pages and pictures. We know caching approaches from network based desktop applications as well as from database based information systems. The research on

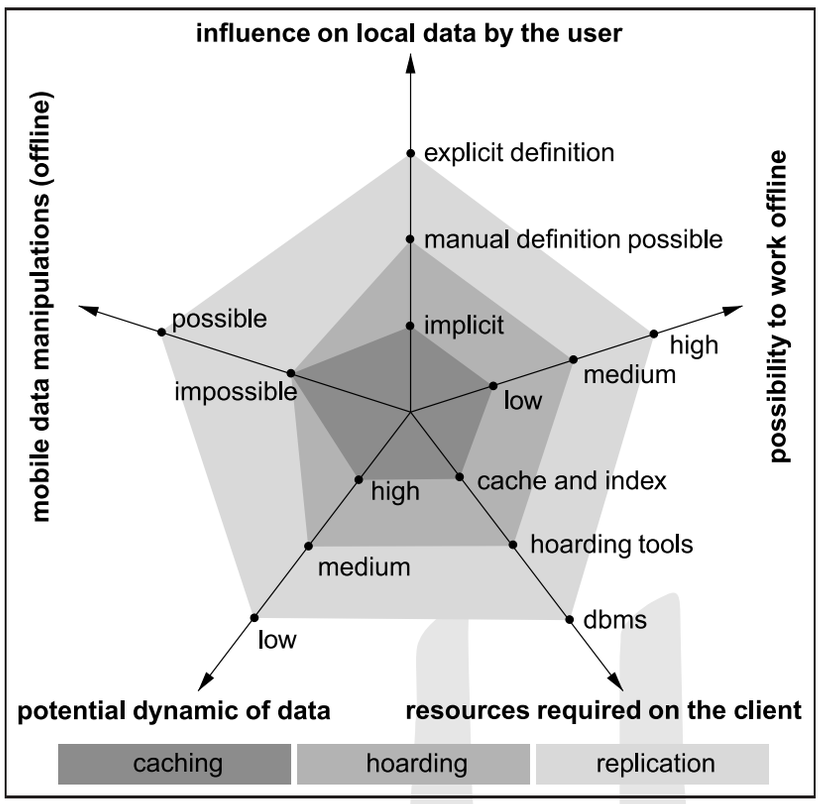

Figure 2 Classification of caching, hoarding, and replication.

caching in context of mobile information system focuses on semantic caching instead of caching pages/blocks or objects. Therefore, query results are stored as they are but indexed using the corresponding query [17; 22; 32]. Hence, it is possible to analyze whether new queries can be answered without communication with the server. To a certain degree it is also possible to recombine cached data with additional data to answer a query [6; 9; 34]. Unfortunately, the algorithms are complex and suffer from strict limitations of the underlying query containment problem (see [2; 3; 8; 18; 36]).

The availability of data in the cache can not be guaranteed as it depends on the used replacement strategy. If the cache is full, these algorithms decide about removing (hopefully) not longer required data. However, the decision might be wrong because it is typically based on the time, on which a data item is already in the cache, and/or on accesses on this data item. Semantic caches tend to use semantic correlations between different cached items as a replacement criteria [4]. Furthermore, it is possible to analyze contextual information such as location and to replace data because it does not longer belong to the current context.

Caches are primarily used for reducing costs and delays of data transmissions. However, they do not provide a full offline work with the information system. On the other hand caches are easy realizable and do not require additional functionalities such as synchronization or data mining that are required for replication or hoarding. Nevertheless, semantic caches need an index structure and algorithms for retrieving and recombining cached data. Another aspect of caching is that local data modifications are normally not supported. So, it is used in read only scenarios.

For mobile data caching of location-based services, location-dependent caching strategies can be applied. On the basis of semantic caching, the performance of location-dependent queries of mobile users can be improved by exploiting the movement patterns of the user. For example, in [33], the cache is organized in semantic segments that represent results of location-dependent queries. If data has to be deleted on the cache because of storage space, the *Furthest Away Replacement* (FAR) strategy is used. This approach extends the directional Manhattan distance function from [4] in three ways: firstly, it does not calculate the center of the segment, but uses the attached location information; secondly, FAR derives relationships between cached segments and the current query instead of calculating Manhattan distances using estimated weights; and finally, FAR uses knowledge about the movement direction of the mobile user. However, these approaches do assume that location-dependent data is queried only if the user is in that region – queries to that region from other locations are not supported well. Thus, Zheng et al. in [37] propose two other cache replacement policies called *Probability Area* (PA) and *Probability Area Inverse Distance* (PAID). Both policies use the concept of the valid scope of a data value, which is a geometric area that reflects the access probability to that data value from users in that area. While PA mainly relies on that concept, PAID further combines it with the data distance which is similar to the FAR strategy.

3.2 Replication

Replication approaches enable the possibility to work offline with the data stored on the mobile device. The user explicitly defines the replicas and copies the required data on the mobile device [7]. In fact, nearly each database management system vendor offers mobile DBMS that can be used in combination with a backend DBMS for replication purposes (e. g., Oracle Lite [27], IBM DB2 Everyplace [16], Microsoft SQL Server CE [25],

or Sybase Adaptive Server Anywhere [14]). The replication process depends on the underlying system architecture which could be a simple client/server approach or a middleware solution. A client/server architecture offers a direct access from the client to the server database. Middle-ware approaches use additional components (the middleware) that manage the communication between the server database(s) and mobile clients. Replicas are defined similar to (materialized) database views. Therefore, selection and projection operations are allowed. In order to allow updates to the replicated data, join operations are mostly not allowed. Besides the content definition, the user can also specify how and when the system shall synchronize updates. If the mobile device runs out of memory, the user has to free it manually by deleting replicated data.

The data replication strategies can also be adapted. If the mobile device is always connected to a costly network, it depends on the ratio between mobile access and update rate of a data item, whether it should be replicated on the mobile device or fetched on demand over the network. Huang et al. in [13] present and analysis various strategies based on these metrics.

3.3 Hoarding

The class of hoarding approaches covers aspects of caching and of replication. Similar to caches it uses a replacement strategy and implicitly stores data on the mobile device. In addition algorithms analyze the usage of data. So, necessary data that was not explicitly defined but is needed for working offline can be found and cached. Furthermore, some hoarding systems do also support manual specification of data that needs to be cached. Such techniques overlap with the replica specification. However, hoarding can be found mainly in form of hoarding file systems such as CODA [35] and its Seer extension [21]. This file system analyzes data accesses. If, e. g., the user compiles a C-programm, the source code as well as the involved header files are cached. For an information systems point of view hoarding can be realized via query rewriting such as harming select conditions or extending the projection list of database queries. Furthermore, contextual information might be used as hoarding criteria, e. g., location. For example, [28] or [20] propose systems that cache data with regard to the current or a predicted future location.

For example, [23] uses a prediction function based on Markov Chain Models to predict the movement of a mobile user. Data that is likely needed in the future by the user is pre-fetched into the cache. The prediction model deploys a regular-pattern detection algorithms to deal with both regular movement patterns (e. g., daily routine) and random parts. Even though the prediction algorithm is quite simple, simulations have shown a prediction efficiency of about 95%. Newer approaches like [20] use location-dependent log data about frequent queries of mobile users to build up so called probability maps. Using this information, the hoarding is not only adaptive to the movement of the users but also to changing data access patterns. Later, this work has been extended for hoarding web pages on mobile devices [1] using semantic clustering of related data items.

4 Summary and Conclusions

This paper was dedicated to the task of designing the data management of mobile application in the client/server paradigm. We have introduced approaches for storing and managing data on the mobile clients and presented a classification of approaches for managing redundant data that are necessary to face the characteristics of mobile devices and wireless networks. Furthermore, we discussed the specifics of the three classes caching, hoarding and replication and, therefore, gave a guideline for developers based on the characteristics of the targeted application.

The conclusion is: caching is easy to implement as it simply stores data implicitly. However, caches are not suitable if data availability needs to be guaranteed even if the device is not connected to an information provider. In this case replication is a must. Hoarding covers aspects of both, caching and replication and works perfectly in form of the CODA-file system with laptops. For more lightweight devices it is an option but not worthwhile. The additional effort that is needed for analyzing data accessed is much bigger then the effort for cache management and the user gets no data availability guarantees. To conclude this one can say: If the application has to guarantee data availability, the developer must use replication techniques. If the developer only wants to reduce data communications, caching is sufficient.

We explicitly left out peer-to-peer approaches and did not discuss context aware data management in detail. Both are interesting research topics. However, they are not common in today's applications or follow simple straight forward concepts.

References

[1] S. Bürklen, P. J. Marrón, K. Rothermel, and T. Pfahl. Hoarding location-based data using clustering. In: *Proc. of the Int'l Workshop on Mobility Management and Wireless Access*, pages 164–171, 2006.

[2] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In: *Proc. of the 4th Annual ACM Symp. on Theory of Computing*, pages 77–90, 1977.

[3] C. Chekuri and A. Rajaraman. Conjunctive Query Containment Revisited. In: F. N. Afrati and P. G. Kolaitis, editors, *Proc. of the 6th Int'l Conf. on Database Theory – ICDT '97*, vol. 1186 of *LNCS*, pages 56–70, 1997.

[4] S. Dar, M. J. Franklin, B. Jónsson, D. Srivastava, and M. Tan. Semantic Data Caching and Replacement. In: T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proc. of 22th Int'l Conf. on Very Large Data Bases (VLDB'96)*, pages 330–341, Sep 1996.

[5] L. Eloy, V. Vasconcelos, J. M. Monteiro, and Â. Brayner. NanoBase: A tiny relation database manager for the JME CLDC/MIDP Platform. *Revista Tecnologia (UNIFOR)*, 29(1):7–15, 2008. available online: http://www.unifor.br/notitia/file/2200.pdf.

[6] P. Godfrey and J. Gryz. Answering queries by semantic caches. In: T. Bench-Capon, G. Soda, and A. M. Tjoa, editors, *Database and Expert Systems Applications: Proc. of the 10th Int'l Conf. (DEXA'99)*, vol. 1677 of *LNCS*, pages 485–498, Sep 1999.

[7] C. Gollmick. Nutzerdefinierte Replikation zur Realisierung neuer mobiler Datenbankanwendungen. In: *Tagungsband der 10. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, no. P-26 in LNI, pages 453–462, Feb 2003.

[8] S. Guo, W. Sun, and M. A. Weiss. Solving satisfiability and implication problems in database systems. *ACM TODS*, 21(2):270–293, June 1996.

[9] H. Höpfner and K.-U. Sattler. Towards Trie-Based Query Caching in Mobile DBS. In: B. König-Ries, M. Klein, and P. Obreiter, editors, *Post-Proc. of the Workshop Scalability, Persistence, Transactions – Database Mechanisms for Mobile Applications*, no. P-43 in LNI, pages 106–121, Apr 2003.

[10] H. Höpfner, C. Türker, and B. König-Ries. *Mobile Datenbanken und Informationssysteme – Konzepte und Techniken*. dpunkt.verlag, July 2005.

[11] H. Höpfner. *Encyclopedia Of Database Technologies And Applications*, chapter Caching, Hoarding, and Replication in Client/Server Information Systems with Mobile Clients. IGI Global, 2nd edition, 2009. Accepted for publication, forthcoming.

[12] H. Höpfner and C. Bunse. Ressource Substitution for the Realization of Mobile Information Systems. In: J. Filipe, M. Helfert, and B. Shishkov, editors, *Proc. of the 2nd Int'l Conf. on Software and Data Technologie (ICSOFT 2007)*, vol. Software Engineering, pages 283–289, July 2007.

[13] Y. Huang, P. Sistla, and O. Wolfson. Data replication for mobile computers. In: *Proc. of the 1994 ACM SIGMOD Int'l Conf. on Management of Data*, pages 13–24, 1994.

[14] iAnywhere, A Sybase Company. *Adaptive Server Anywhere Datenbankadministration, Bestellnummer: DC03928-01-0902-01*, Oct 2004.

[15] IBM. DataMirror Product Offering Support Lifecycle. online, December 2008. http://www-01.ibm.com/support/docview.wss?uid=swg27012641&aid=1.

[16] IBM Corporation. *IBM DB2 Everyplace Application and Development Guide Version 8.2*, Aug 2004.

[17] A. M. Keller and J. Basu. A predicate-based caching scheme for client-server database architectures. *The VLDB Journal*, 5(1):35–47, Jan 1996.

[18] A. Klug. On conjunctive queries containing inequalities. *JACM*, 35(1):146–160, Jan 1988.

[19] B. König-Ries, C. Türker, and H. Höpfner. Informationsnutzung und -verarbeitung mit mobilen Geräten – Verfügbarkeit und Konsistenz. *Datenbank-Spektrum*, 7(23):45–53, 2007.

[20] U. Kubach and K. Rothermel. Exploiting location information for infostation-based hoarding. In: Ch. Rose, editor, *Proc. of the 7th Annual Int'l Conf. on Mobile Computing and Networking*, pages 15–27, 2001.

[21] G. H. Kuenning and G. J. Popek. Automated Hoarding for Mobile Computers. In: W. M. Waite, editor, *Proc. of the 16th ACM Symp. on Operating Systems Principles*, pages 264–275, 1997.

[22] K. C. K. Lee, H. V. Leong, and A. Si. Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(2):28–36, 1999.

[23] G. Y. Liu, A. Marlevi, and G. Q. Maguire. A mobile virtual-distributed system architecture for supporting wireless mobile computing and communications. *Wireless Networks*, 2(1):77–86, 1996.

[24] Q. Mahmoud. MIDP Database Programming Using RMS: a Persistent Storage for MIDlets. online article, December 2000. http://developers.sun.com/mobility/midp/articles/persist/.

[25] Microsoft Corporation. http://msdn.microsoft.com/library/, 2001.

[26] Sun Microsystems. Mobile Information Device Profile (MIDP). website, December 2008. http://java.sun.com/products/midp/.

[27] Oracle Corporation. *Oracle Database Lite, Administration and Deployment Guide 10g (10.0.0)*, June 2004.

[28] J. Peissig. guidePort – An Information and Guidance System. In: K. Kyamakya, editor, *WPNC 04 Proc.*, no. 0.1 in Hannoversche Beiträge zur Nachrichtentechnik, pages 1–17, Mar 2004.

[29] Java Community Process (JCP) Program. JSR 118: Mobile Information Device Profile 2.0. website, Dec 2008. http://jcp.org/en/jsr/detail?id=118.

[30] Java Community Process (JCP) Program. JSR 37: Mobile Information Device Profile for the J2ME™ Platform. website, Dec 2008. http://jcp.org/en/jsr/detail?id=37.

[31] Java Community Process (JCP) Program. JSR 75: PDA Optional Packages for the J2ME™ Platform. website, Dec 2008. http://jcp.org/en/jsr/detail?id=75.

[32] Q. Ren and M. H. Dunham. Using clustering for effective management of a semantic cache in mobile computing. In: S. Banerjee, P. K. Chrysanthis, and E. Pitoura, editors, *Proc. of the 1st ACM Int'l Workshop on Data Engineering for Wireless and Mobile Access*, pages 94–101, 1999.

[33] Q. Ren and M. H. Dunham. Using semantic caching to manage location dependent data in mobile computing. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 210–221. ACM, 2000.

[34] Q. Ren and M. H. Dunham. Semantic Caching and Query Processing. *Transactions on Knowledge and Data Engineering*, 15(1):192–210, Jan 2003.

[35] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel,

and D. C. Steere. Coda: A Highly Available File System for a Distributed Workstation Environment. *IEEE Trans. on Computers*, 39(4):447–459, Apr 1990.

[36] R. v. d. Meyden. The complexity of querying indefinite data about linearly ordered domains. *Journal of Computer and System Sciences*, 54:113–135, Feb 1997.

[37] B. Zheng, J. Xu, and D. L. Lee. Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments. *IEEE Trans. on Computers*, pages 1141–1153, 2002.


1


2


3

1 Dr.-Ing. Hagen Höpfner is Assistant Professor for Databases and Information Systems at the International University in Germany (Bruchsal, Germany). He received his Diplom in Computer Science and his Ph. D. in Computer Science in 2005 from the University of Magdeburg, Germany. His research interests include database theory, transaction processing as well as mobility aspects of databases and information systems in general. He is co-author of the German textbook on "Mobile Databases and Information Systems" and works as a freelance author for various German and international magazines.
Address: International University in Germany, School of Information Technology, Campus 2, 76646 Bruchsal, Germany, Tel.: +49-7251-700239, Fax: +49-7251-700250, E-Mail: hoepfner@acm.org

2 Dr. Essam Mansour is Research Associate for Databases and Information Systems at the International University in Germany (Bruchsal, Germany). He graduated from Cairo University, Egypt in 2000 with a First Class Honours in Information Systems. He then completed a M. Sc. in Information Systems in 2003 at the same university. He has received his Ph. D. in Computer Science in 2008 from the Dublin Institute of Technology (DIT), Ireland. His research interests include mobile databases and information systems, context aware systems, temporal XML data management, and active database systems.
Address: International University in Germany, School of Information Technology, Campus 2, 76646 Bruchsal, Germany, Tel.: +49-7251-700240, Fax: +49-7251-700250, E-Mail: essam.mansour@i-u.de

3 Prof. Dr. Daniela Nicklas is Junior Professor for database- and internet technology at the Department for Computer Science, University of Oldenburg. She received her Dr. rer. nat. (Ph. D.) 2005 in Computer Science from the University of Stuttgart, working on spatial context models and architectures for location-based applications, and her Diplom (MSc) 2000, also in Computer Science, with an emphasis on Software Engineering. Her current research interests are mobile applications, context-aware computing and data stream technologies.
Address: University of Oldenburg, Department for Computer Science, Escherweg 2, 26121 Oldenburg, Germany, Tel.: +49-441-9722211, Fax: +49-441-9722202, E-Mail: dnicklas@acm.org