The Ubiquitous DBMS

Kyu-Young Whang[1], Il-Yeol Song[2], Taek-Yoon Kim[1], and Ki-Hoon Lee[1]

[1] Department of Computer Science, KAIST, Daejeon, Korea, {kywhang, tykim, khlee }@mozart.kaist.ac.kr

[2] College of Information Science and Technology, Drexel University, Philadelphia, USA, songiy@drexel.edu

ABSTRACT

Advancement in mobile computing technologies has prompted strong needs for database systems that can be used in small devices such as sensors, cellular phones, PDAs, car navigators, and Ultra Mobile PCs (UMPCs). We term the database systems that are customizable for small computing devices as *Ubiquitous Database Management Systems* (*UDBMSs*). In this paper, we first review the requirements of the UDBMS. The requirements identified include lightweight DBMSs, selective convergence, flash-optimized storage systems, data synchronization, support of unstructured/semi-structured data, complex database operations, self-management, and security. Next, we review existing systems and research prototypes. We review the functionality of UDBMSs including the footprint size, support of standard SQL, transaction management, concurrency control, recovery, indexing, and access control. We then review the supportability of the requirements by those UDBMSs surveyed. We finally present research issues related to the UDBMS.

1. INTRODUCTION

The growing popularity of mobile technologies and advancement in computing power have prompted strong needs for database systems that can be used in small computing devices such as sensors, smart cards, cellular phones, PDAs, car navigators, and Ultra Mobile PCs (UMPCs). These small devices with mobility and embedded processors are called *ubiquitous devices* [WH04]. As the ubiquitous devices get computationally powerful and the bandwidth of the wireless network rapidly expands, we can use them to perform tasks anytime and anywhere often downloading a variety of data from servers and uploading sensor data to servers. This kind of computing environment is commonly called the *ubiquitous environment*. New storage devices suitable for ubiquitous devices such as flash memory [GT05] and MEMS (Micro-Electro-Mechanical Systems)-based storage devices [SG04] have been developed. As the capacity of the storage devices is getting bigger and bigger, we can easily store and manage a huge amount of data in a ubiquitous device. This trend prompted strong needs for the database systems that can be used in ubiquitous devices. We term the database systems customizable for small computing devices as *Ubiquitous Database Management Systems* (*UDBMSs*).

Representative *ubiquitous devices* include sensors, smartcards, cellular phones, PDAs, car navigators, and UMPCs. Sensors and smartcards are extremely small in size and have low computing power. A sensor is as small as a coin and is used for gathering data from its surrounding environment and for processing the data. A smartcard embeds a CPU, memory, and storage device for storing and managing data. The other devices are small in size but have high computing power. A cellular phone is used for managing personal information and playing multimedia data. A car navigator is used for finding the shortest way from the current location to the destination. A PDA is used for personal business management and for supporting applications such as an e-mail client, word processor, and spreadsheet. A UMPC is a general purpose PC but much smaller than laptops. Various applications that used to be run on a server can now be run on a UMPC.

The primary *storage* of ubiquitous devices is flash memory. Flash memory is non-volatile and has many advantages over the disk. Since the capacity of flash memory is increasing and the cost decreasing, flash memory will be widely used also in PCs and servers. Another type of storage media for ubiquitous devices is MEMS-based storage devices. A MEMS-based storage device is a secondary storage device and also has many advantages over the disk. Currently, there are some prototypes of MEMS-based storage devices but no products are available yet.

Ubiquitous devices usually have a limited storage capacity. Hence, users store bulk of data in the server and download the necessary parts to the ubiquitous devices. In this environment, when the data are modified in the ubiquitous device, the data need to be transmitted back and stored at the server to maintain consistency of data between them. This issue is called *data synchronization* [IBM06].

As the applications for ubiquitous devices are diverse, many *different types of data* need to be handled. The data types that need to be supported in ubiquitous devices include text data, web pages, XML data, spatial data, multimedia data, and sensor/stream data. E-mail clients, word processors, and spreadsheet applications manage text data. Web browsers manage web pages

and XML data. Car navigation systems manage spatial data. Image viewers, MP3's, and movie players manage multimedia data such as JPEG, MP3, and AVI files. A sensor transmits the data sensed to the server as a stream.

To support the ubiquitous environment, the UDBMS needs to be able to be deployed to different types of ubiquitous devices and to be able to support various applications. In addition, the UDBMS should support new types of storage devices, different types of data, data synchronization, self-management, and security.

There are some research prototypes and commercial products of UDBMSs. Representative research prototypes are TinyDB [FHM07], PicoDBMS [ABP07, ABPV08], and Odysseus/Mobile [1]. Representative commercial products are IBM DB2 Everyplace [IBM06, IBM08], Oracle 10g Lite [Orac06], Oracle Berkeley DB [Orac08, SO07], and Microsoft SQL Server CE (Compact Edition) [DS07, MS05]. TinyDB and PicoDBMS have been developed for extremely small devices with low computing power. TinyDB runs on sensors, and PicoDBMS on smartcards. The other DBMSs have been developed for small devices with high computing power such as cellular phones, PDAs, and UMPCs.

Research on UDBMSs is still at an initial stage. Research groups and commercial DBMS vendors have mainly focused on shrinking the footprint size but have not actively dealt with other important problems such as managing various types of data and supporting flash memory and MEMS-based storage devices.

The rest of this paper is organized as follows. In Section 2, we present important requirements of the UDBMS. In Section 3, we survey the functionalities of existing UDBMSs and analyze how well they satisfy the requirements identified. In Section 4, we introduce research issues related to the UDBMS and present the current status and future work of each issue. Finally, in Section 5, we summarize this paper.

2. REQUIREMENTS OF THE UDBMS

In this section, we present important requirements of the UDBMS. Nori [Nori07] has presented a broad survey of mobile and embedded DBMSs. Bernhard et al. [BBB+04] have presented open issues and research topics on mobile DBMSs. In this paper, we consider mobile and embedded DBMSs as UDBMSs. Based on existing surveys [BBB+04, Nori07, Whan07], we identify important requirements for the UDBMS as follows.

[1] Odysseus/Mobile is the ubiquitous version of the Odysseus DBMS [WLK+09, WLL+05, WPHL02].

- **Lightweight DBMSs**

Table 1 shows a summary of typical specifications of ubiquitous devices in 2008. As shown in Table 1, ubiquitous devices have lower computing power than PCs or servers.

Table 1. Typical specifications of ubiquitous devices in 2008.

Ubiquitous Devices	CPU Clocks	Main Memory Sizes	Storage Sizes
Sensors	7 MHz	0.5 ~ 8 KBytes	8 ~ 128 KBytes
Smartcards	14 MHz	4 KBytes	128 KBytes
Cell Phones	300 MHz	64 MBytes	128 MBytes
PDAs	624 MHz	128 MBytes	256 MBytes
Car Navigators	1 GHz	256 MBytes	16 GBytes
UMPCs	1.3 GHz	1 GBytes	80 GBytes

Using a low-clock CPU, small memory, and small storage, a UDBMS needs to support the functionalities required by applications with acceptable performance. Furthermore, since ubiquitous devices have limited power sources such as batteries, a UDBMS needs to support the functionalities with low power consumption. Thus, it is important to design and implement a UDBMS as simple as possible considering the performance of devices [BBB+04, Nori07]. We also need to consider co-design of hardware and software [ABPV08, BBB+04], specifically, for devices with tight hardware constraints such as sensors and smartcards.

- **Selective Convergence**

To support the lightweight DBMS requirement, it is important to selectively compose the modules of a UDBMS depending on the applications and the device type. In order to emphasize the capacity that selects only necessary modules, we call this property "*selective convergence*" [Whan07]. Selective convergence is a notion that contrasts to extensibility [SAH87] in that it requires all the functionalities be implemented apriori and then customized according to individual needs while extensibility requires new functionalities be easily implemented for added features. For low performance devices such as sensors and smartcards, users would want only simple and basic functionalities. In contrast, for high performance devices such as PDAs and UMPCs, users would want advanced functionalities such as data synchronization. For example, if a user wants to run a GIS application in his PDA, the user would want spatial functionalities. A similar notion has been introduced by Nori [Nori07] as "componentization."

- **New Storage Devices**

For ubiquitous devices, non-volatile memories (e.g., flash, EEPROM, and FeRAM) and very small

secondary storage devices (e.g., MEMS) have many advantages over the disk.

Flash memory is a representative non-volatile memory. Compared with the disk, flash memory has attractive features such as small size, better shock resistance, lower power consumption, fast access time, and no mechanical seek and rotational latency [KV08]. Besides, there is an erase operation, which does not exist in the disk. In order to update existing data in a page, an erase operation should be performed first on the entire block to which the page belongs. The erase time is about ten times slower than the write time, and the number of erase operations is limited to 100,000 ~ 1,000,000 times [GT05, NG08].

A MEMS-based storage device is a very small non-volatile secondary storage. The size is as small as $1cm^2$, and the average access time is ten times faster than that of the disk with lower power consumption. The MEMS device is composed of a media sled and a probe tip array. The *media sled* is a square plate on which data are recorded, and the *probe tip array* is a set of heads. The MEMS device reads and writes data by moving the media sled in the direction of both X and Y axes. By selecting and activating a portion of the heads simultaneously, users can access multiple data sectors in parallel [GSGN00, SG04].

The storage system of the traditional DBMSs has been developed for the disk but not for new storage devices such as flash memory or MEMS devices. To fully exploit the advantages of the new storage devices, we need to develop new storage systems optimized for them [Nori07]. For flash memory, data update time can be optimized by considering overhead caused by updates and the number of erase operations. For MEMS devices, data access time can be optimized by considering data placement and movements of heads.

- **Complex Operations for Advanced Applications**

Advanced database applications require complex operations such as data mining that cannot be implemented by SQL. As ubiquitous devices are rapidly evolving, many advanced applications that have been used in servers will also be required of ubiquitous devices. Thus, a UDBMS should be able to support complex operations.

For example, on PDAs or UMPCs, we foresee the need for running advanced search applications. For example, there may be an application that finds documents similar to a given document. The application needs complex search operations that analyze relationships and similarity between the given document and the stored documents.

In a u-health care environment, we may need advanced applications for patient management. U-health means an anytime and anywhere medical service [IBM09]. In a u-health care environment, sensors are attached to the patient and gather and send the patient's health data to doctors' PDAs. Using PDAs, doctors can check their patients' conditions and decide prescriptions for them anytime and anywhere. As the computing power of the PDA increases, there may be advanced applications that find a patient's walking pattern and abnormal symptoms. The applications need a complex operation that finds meaningful or frequent patterns from a patient's health data.

- **Unstructured and Semi-structured Data**

Various types of unstructured/semi-structured data such as text, multimedia, XML, spatial, stream, and sensor data are widely used in database applications. Those unstructured/semi-structured data are important not only in servers but also in ubiquitous devices. Examples are lyrics data in MP3 players, map data in car navigators, and multimedia data in PDAs. Thus, efficient management and search of unstructured/semi-structured data will also be required of the UDBMS.

- **Data Synchronization**

Ubiquitous devices cannot stay connected to the server all the time due to limited power resources and unstable wireless connection. Furthermore, ubiquitous devices are not able to store a large amount of data due to lack of storage capacity. Thus, users store bulk of data in the server and download only the necessary parts of the data from the server to the ubiquitous device.

In this environment, many users can share and manage data by replicating the same data in the server to their own devices. For example, in a hospital, bulk of patients' data is stored in the server database, and doctors replicate a portion of these data to their PDAs. In this way, many doctors can share and manage the same patient's data anytime and anywhere. As a result, different versions of the data may exist and result in data inconsistency. Therefore, a UDBMS needs to support a functionality to integrate different versions of data into a consistent version [BBB+04, Nori07], i.e., data synchronization.

- **Self-Management**

Unlike in traditional DBMSs, in a UDBMS, there can be no database administrator (DBA) to manage the database. Thus, a UDBMS needs to support self-management functionalities. That is, it needs to automatically perform operations like backup, restore, recovery, indexing, and tuning [Nori07].

- **Security**

Since ubiquitous devices often contain personal data such as banking and healthcare data, a UDBMS needs

to ensure the data security by providing access control policies [ABPV08].

- **Longevity and Distributed Processing**

In a wireless sensor network, several thousands of sensor nodes with limited resources, e.g., battery power, are connected through the network to the server (or base station). The server sends queries into the sensor network, and sensor nodes collect, filter, aggregate, and route data back to the server. Thus, a UDBMS in a sensor node needs to support distributed query processing and to minimize power consumption for longevity [MFHH05].

3. EXISTING SYSTEMS

In this section, we survey representative research prototypes and commercial products of the UDBMS and compare their functionalities. Then, we review the supportability of the requirements identified in Section 2 for the UDBMSs surveyed.

3.1 Representative Systems

Some research groups and commercial DBMS vendors have developed UDBMSs. Table 2 shows the research prototypes and commercial products we surveyed. Commercial products include IBM DB2 Everyplace, Oracle 10g Lite, Oracle Berkeley DB, and Microsoft SQL Server CE. Oracle Berkeley DB has been developed by University of California, Berkeley, but its license has moved to Oracle corp.

Table 2. Research prototypes and commercial products of the UDBMS.

Research Prototypes	Commercial Products
TinyDB, PicoDBMS, Odysseus/Mobile	IBM DB2 Everyplace, Oracle 10g Lite, Oracle Berkeley DB, MS SQL Server CE

Research prototypes include TinyDB, PicoDBMS, and Odysseus/Mobile. TinyDB has been developed at University of California, Berkeley. PicoDBMS has been developed at University of Versailles and INRIA. Odysseus/Mobile is the ubiquitous version of the Odysseus DBMS [WLK+09, WLL+05, WPHL02] that has been continually evolving for the last 19 years at KAIST. Odysseus DBMS is tightly coupled with information retrieval (IR) and spatial database functionalities.

Odysseus/Mobile supports all the functionalities of the Odysseus DBMS and additionally supports selective convergence, data synchronization, and the flash-optimized storage system (ongoing) for ubiquitous devices. Odysseus/Mobile supports selective convergence through the architecture that allows users to choose necessary modules at compile time.

Existing UDBMSs can be categorized by target devices in which the DBMS is deployed. Table 3 shows the summary. In Table 3, it seems that sensors, smartcards, and high performance devices are different enough to justify different DBMSs. However, we expect that the difference will become less obvious as device technology evolves, and UDBMSs that support the requirements in Section 2.1 will be needed for all those devices. For example, even in sensors, complex operations such as data mining will be needed to detect and filter out outliers in sensed data.

Table 3. Existing UDBMSs categorized by target devices.

Target Devices		UDBMSs
Extremely Small Devices with Low Computing Power	Sensors	TinyDB
	Smartcards	PicoDBMS
Small Devices with High Computing Power	Cell Phones, PDAs, Car Navigators, and UMPCs	IBM DB2 Everyplace, Oracle 10g Lite, Oracle Berkeley DB, MS SQL Server CE, Odysseus/Mobile

3.2 Functional Analysis

Table 4 shows the functionalities of existing UDBMSs. General functionalities of server DBMSs include SQL query processing, views, integrity constraints, transaction management, concurrency control, recovery, indexing, access control, encryption, and compression[2]. Existing UDBMSs support only essential functionalities among those of server DBMSs.

TinyDB supports only essential functionalities for sensor applications. Since most of the sensor applications are used to filter out some data, they just need the functionality that selects data satisfying given conditions. Thus, TinyDB supports only SELECT statements of TinySQL[3], and its footprint is extremely small—only 3 KBytes. For more information on TinyDB, refer to Madden et al. [FHM07].

PicoDBMS supports sufficient functionalities for smartcard applications. Smartcard applications are used for data management such as insert, delete, update and search. Thus, PicoDBMS supports a part of SQL such as INSERT, UPDATE, DELETE, SELECT (with join and aggregation), and CREATE/DROP TABLE statements. Additionally, it supports CREATE/DROP VIEW and GRANT/REVOKE statements. PicoDBMS

[2] Memory consumption is also important, but we excluded since it is not published by commercial vendors or research groups.

[3] TinySQL supports a very limited part of SQL99 such as INSERT, UPDATE, SELECT, and CREATE/DROP TABLE statements.

also supports functionalities for indexing, transaction management, recovery, access control, and compression. The footprint size of PicoDBMS is about 30 KBytes, larger than TinyDB. For more information on PicoDBMS, refer to Bobineau et al. [ABP07, ABPV08].

Oracle Berkeley DB, Oracle 10g Lite, IBM DB2 Everyplace, Microsoft SQL Server CE, and Odysseus/Mobile support most of the functionalities of server DBMSs since their target devices, such as cellular phones, PDAs, car navigators, and UMPCs, have a lot more computing power than sensors and smartcards. These DBMSs commonly support views, transaction management, concurrency control, recovery, and indexing. All the above-mentioned DBMSs, except Oracle Berkeley DB, support a part of the SQL99 standard, which is broader than that of PicoDBMS. Oracle Berkeley DB, however, uses proprietary APIs instead of SQL. The footprint sizes of these DBMSs range from 350 KBytes to 2.5 MBytes.

There is no existing UDBMS that supports all the requirements of the UDBMS. Table 5 shows the supportability of the requirements for each product we reviewed. The UDBMSs that support selective convergence are Oracle Berkeley DB and Odysseus/Mobile. In these UDBMSs, users can choose functionalities such as concurrency control, recovery, and indexing at the compile time. None of the existing UDBMSs support the flash-optimized storage system and complex database operations for advanced applications. The UDBMSs that support self-management are TinyDB, PicoDBMS, and Oracle BerkeleyDB. For Oracle 10g Lite and MS SQL Server CE, support of self-management is not clearly known. All the existing UDBMSs except TinyDB and Oracle BerkeleyDB support security. The UDBMSs that support the data synchronization functionality are Oracle Berkeley DB, Oracle 10g Lite, IBM DB2 manages XML data, and Odysseus/Mobile manages XML, text, and spatial data. In these DBMSs, management of unstructured/semi-structured data can also be selectively converged to the DBMS.

4. RESEARCH ISSUES

In this section, we present five research areas that are important for satisfying the requirements of the UDBMS. We present the current status of each research issue and propose future work.

4.1 Lightweight DBMSs

Commercial UDBMSs have shrunken the footprint size and memory usage by simplifying the functionalities of server DBMSs, but commercial vendors have not reported the techniques they adopted. The representative research work that deals with lightweight DBMS techniques is PicoDBMS [ABP07, ABPV08]. The paper proposes a new storage system and a new query processor for smartcards, which have a very small storage and memory. The storage system uses a pointer-based storage model where tuples reference their attribute values by means of pointers to preclude any duplicate value. In addition, considering small memory of smartcards, the paper presents techniques for processing queries with a limited memory capacity.

As the ubiquitous environment evolves, many advanced applications that have been used in servers will also be required of ubiquitous devices. However, even high performance devices such as PDAs and UMPCs still lack computing power to run a fully-fledged DBMS that supports as many functionalities as server DBMSs do. Thus, selective convergence will become a very important technique since it enables lightweight DBMSs by selectively composing the modules of a fully-fledged DBMS. We expect research on designing and implementing a new DBMS architecture that supports selective convergence will soon become active.

4.2 Storage Systems for New Storage Devices

Research on flash memory has been actively conducted in the field of operating systems (OS) [Alep02, Ban99, Wood01], and recently, in the field of databases [LM07, LMP+08]. The main goal of the research is to optimize the data update cost considering the cost of the erase operation of flash memory. In flash memory, the unit of read/write operations is a page, but the unit of the erase operation is a block, which consists of multiple pages.

In the early days of flash memory, update of an existing page in a block is performed by the method called *in-place update* [GT05]. In the in-place update method, the updated page is overwritten into the original location of the page. The method consists of the following four steps: (1) read all the pages of the block into memory, (2) update the page, (3) erase the block in the storage, and (4) write all the pages in memory to the erased block. This method has an overhead of reading and writing the entire block and needs an expensive erase operation. To solve these problems, the *out-place update* method [GT05] has been proposed. This method writes the updated page into an empty page and

Table 4. Functionalities of the existing UDBMSs.

	TinyDB	PicoDBMS	Oracle Berkeley DB	Oracle 10g Lite	IBM DB2 Everyplace	MS SQL Server CE	Odysseus/ Mobile
Minimum Code Footprint Size	3 KBytes	30 KBytes	500 KBytes (embedded Linux, storage system only)	970 KBytes (Windows)	350 KBytes (embedded Linux)	2 MBytes (Windows)	2.5 MBytes (embedded Linux) 410 KBytes (embedded Linux, storage system only)
SQL	SELECT only	a part of SQL99	N	a part of SQL99	a part of SQL99	a part of SQL99	a part of SQL99
Views	N	Y	N	Y	Y	Y	Y
Integrity Constraints	N	N/A	N	Y	Y	Y	N
API	TinyDB API	JDBC	Berkeley DB API	JDBC, ODBC, ADO.NET, SODA API	DB2 CLI, JDBC, ODBC, ADO.NET API	ADO.NET API	OOSQL CLI, JDBC, ODBC, PHP, Python API
Transaction Management[4]	N	Y	Y	Y	Y	Y	Y
Concurrency Control	N	N	Y	Y	Y	Y	Y
Indexing	N	Y	Y	Y	Y	Y	Y
Access Control	N	Y	N	Y	Y	Y	Y
Encryption	N	N/A	Y	Y	Y	Y	N
Compression	N	Y	N/A	Y	Y	Y	text only

Table 5. Supportability of the requirements of the UDBMS.

	TinyDB	PicoDBMS	Oracle Berkeley DB	Oracle 10g Lite	IBM DB2 Everyplace	MS SQL Server CE	Odysseus/ Mobile
Lightweight DBMS	Y	Y	Y	Y	Y	Y	Y
Selective Convergence	N	N	Y	N	N	N	Y
Flash-optimized Storage System	N	N	N	N	N	N	under development
Complex Operations	N	N	N	N	N	N	N
Unstructured/Semi-structured Data	N	N	XML only	N	N	N	text, spatial, XML
Data Synchronization	N	N	Y	Y	Y	Y	Y
Self-Management	Y	Y	Y	N/A	Y	N/A	N
Security	N	Y	N	Y	Y	Y	Y

invalidates the original page. The invalidated pages are erased at once in a batch fashion.

There are two active research topics on flash-memory management—the flash translation layer (FTL) and the flash-optimized storage system. FTL is a layer that emulates a disk using flash memory. Using the FTL, most of commercial OS file systems and UDBMSs with a disk-based storage system are able to run on flash memory [Ban99]. The emulation approach, however, has a disadvantage that it is hard to achieve the best performance since the storage system indirectly manages flash memory. The flash-optimized storage system directly manages flash memory without the FTL and is able to achieve the best performance. It has been developed by adapting the data placement method of the log-structured file system to flash memory [GT05]. Representative flash-optimized OS file systems are JFFS [Wood01] and YAFFS [Alep02]. For the flash-based storage system of DBMSs, a prototype called LGeDBMS [KBL+06] and data update methods [KWS10, LM07] have been proposed.

The specificity of considering flash memory in small devices such as smartcards is that they could have extremely small main memory. Recently, there has been research on flash (or EEPROM) storage system for those devices. Yin et al. [YPM09] and Bolchini et al. [BSST03] have addressed the problem of adapting data structures and algorithms to extremely small main memory considering the hardware constraints of flash (or EEPROM) devices.

[4] In general, transaction management includes concurrency control. However, in a single user environment like PicoDBMS, concurrency control is not needed for transaction management.

Research on the MEMS-based storage device is still at an initial stage. Data placement methods considering the movement of headers have been proposed. Similar to flash memory, there are methods proposed to emulate the disk using the MEMS-based storage device [DM03, GSGN00] and those to develop the MEMS-optimized storage system by directly controlling the device [KWKS09, YAA07].

Research on utilizing flash memory and MEMS-based storage devices for DBMSs will become more active. Especially, as the capacity of flash memory per unit cost increases, flash memory is expected to be widely used as storage devices not only for ubiquitous devices but also for PCs and servers. There are many challengeable optimization problems considering the characteristics of flash memory—including optimization of index structures, buffer management, recovery, query processing, and sorting methods. Adaptation of data structures and algorithms to an extremely small main memory environment is also an important issue.

4.3 Data Synchronization

In a mobile environment, data synchronization is a very important issue. However, research on data synchronization between the UDBMS and the server has been rare. Representative synchronization modules are the sync server of IBM DB2 Everyplace [IBM06], the mobile server of Oracle 10g Lite [Ora06], and the active sync of Microsoft SQL Server CE [DS07]. Commercial synchronization solutions consist of client databases on ubiquitous devices, a synchronization server, and a server database. The synchronization server controls the consistency of replicated data in the client and server databases.

Main research issues on data synchronization are (1) efficiently maintaining data synchronization between a huge number of ubiquitous devices and the server, (2) resolving conflicts when there are different versions of the same data among ubiquitous devices and the server, (3) recovering from crash and restarting data synchronization when system failure occurs during data synchronization.

4.4 Unstructured/Semi-structured Data

Research on managing unstructured/semi-structured data has been active in the context of server DBMSs and will be also important in the context of UDBMSs. Among various types of unstructured/semi-structured data, text, XML, stream, and spatial data have been actively studied. Recently, DB-IR integration [ACR+05, WPHL02] has become a subject of active research.

For server DBMSs, commercial vendors and research groups have developed database systems for managing unstructured/semi-structured data. For XML data, vendors have released extended versions of their DBMSs that support XML data. Research groups have also published on prototypes such as MonetDB/XQuery (CWI) [BGK+06], dbXML (dbXML Group) [dbXML08], and Odysseus/XML (KAIST) [HLL03, LKWL06]. For stream data, many prototypes have been developed. Representative prototypes are TelegraphCQ (University of California, Berkeley) [CCD+03], NiagaraCQ (University of Wisconsin) [CDTW00], and STREAM (Stanford University) [ABB+03]. For text data and spatial data, commercial vendors and research groups have released their products or prototypes. The Odysseus DBMS supports managing text and spatial data in a tightly-coupled fashion [WLK+09, WLL+05, WPHL02]. An early version of the Odysseus DBMS has been used (1997-2000) for the Naver search engine of NHN Co., which is currently the best portal in Korea.

There are two approaches to support unstructured/semi-structured data. One is the loose coupling approach, and the other is the tight coupling approach. Commercial vendors use the loose coupling approach. In the loosely-coupled architecture, the functionality of managing the data is implemented using the DBMS API on top of the DBMS engine. Thus, the loosely-coupled architecture incurs overhead caused by the high-level (typically, SQL-level) API calls between the unstructured/semi-structured data management module and the DBMS engine. In contrast to commercial vendors, the Odysseus DBMS uses the tight coupling approach. In the tightly coupled architecture, the functionality of managing unstructured/semi-structured data is integrated into the DBMS engine [WLK+09, WLL+05, WPHL02]. The tight coupling architecture eliminates the overhead caused by the high-level API calls—obtaining high performance.

Since ubiquitous devices (including PDAs and UMPCs) lack computing power to run a fully-fledged DBMS that supports unstructured/semi-structured data, research on developing a lightweight version of the unstructured/semi-structured data management module considering the specification and performance of the ubiquitous devices needs to be conducted. For example, in a fully-fledged DBMS, one could implement many existing query processing methods and combine them under a cost-based optimization framework so as to select the best query evaluation plan. However, since ubiquitous devices have small memory and storage, it may not be feasible to include many query processing methods and the optimization module in the UDBMS. Thus, we need to study query processing methods that show robust and predictable performance in a lightweight environment where cost-based optimization is not available [MVT05].

4.5 Data Mining for Complex Database Operations

Advanced applications perform complex operations such as finding relationships, trends, and meaningful patterns of data. Moreover, target data of the operations are not only relational but also unstructured/semi-structured. Those operations are hard to be implemented with SQL only, but require more advanced techniques such as data mining, which have been a useful tool for supporting complex operations on relational data as well as unstructured/semi-structured data.

In the context of server DBMSs, there has been active research on data mining for various types of unstructured/semi-structured data [HK05]. Examples are text data mining, web page mining, spatial/temporal data mining, stream data mining, multimedia data mining, and bioinformatics data mining. Currently, active research is in progress on adapting traditional mining techniques for relational data to unstructured/semi-structured data [HK05].

Research on data mining will be expanded to the context of the UDBMS. For advanced applications such as advanced search and u-health care mentioned in Section 2, data mining techniques are very useful in implementing complex operations needed for them. For example, in simple filtering applications for the sensor devices, the sensed data are filtered out by checking whether the data are compatible with given arithmetic conditions such as larger-than, less-than, or equal-to; in advanced filtering applications, however, more complex filtering operations such as detecting outliers can be executed by discovering overall trends of the sensed data [SLMJ07]. Considering the specification and performance of ubiquitous devices, research on developing a lightweight version of mining techniques needs to be conducted.

5. SUMMARY

In this paper, we have introduced the concept of the UDBMS and related research issues. We have defined the UDBMS as a customizable database system for small computing devices. We have identified important requirements of the UDBMS such as lightweight DBMSs, selective convergence, flash-optimized storage systems, data synchronization, support of unstructured/semi-structured data, complex database operations, self-management, and security. We then have reviewed the functionalities of existing UDBMSs and research prototypes. We have also discussed how well those existing UDBMSs and research prototypes support the requirements identified. We have found that, as of now, there is no UDBMS that supports all the requirements. Especially, flash-optimized storage systems and complex database operations are not supported by any UDBMS. Finally, we have presented research issues related to the UDBMS for each category of the requirements identified.

6. ACKNOWLEDGEMENTS

This work was supported by the National Research Lab orogram (No. 2009-0083120) through National Research Foundation (NRF) of Korea funded by Ministry of Education, Science, and Technology [Whan07]. We deeply appreciate anonymous reviewers' incisive comments on the earlier version that made this paper more complete and readable.

7. REFERENCES

[ABB+03] Arasu, A. et al., "STREAM: The Stanford Stream Data Manager," *IEEE Data Eng. Bull.*, 26(1), pp. 19-26, 2003.

[ABP07] Anciaux, N., Bouganim, L., and Pucheral, P., "Future Trends in Secure Chip Data Management," *IEEE Data Eng. Bull.*, 30(3), pp. 49-57, 2007.

[ABPV08] Anciaux, N., Bouganim, L., Pucheral, P., and Valduriez, P., "DiSC: Benchmarking Secure Chip DBMS," *IEEE TKDE*, 20(10), pp. 1363-1377, 2008.

[ACR+05] Amer-Yahia, S., Case, P., Rolleke, T., Shanmugasundaram, J., and Weikum, G., "Report on the DB/IR Panel at SIGMOD 2005," *SIGMOD Record*, 34(4), pp. 71-74, 2005.

[Alep02] Aleph One Ltd., "YAFFS: Yet Another Flash Filing System," http://www.yaffs.net, 2002.

[Ban99] Ban, A., Flash File System Optimized for Page-Mode Flash Technologies, US patent 5,937,425, 1999.

[BBB+04] Bernard, G. et al., "Mobile Databases: a Selection of Open Issues and Research Directions," *SIGMOD Record*, 33(2), pp. 78-83, 2004.

[BGK+06] Boncz, P. et al., "MonetDB/XQuery: a Fast XQuery Processor Powered by a Relational Engine," In *SIGMOD*, pp. 479-490, 2006.

[BSST03] Bolchini, C. et al., "Logical and Physical Design Issues for Smart Card Databases," *TOIS*, 21(3), pp. 254-285, 2003.

[CCD+03] Chandrasekaran, S. et al., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," In *CIDR*, pp. 269-280, 2003.

[CDTW00] Chen, J. et al. "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," In *SIGMOD*, pp. 379-390, 2000.

[dbXML08] dbXML 2.0, http://www.dbxml.com, 2008.

[DM03] Dramaliev, I. and Madhyastha, T., "Optimizing Probe-Based Storage," In *FAST*, pp. 379-390, 2003.

[DS07] Dhingra, P. and Swanson, T., *Microsoft SQL Server 2005 Compact Edition*, Sams, 2007.

[FHM07] Franklin, M. J., Hellerstein, J. M., and Madden S., "Thinking Big About Tiny Databases," *IEEE*

Data Eng. Bull., 30(3), pp. 37-48, 2007.

[GSGN00] Griffin, J. L., Schlosser, S. W., Ganger, G. R., and Nagle, D. F., "Operating Systems Management of MEMS-Based Storage Device," In *OSDI*, pp. 227-242, 2000.

[GT05] Gal, E. and Toledo, S., "Algorithms and Data Structures for Flash Memories," *Computing Surveys*, 37(2), pp. 138-163, 2005.

[HK05] Han, J. and Kimber, M., "Data Mining—On What Kind of Data?," In Book *Data Mining: Concepts and Techniques*, 2nd ed., Morgan Kaufmann, 2005.

[HLL03] Han, W., Lee, K., and Lee, B., "An XML Storage System for Object-Oriented/Object-Relational DBMSs," *JOT*, 2(3), pp. 113-126, 2003.

[IBM06] IBM, *DB2 Everyplace Enterprise Edition Release Notes for Version 9.1*, 2006.

[IBM08] IBM Information Center for DB2 Everyplace v9.1, http://publib.boulder.ibm.com/infocenter/db2e/v9r1/index.jsp, 2008.

[IBM09] IBM Ubiquitous Solution, http://www-903.ibm.com/kr/ubiquitous/ucity/health.html, 2009 (in Korean).

[KBL+06] Kim, G., Baek, S., Lee, H., Lee, H., and Joe, M., "LGeDBMS: a Small DBMS for Embedded System with Flash Memory," In *VLDB*, pp. 1255-1258, 2006.

[KV08] Koltsidas, I. and Viglas, S. D., "Flashing up the Storage Layer," In *VLDB*, pp. 514-525, 2008.

[KWKS09] Kim, Y., Whang, K., Kim, M., and Song, I., "A Logical Model and Data Placement Strategies for MEMS Storage Devices," *IEICE Trans. on Information and Systems*, E92-D(11), pp. 2218-2234, 2009.

[KWS10] Kim, Y., Whang, K., and Song, I., "Page-Differential Logging: An Efficient and DBMS-independent Approach for Storing Data into Flash Memory," In *SIGMOD*, 2010 (to appear).

[LKWL06] Lee, K., Kim, S., Whang, E., and Lee, J., "A Practitioner's Approach to Normalizing XQuery Expressions," In *DASFAA*, pp. 437-453, 2006.

[LM07] Lee, S. and Moon, B., "Design of Flash-Based DBMS: An In-Page Logging Approach," In *SIGMOD*, pp. 55-66, 2007.

[LMP+08] Lee, S. et al., "A Case for Flash Memory SSD in Enterprise Database Applications," In *SIGMOD*, pp. 1075-1086, 2008.

[MFHH05] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W., "TinyDB: an Acquisitional Query Processing System for Sensor Networks", *TODS*, 30(1), pp. 122-173, 2005.

[MS05] Microsoft, "SQL Server Compact 3.5," http://www.microsoft.com/sqlserver/2005/en/us/compact.aspx, 2005.

[MVT05] Moro, M. M., Vagena, Z., and Tsotras, V. J., "Tree-Pattern Queries on a Lightweight XML Processor," In *VLDB*, pp. 205-216, 2005.

[NG08] Nath, S. and Gibbons, P. B., "Online Maintenance of Very Large Random Samples on Flash Storage," In *VLDB*, pp. 970-983, 2008.

[Nori07] Nori, A. K., "Mobile and Embedded Databases," *IEEE Data Eng. Bull.*, 30(3), pp. 3-12, Sept. 2007 (also in 2007 *SIGMOD* as a tutorial abstract, June 2007).

[Orac06] Oracle, *Oracle Database Lite 10g Technical White Paper*, 2006.

[Orac08] Oracle Berkeley DB, http://www.oracle.com/technology/products/berkeley-db/index.html, 2008.

[SAH87] Stonebraker, M., Anton, J., and Hirohama, M., "Extendability in POSTGRES," *IEEE Data Eng. Bull.*, 10(2), pp. 16-23, 1987.

[SG04] Schlosser, S. W. and Ganger, G. R., "MEMS-Based Storage Devices and Standard Disk Interfaces: A Square Peg in a Round Hole?," In *FAST,* pp. 87-100, 2004.

[SO07] Seltzer, M. and Oracle Corporation, "BerkeleyDB: A Retrospective," *IEEE Data Eng. Bull.*, 30(3), pp. 21-28, 2007.

[SLMJ07] Sheng, B., Li, Q., Mao, W., and Jin, W., "Outlier Detection in Sensor Networks," In *MobiHoc*, pp. 219-228, 2007.

[WH04] Wu, J. and Hisa, T., "Analysis of E-commerce Innovation and Impact: a Hypercube Model," *Electronic Commerce Research and Applications*, 3(4), pp. 389-404, 2004.

[Whan07] Whang, K., "Development of Customizable/Lightweight DB Engine Technologies for Ubiquitous Small Devices," *National Research Lab Program*, National Research Foundation (NRF) of Korea, Proposal Apr. 2007, Project July 2007-June 2012.

[WLK+09] Whang, K. et al., "Tightly-Coupled Spatial Database Features in the Odysseus/OpenGIS DBMS for High-Performance," *GeoInformatica*, to appear, 2009 (on-line version at http://www.springerlink.com/content/m6851246706v6n65).

[WLL+05] Whang, K. et al., "Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features," In *ICDE*, pp. 1104-1105, 2005. This paper received the Best Demonstration Award.

[Wood01] Woodhouse, D., "JFFS: The Journaling Flash File System," http://sources.redhat.com/jffs2/jffs2.pdf, 2001.

[WPHL02] Whang, K., Park, B., Han, W., and Lee, Y., Inverted Index Storage Structure Using Subindexes and Large Objects for Tight Coupling of Information Retrieval with Database Management Systems, US Patent 6349308, Feb. 2002. (Appl. No. 09/250,487, Feb. 15, 1999)

[YAA07] Yu, H., Agrawal, D., and Abbadi, A. E., "MEMS-Based Storage Architecture for Relational Databases," *The VLDB Journal*, 16(2), pp. 251–268, 2007.

[YPM09] Yin S., Pucheral, P., and Meng, X., "A Sequential Indexing Scheme for Flash-Based Embedded Systems," In *EDBT*, pp. 588-599, 2009.