JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2007**

Vol. 6, No. 8, September - October 2007

# On Flash-Based DBMSs: Issues for Architectural Re-Examination

By **Sang-Won Lee** and **Won Kim**

**Abstract**

Flash memory is being rapidly deployed as data storage for mobile devices. Flash memory offers many advantages over hard disk, and considering its rapid technical improvement both in capacity and speed, it will have competitive advantage over mini-drive under 100 Gbytes within a few years. As the applications in hand held devices become large, complex, and more data-intensive, they require database technology. However, flash memory has three distinct characteristics that make today's disk-based database technology unsuitable. In this article, we first review the characteristics of the flash memory. Then we identify several key issues in the architecture of a flash-based database system that require careful re-examinations.

## 1 INTRODUCTION

Flash memory offers many advantages over its competitor, hard disk, including low power consumption, non-volatile storage, high performance, physical stability, small size, light weight, and portability. As a result, it is being rapidly deployed as data storage for mobile devices such as PDAs, MP3 players, mobile phones and digital cameras. In particular, for its popular MP3 player iPod and its cell phone iPhone, Apple is known to have made long-term supply agreements with major flash memory vendors including Samsung.

Flash memory is a type of EEPROM (electronically erasable programmable read only memory), originally intended to replace the hard disk when developed by Toshiba [Toshiba 2004]. It has no moving parts, unlike the hard disk. The term "flash" is said to have originated from the observation that it can write a sector of data (512bytes, also called as page) or erase blocks of multiple pages (usually 16 or 32 sectors) simultaneously in one action, in contrast to the byte-by-byte EEPROM.

Depending on the logic gate type used, the flash memory can be divided into two types: NOR and NAND. NOR flash, developed by Intel, is a random-access device, like RAM, that is directly addressable by the processor, and so it is good for executing program code. NAND flash is not directly addressable and is controlled using an indirect disk I/O-like interface through an 8-bit bus to an internal command and address register.

Cite this column as follows: Won Kim and Sang-Won Lee: "On Flash-Based DBMSs: Issues for Architectural Re-Examination", in *Journal of Object Technology*, vol. 6, no. 8, September-October 2007, pp. 39-49 http://www.jot.fm/issues/issue_2007_09/column4

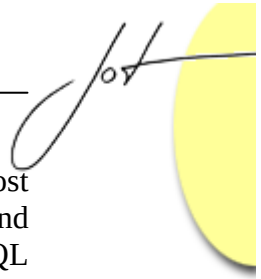
NAND flash requires fewer gates than NOR to store the same number of bits, and so it is smaller and denser and thus is appropriate for large data storage.

For the past decade, the NOR flash has dominated the flash memory market. Due to the rapidly escalating demand for large data storage media in hand-held device, however, the revenue from the NAND flash memory has recently surpassed that of the NOR flash Intel, the leader in NOR flash memory, recently launched a joint NAND flash venture company, IM Flash Technologies, with Micron Technology. In particular, vendors of mobile phones, where various complex applications with relatively small data run, will switch from NOR to NAND flash. In fact, Samsung Electronics recently launched a mobile phone with its 4Gbyte NAND flash, and also announced 128Gbyte solid-state disk (SSD), a NAND flash-based replacement for hard-disk drives, targeting the next generation laptop's storage market. The company expects that the market share of flash-based laptops will reach 20 ~ 30% in 2010 [Cebit 2006]. Recently, hybrid types of flash memory which have all the advantages of both NOR and NAND flash memory have been announced, including Samsung's OneNand and M-Systems' mDOC [Santarini 2005]. In this article, we will focus on NAND flash memory technology, because it is more adequate for large volume data management than NOR.

The strongest rival of the NAND flash memory are small hard disks. However, according to [Paulson 2005], the flash memory will win over the mini-drive because of its higher storage density. In terms of price per Mbyte, the NAND flash memory is currently more expensive than the mini-drive. However, the price per Mbyte of flash memory continues to drop nearly 50% every year. In other words, the density doubles every year, and this trends is expected to continue for the coming five years [Lawton 2006]. In contrast, the recent annual average increase in area density of the mini-drive is between 10 to 30 percent because the polarity of data written in hard-disks may change unexpectedly and thus corrupt the data if the area is too small. In addition, the mini disk has a limit in size shrinkage because of the motors and other moving parts. IDC reports that 0.85 inches in diameter is on the absolute edge – a tiny disk drive won't be able to compete with flash storage [Paulson 2005]. Currently, the winning point of the NAND flash memory over the mini-drive in terms of both capacity and price is around a few Gbytes. It appears reasonable to anticipate that flash memory will beat the mini-drive even in several tens of Giga bytes storage markets within five years.

Besides this pure capacity/price economics, certain types of military applications embedded in military equipment such as tanks and air planes will definitely prefer flash memory as its storage media because of environmental requirements, e.g. resistance to shock and heat. Another advantage of the flash memory comes from the fact that it is an electronic device, unlike the hard disk which requires disk head and arm movement. This advantage frees the flash memory from the time-consuming seek and rotational delay. Even in high-end applications, flash memory can be arrayed together to offer capacity comparable to that of hard drives at higher speeds.

As the applications for mobile devices that use the flash memory as storage become more complex and data-intensive, there will need an embedded SQL-based database management system (DBMS) rather than a file system to both support application development and provide a runtime engine [Kim et al 2006]. Today every major



commercial RDBMS vendor offers a mobile/embedded/light-weight DBMS for most handheld or embedded OS platforms including Symbian OS, Palm OS, Window/CE, and embedded Linux. These products include Oracle Lite [Oracle 2006], Sybase SQL Anywhere [Sybase 2006], MS SQL Server for Window/CE [Microsoft 2006], and IBM DB2 Everyplace [IBM 2006]. They have been developed for reduced code footprint by stripping down database features. This is true also for open source DBMSs such as MySQL, BerkelyDB, and PostgresDB. However, as far as we know, no product supports flash memory as secondary storage.

There are promising applications for flash-based DBMSs, besides the handheld devices. As the MyLifeBits project in Microsoft [Gemmel et al 2006] has shown, personal information management systems need to manage complex types of data and queries [CACM 2006], and the flash memory can be used for personal information management systems. The flash memory is a candidate for supporting the data management needs of the next generation smart-card systems. Further, wireless sensor devices almost use on-chip/off-chip NAND flash as its storage media and they will also require database functionality because of the data size and need for complex searches [Yazti et al 2005]. It is also conceivable for the flash memory to be used to support enterprise applications [Gray 2007, Lee 2007] after re-engineering system software to be flash-aware.

The flash memory has three characteristics that profoundly affect its performance in managing data, that is, read and update of a database. First, the flash memory, unlike hard disk, has asymmetric read and write speed. The read time for a sector from flash memory is typically 30 us, while the write time is 300 ms. Second, with the flash memory, unlike the hard disk, in order to overwrite existing data, the entire block that contains the data must be erased first and then the new data can be written to the block. The erase operation is much slower than the write operation; the erase time for a block (= 32 sectors) is around 2 ms. Third, because of its electronic property, the flash memory, unlike the hard disk, has no moving parts, and thus it has no seek and rotational latency. These characteristics make it infeasible for a DBMS developed for hard disks as secondary storage to readily be used for the flash memory, and therefore force a re-examination of many key parts of the DBMS architecture.

In Section 2, we review NAND flash technology and a technique for making the flash appear to applications as a disk drive. In Section 3, we examine several issues in implementing various aspects of a DBMS engine that needs to use the NAND flash as secondary storage. Section 4 concludes the article.

## 2 OVERVIEW OF THE NAND FLASH MEMORY TECHNOLOGY

Flash memory stores data in an array of floating gate transistors called cells. Originally, NAND flash memory was intended to replace the hard disk, and thus its smallest unit of data storage is a sector of 512 bytes (also called a page) as is the case with the hard disk. Each sector has additional spare area of 16 bytes for metadata such as error correction code. Thus, the physical size of a sector is 528 bytes. One block consists of 32 sectors,

and thus its size is usually 16 Kbytes. We call such flash memory small block NAND flash. Flash memory vendors have started producing large block NAND flash with blocks of 64 sectors and sectors of 2,212bytes (thus, the size of a block is 128Kbytes) in order to allow faster write and erase operations for high-end applications.

There are only three basic operations in a NAND flash: read a page, write a page, and erase a block. A read or write command specifies (chip#, block#, sector#), where chip# is the flash chip number, block# the block number in the device, and sector# the sector number in the block.

Table 1 compares the characteristics of the NAND flash and the mini hard disk.

| **Category** | **Mini Hard-Disk** | **NAND Flash** |
|---|---|---|
| Read/Write Unit | Sector (512bytes) | Sector (512bytes) |
| Mechanical Moving Parts | Spinning disk head (seek + latency) | No (all electronics) |
| Energy Consumption | High | Low |
| Shock Resistance | Bad | Good |
| Density | Has reached limit | Room for improvement |
| Endurance | 1,000,000 overwrites | 10-100,000 erases/cell |
| Random Read/Write | High (in mili-seconds) | Low (in tens of micro-seconds) |
| Sequential Read/Write | High bandwidth | High bandwidth |
| Size Reduction | Has reached limit (0.85 inch) | Room for improvement |
| Noise & Vibration | High | No |

Table 1. Comparison of the mini hard-disk and the NAND flash

In order to make the flash memory appear to applications as a disk drive, the flash translation layer (FTL) has been developed [Kim et al 2002]. Figure 1 shows the general organization of a NAND flash memory system and the position of the FTL within it. A NAND flash memory system consists of one or more flash memory chips, a controller that executes the FTL code in ROM, an SRAM (static RAM) that maintains the address mapping information, and a PCMCIA (Personal Computer Memory Card International Association) host interface. The host system views the flash memory as a hard disk-like device, and thus issues read or write commands along with "*logical*" sector addresses and data. The FTL translates the commands into low-level operations, namely read, write and erase, using "*physical*" sector addresses. To do the address mapping, the FTL looks up the address mapping information in the SRAM.

Because the erase operation is the performance bottleneck for the flash memory, it is very important to reduce the number of erase operations resulting from write operations. The FTL, in general, reserves a small number of log blocks in the flash memory as temporary storage for overwrites [Kim et al 2002], and redirects each overwrite request from the host to a reserved log block, softening the performance impact of the "*erase-before-write*" paradigm of the flash memory. In fact, various FTL algorithms have been proposed so far [Kim et al 2002], and the performance of each FTL varies considerably, depending on the characteristics of the applications. After finishing an overwrite operation, the FTL changes the address mapping information in the SRAM. The outdated block can be erased later.

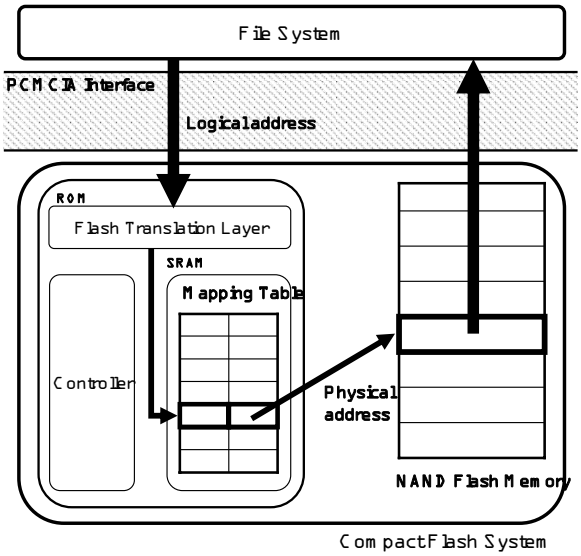



Figure 1. The Architecture of a NAND Flash Memory System

Besides the address mapping, the FTL carries out several important functions, such as guaranteeing data consistency and flash wear-leveling (or block-recycling). The FTL maintains a consistent state for data and metadata, even when the flash memory encounters an unexpected power outage. Wear-leveling is important because there is a physical upper limit on the maximum number of erases allowed for each block (usually, 100,000 times). If a block is erased above the upper limit, the block may not function correctly. For wear-leveling, the FTL tries to make the physical blocks erased as evenly as possible without incurring performance degradation in the process. Even though a block that exceeds the upper limit may still work, the FTL marks it as invalid for the safety of the flash memory.

## 3 ARCHITECTURAL CONSIDERATIONS FOR A FLASH-BASED DBMS

In this section, we examine the major DBMS implementation techniques which are heavily impacted by the three characteristics of the flash memory. Table 2 summarizes the impact of each of the three characteristics on seven different aspects of the architecture of a DBMS.

| DBMS modules | No overwrite | No mechanics | Asymmetric Read/Write Speed |
|---|---|---|---|
| Buffering | O | | O |
| Unit of IO | | O | O |
| Data Clustering | O | | O |
| Indexing | O | O | |
| Page Layout | O | | |
| Query Processing & Optimization | O | O | |
| Transaction Management | O | | |

Table 2. Impact of the flash memory on DBMS architecture

One technical issue with the flash file system is how to deal efficiently with small random writes, that is writing a small amount of data to blocks of the flash memory. The trouble with the small random writes is that they require the very expensive erase operations. One source of the small random writes is updates to metadata [Kim et al 2002]. Most (disk-based) DBMSs today support record-oriented page layout, where the attributes of a record are placed contiguously. This layout works well for the OLTP (online transaction processing) applications. To compound the problem for a flash-based DBMS, a data block (also an index block) usually consists of a header structure and the record area. Thus, any record update, including insert, delete, and updates, entails changes in the header structure, which means erase operations for the head structure in a flash-based DBMS. One possible approach to solve this problem is to adopt an approach that had been used in the very early days of RDBMSs but which has since been discredited. In the storage manager of PostgresDB [Stonebraker 1987], records are not overwritten, and instead and every update is turned into difference-record insertion. Multiple versions of a logical record exist and they are linked together.

The issue of an optimal block size for the unit of I/O needs a re-examination. In an OS or a DBMS, the unit of IO is a block, and, with increases in main memory size, CPU power, and IO bandwidth, the size of the IO unit has continuously increased. For example, the default block size is 16Kbytes and in some read-intensive applications such as data warehouses, they recommend a block size of up to 64Kbytes. In flash-based systems, a large block size as the unit of I/O is not justified. First, considering the fast read time, prefetching of data, which is one of the justifications for a larger block size, is not as important. Instead, a large block size causes the buffer space to be wasted. To make matters worse, a small update to a large block will result in overwrites, which in turn will lead to the erase operations.

Buffer replacement algorithms used in an OS or a DBMS in general assume that the speed of the read and write operations are about the same, which is true in the case of hard disks. In flash-based systems, however, the traditional metrics such as ‘buffer hit ratio’ are not appropriate as performance indicator because of the speed asymmetry

between the read and update operation. Buffer replacement algorithms need to be re-examined for flash-based systems. One naïve guide for this scheme may be stated as follows: “Try to reduce the number of writes/erases at the expense of the read operations.” A new performance metric is also needed, similar to the ‘buffer hit ratio’.

Current DBMS use data clustering techniques to store related data close together on secondary storage. This is to minimize disk head movement, and also to prefetch related data together. In this respect, the traditional clustering techniques are read-oriented [Guinepain and Gruenwald 2005]. However, the justification for prefetching data is significantly reduced in flash-based systems because of the fast read time. Instead, considering the relatively slow write/erase operations in flash-based systems, it is more important to write-oriented clustering techniques, that is, to locate the data objects to be written to the same physical block.

In disk-based DBMSs, indexes are used to support selective access to records based on the values of the columns indexed. The indexes have to be updated when the values of the indexed columns are updated, new records are inserted, existing records are deleted, and index pages are split or merged. Again, small random writes to the indexes will lead to the erase operations in flash-based systems. Some recent work, such as the write-optimized B+ tree [Graefe 2006], for disk-based DBMSs, log structured merge(LSM) tree [O’Neil et al 1996], and NAND flash-based hash indexing [Yazti et al 2005], may be a reasonable starting point for work on indexing techniques for flash-based DBMSs.

Query processing and optimization modules of the DBMS architecture are affected by all three characteristics of the flash memory. In disk-based DBMSs, the hash join and sort merge join methods outperform the nested loop join method. This is attributable to the fact that the read and write speeds are the same, and that there is sufficient RAM to run the hash or sort merge join algorithm. However, for flash-based systems, sort merge and hash join require write operations to secondary storage when the data could not fit in the RAM, leading to a performance problem. In contrast, nested loop join becomes more attractive, since it works in read-only mode; it is even more attractive if an index exists on the inner table. In addition, nested loop join does not require extra memory for the joins. For this reason, we need to revisit the cost model of the join algorithms for flash-based systems.

For a given query, the query optimizer in a DBMS enumerates various execution plans, estimates the cost of each execution plan, and chooses the least costly plan for the execution of the query [Selinger 1978]. The cost estimation part of the query optimization techniques needs a careful re-examination, in light of the fact that the cost of a write in the flash memory is not the same as the cost of a read, and may also involve the cost of an erase. At this point, we should mention the power issues in the flash memory. We may simply assume a linear relationship between the time that an operation takes and the amount of power it consumes. The power consumption for each operation in the flash memory is currently approximated as follows: 24uJ (read), 763uJ (write), 425uJ (erase) [Zheng et al 2003]. Another cost modeling issue is the cost of index scan. In the flash memory, an index scan is preferred to a full table scan because of uniform random access and fast read time. That is, the cost of an index scan in the flash memory is much less

than that for the hard disk, and we can estimate its cost more correctly because of uniform random access speed.

Transaction processing has two key technical components: concurrency control and recovery. For concurrency control in flash-based systems, the snapshot isolation approach may be appropriate, rather than locking [Berenson et al 1995]. In this approach, several versions of the same data are maintained in order to answer queries which require different snapshot values at different time points. This concurrency control model can be efficiently supported in flash memory if we exploit the not-in-place update characteristics of the flash memory. Each version may be written to a different physical sector in order to avoid the erase operation. The fact that there might exist several versions of the same data can also be exploited to replace the traditional log-based recovery mechanism [Lee 2007]. That is, each version may be regarded as a form of a log. However, we can not keep all historical copies in the flash memory, and we need to guarantee the consistency of data in combination with the check point technique.

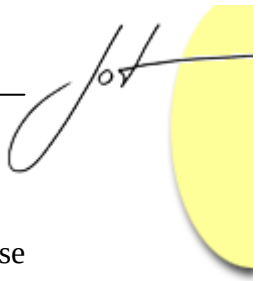
## 4 CONCLUDING REMARKS

In this article, we gave a brief review of the flash memory technology, which is a strong competitor against the hard disk, at least in the storage market for handheld devices. In the review, we emphasized three characteristics of the NAND flash memory that impact the architecture of today's DBMS that have been designed with the view to using hard disks as storage. Then we examined several DBMS architecture issues that require careful re-examinations.

In concluding, we would like to remark that in this article we assumed that the flash memory is just storage. However, considering the rapid growth of CPU power and memory embedded in the flash memory controller, we can imagine that the flash memory controller will eventually have a CPU and memory which are powerful enough to run a full-blown simple DBMS. That is, the flash memory controller itself will become a computing platform. This is similar to the database machine vision of Jim Gray [Gray 2005], which says it is quite feasible to have intelligent disks that offer either database access (SQL or some other non-procedural language) because each disk controller now has tens of megabytes of storage and a very capable processor which can run a complete DBMS engine. In fact, a few SQL-on-Chip approaches such as [Calpont 2006, Anciaux et al 2003] have recently been proposed.

## ACKNOWLEDGMENTS

This research was supported in part by MIC, Korea under ITRC IITA-2006-(C1090-0603-0046), and in part by MIC & IITA through IT Leading R&D Support Project.



## REFERENCES


[Anciaux et al 2003] Nicolas Anciaux, Luc Bouganim, Philippe Pucheral, Database Components on Chip, ERCIM News No. 54 July 2003.

[Berenson et al 1995] Hal Berenson, Phil Bernstein, Jim Gray, Jim Melton, Elizabeth O'Neil, Patrick O'Neil, A Critique of ANSO SQL Isolation Level, Proceedings of the ACM SIGMOD, 1995.

[CACM 2006] Jaime Teevan, William Jones, Benjamin B. Bederson(Guest Editors), Personal Information Management, Communications of the ACM, Vol. 49, No. 1, January 2006.

[Calpont 2006] Calpont Corp., Introduction to the Calpont Database Appliance, Calpont White Paper, http://www.calpont.com, 2006.

[Cebit 2006] http://www.webwereld.nl/articles/40332/samsung-offering-flash-based-disk-to-customers.html, Web Wereld, 2006

[Gemmel et al 2006] Jim Gemmell, Gordon Bell, Roger Lueder, MyLifeBits: A Personal Database for Everything, Communications of the ACM, Vol. 49, No. 1, January 2006.

[Graefe 2006] Goetz Graefe, B-tree indexes for high update rates, ACM SIGMOD Record, Vol. 35, No. 1, March 2006.

[Gray 2005] Jim Gray, What's Next For Database Systems?, http://research.microsoft.com/~gray

[Gray 2007] Jim Gray et al., Flash Disk Opportunity for Server-Applications, http://research.microsoft.com/~gray

[Guinepain and Gruenwald 2005] Sylvain Guinepain, Le Gruenwald, Research Issues in Automatic Database Clustering, ACM SIGMOD Record, Vol. 34, No. 1, March 2005.

[IBM 2006] IBM Corp., DB2 Everyplace, http://www-306.ibm.com/software/data/db2/everyplace/, 2006.

[Kim et al 2002] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, A Space-Efficient Flash Translation Layer for CompactFlash Systems, IEEE Transactions on Consumer Electronics, Vol. 48, No. 2, May 2002.

[Kim et al 2006] Gye-Jeong Kim et al., LGeDBMS: a Small DBMS for Embedded System with Flash Memory, Proceedings of VLDB 2006.

[Lawton, 2006] George Lawton, Improved Flash Memory Grows in Popularity, IEEE Computer, January 2006.

[Lee 2007] Sang-Won Lee, Bongki Moon, Design of Flash based DBMS: An In-page Logging Approach, Procedings of the ACM SIGMOD, Beijing, China, 2007

[Microsoft 2006] Microsoft Corp., MS SQL Server 2005 Mobile Edition, http://www.microsoft.com/sql/editions/sqlmobile/default.mspx, 2006.

[O'Neil et al 1996] Patrck O'Neil, Edward Cheng, Dieter Gawlick, Elizabeth (Betty) O'Neil, The Log-Structured Merge-Tree, Acta Informatica Vol. 33, No. 4, June 1996.

[Oracle 2006] Oracle Corp., Oracle(r) Database Lite to Support Symbian OS, http://www.oracle.com/database/Lite_Edition.html

[Paulson, 2005] Linda Dailey Paulson, Will Hard Drives Finally Stop Shrinking?, IEEE Computer, May 2005.

[Santarini 2005] Michael Santarini, NAND versus NOR, EDN, October 2005.

[Selinger 1978] Patricia Selinger et al., Access Path Selection in a Relational Database Management System, Proceedings of SIGMOD, 1978.

[Stonebraker 1987] Michael Stonebraker, The Design of the POSTGRES Storage System, Proceedings of VLDB, 1987.

[Sybase 2006] Sybase Inc., Sybase SQL Anywhere, http://www.sybase.com/products/mobilesolutions/sqlanywhere, 2006.

[Toshiba 2004] Toshiba America Electronic Components, Inc., NAND Flash Applications Design Guide, 2004.

[Yazti et al 2005] Demetrios Zeinalipour-Yazti, Song Lin, Vana Kalogeraki, Dimitrios Gunopulos, Walis A. Najjar, MicroHash: An Efficient Index Structure for Flash-Based Sensor Devices, 4th USENIX Conference on File and Storage Technologies (FAST 2005), December 2005.

[Zheng et al 2003] Fengzhou Zheng, Nitin Garg, Sumeet Sobti, Chi Zhang, Russell E. Joseph, Arvind Krishnamurthy, Randolph Y. Wang, Considering the Energy Consumption of Mobile Storage Alternatives, Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Orlando, Florida. Oct 2003.

## About the authors

**Won Kim,** Professor and Univeristy Fellow with the School of Information and Communication Engineering at Sungkyunkwan University, Suwon, S. Korea. He is Editor-in-Chief of ACM Transactions on Internet Technology (www.acm.org/toit). He is Global General Chair of the Human.Society@Internet International Conference. He is the recipient of the ACM 2001 Distinguished Services Award, and is an ACM Fellow. He can be reached at wonkim@skku.edu

**Sang-Won** Lee is an Assistant Professor with the School of Information and Communication Engineering at Sungkyunkwan University, Suwon, S. Korea. Before that, he was a research professor at Ewha Womans University and a technical staff at Oracle, Korea. He received a Ph.D degree from the Computer Science Department of Seoul National University in 1999. His research interest is in flash-based database technology. He can be reached at swlee@skku.edu