Towards Ubiquitous Database in Mobile Commerce

Kimio Kuramitsu and Ken Sakamura
Interfaculty Initiative in Information Studies, University of Tokyo
7-3-1 Hongo Bunkyo-ku, Tokyo 113-0033. Japan
Phone: (+81) 3 5841 2484 / Facsimile: (+81) 3 5841 8459
{kuramitsu, sakamura}@iii.u-tokyo.ac.jp

ABSTRACT
Ubiquitous database places data everywhere. We focus on contactless smartcards combined with a small processor and memory for data storage. Very small DBMS implemented on them can interact through queries on a wireless communication. Ubiquitous database attaches such DBMSs to real world "objects" physically, and then allows different organizations to share information retrieved directly from physical goods, materials, or persons. We combined the theory with practice in the ticket digitalization project. Through the experience of our project, this paper will discuss how to manage ubiquitous database in the context of mobile commerce applications.

Keywords
Smartcard database, Mobile Commerce, Ubiquitous computing, Data Model and Querying for ubiquitous data

1. Introduction
The mobile database, or embedded database on a mobile device, is starting to become an important player in the industry. Many challenges to develop these small databases are being tried on portable computing devices, including smart phones, PDAs (personal digital assistances), intelligent appliances and other embedded systems. [8] In this paper, we will focus on contactless smartcard combined with a small processor, memory for data storage and wireless communication capability.[5]

Smartcard is a tiny computing device for an individual carrying with tokens, personal business data and private information. More recently, not a few researchers and developers begin to discuss a small data management system on the card. [2, 8] These previous works help us develop a small footprint system effectively. However, data applications will be developed against a group of smartcard databases. The objective of this paper is to discuss the design of smartcard database as multi-database systems.

We call a set of the small-embedded database *ubiquitous database*. In the ubiquitous database, a very small DBMS will be attached to each real-world entity, such as goods, materials and persons. A data object on a DBMS can move to another electronically. Furthermore, the DBMS itself moves in the real world as the corresponding entity moves. An external application system interacts with the DBMSs, when each DBMS is moved in the position that can be accessed from the application system. That is, the ubiquitous database enables different organizations to share electronic data through the DBMSs attached to mobile entities. In this context, the contactless smartcard could be thought of the most suitable devices to implement such a ubiquitous DBMS.

Based on the ubiquitous database, we conducted an experimental digital ticket project. The mechanisms we designed for the experiment include: a flexible data model for high interchangeability, the consistency control of number of distributed data objects, two-phase commitment synchronization with remote databases, and role-based access control. Depicting these designs, we will discuss four properties (Interoperability, Consistency, Concurrency, and Privacy) required to use ubiquitous database in mobile commerce between enterprise applications.

This paper will report the possibility (and some discussion stages) of database applications in ubiquitous computing and mobile commerce. Section 2 elaborates on the concept of ubiquitous database. Section 3 distills requirements from our "Electronic Ticketing" application. Section 4 discusses interoperability, consistency, concurrency and privacy using the design of our prototyped system. Section 5 concludes the paper.

2. Concept OF Ubiquitous DATABASE

2.1 Ubiquitous Computing and Database
The ultimate goal of *ubiquitous computing* is to place computers everywhere in the real world environment, providing ways for them to interconnect, talk and work together. [10,11] The concept is now shifting computing paradigm from machines in a room to the augmented

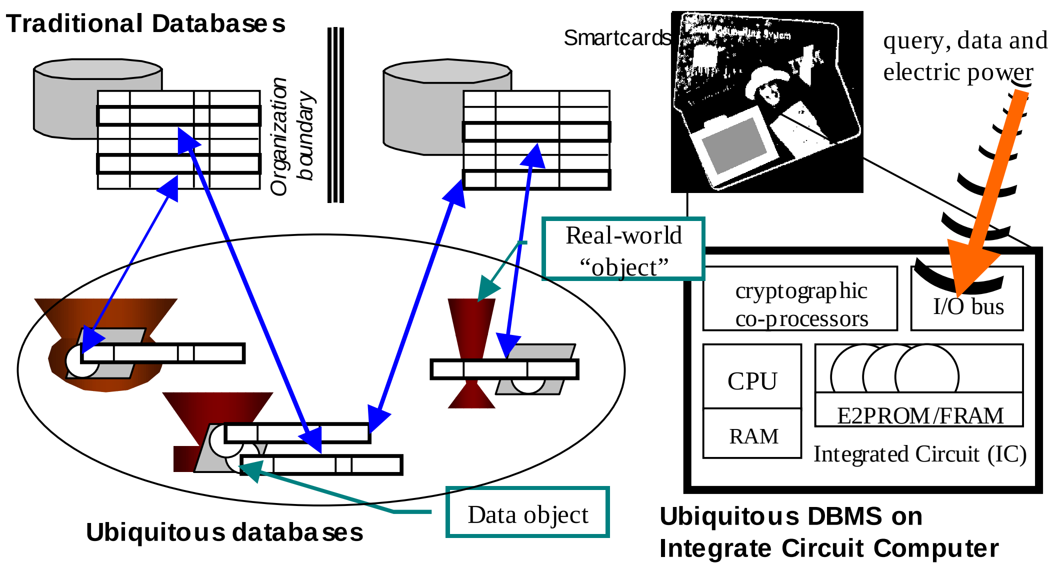

Figure 1: The Concept of Ubiquitous Database

contexts in the real world. Similarly, ubiquitous database will make data everywhere possible. Every real-world object originally has information such as properties and its historical changes. Traditional databases collect such information to manage in a central manner. However, the central management, although it is highly efficient, does not necessary meet the demand to establish data applications across different organizations, as electronic commerce and digital libraries today demand.

The ubiquitous database augments "object" that manages information about itself. A database-augmented "object" enables data application integrations through the movement of "object" in the real world. An augmented product moving from one company to another can carry electronic updating records at the same time. An augmented museum piece can play different roles in showing data for visitors, researchers and librarians. A person wearing a small database can autonomously interact with social information systems under privacy controls. The ubiquitous database provides anyone with the method to retrieve information directly from the real world "objects". To share information across organizations, the ubiquitous database becomes a multi-database environment.

2.2 Integrated Circuit Computer

The ubiquitous DBMSs would be implemented based on the contactless smartcard technology. For this, there are three reasons. First, the smartcard is of an integrated circuit computer, on which multiple components such as CPU, crypto coprocessors, RAM/ROM, nonvolatile storage (E2PROM or FeRAM), and communication capabilities are combined with a single chip. A minimum set of hardware required to develop a DBMS is complete. Second, the smartcard has physical tamper resistance; all accesses from outside are fully controlled and, if needed, automatically encrypted. The hardware tamper-resistance enables us to secure a small DBMS, as if a firewall system secures traditional database servers. Third, the contactless smartcard need no battery. One crucial problem in today's mobile computing is electric power supply. The power for the processor is transmitted as an electromagnetic wave or a micro wave, which carries a query and data simultaneously. Thus, we can theoretically attach such an ICC to any real-world solid "objects", instead of the plastic card.

On the other hand, a problem with the contactless smartcard lies in limited computing resources, because it limits the maximum power consumption of the processor. The latest smartcard can only support an 8bit CPU, a 16~32K ROM and 512bytes of RAM for software, and 4~16K EEPROM for data storage. [5]

2.3 Related Works

An active tag is regarded a preceding system to ubiquitous database. The active tag attached to an real world "object" sends infrared (IR) or radio frequency (RF) signals to make us identify the object both electronically and physically. [4,6,9] The identification thus helps select data objects from remote database systems. However, that research direction is far from an interactive data management system.

We will turn to the data management on a smartcard. The majority of current smartcards have a file system that supports the storage and retrieval of file objects. [6] For example, ISO 7816-4 defines a hierarchical files system, similar to unix's directory and file structure. The ISO standard also defines a class of commands, named Application Protocol Data Units (APDUs), which are used to read and write data between the host and the smartcard.

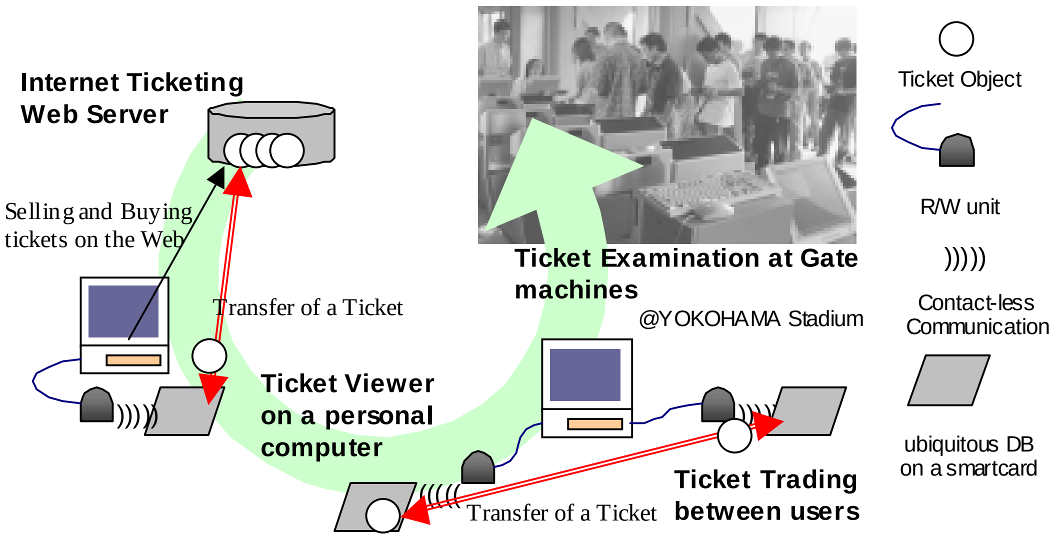
Internet Ticketing Web Server
Selling and Buying tickets on the Web
Transfer of a Ticket
Ticket Viewer on a personal computer
Ticket Examination at Gate machines
@YOKOHAMA Stadium
Ticket Trading between users
Transfer of a Ticket
Ticket Object
R/W unit
)))))
Contact-less Communication
ubiquitous DB on a smartcard

Figure 2: The Overview of Experimental Environment in the Ticket Digitalization Project

However, access control to a file depends on each implementation.

More recently, not a few researchers and developers already began to focus on the advantage of the data management system on smartcard. [2, 8] The main discussion in these previous works was on trade-offs between small footprint and effectiveness. In contrast, we will add the discussion on multi databases accessed by different organizations in the context of mobile commerce applications.

3. Electronic Ticketing

Japanese Information Processing Development Center (www.jipdec.or.jp) organized an experimental project to integrate electronic ticketing services through smartcards. The ubiquitous database we proposed was adopted to manage ubiquitous ticket objects among sellers, users and ticket organizers. Based on our initial ubiquitous DBMS, we developed a ticket wallet for a personal user to hold ticket objects. This section elaborates on the requirements for applying ubiquitous database to mobile commerce, presenting the experiment.

3.1 Overview

The experimental environment was comprised of several data applications. A ticket reseller sold tickets from one's web site, while the buyers downloaded electronic tickets to their ticket wallets. The ticket wallet showed its owner the content of tickets anytime. The personal users also traded or moved tickets freely between ticket wallets. Sometimes, the ticket reseller updated out of date ticket information from the website. Finally, a ticket organizer examined a ticket stored on ticket wallets through the contactless communication at a gating system. All these applications were integrated over our initial ubiquitous database system. Figure 2 shows the overview of experimental environment in the ticket digitalization project.

3.2 Requirements

Here we will distill the requirement of ubiquitous database in mobile commerce applications. Alice is a test user that wants to buy new tickets to a baseball game and then connects her ticket wallet (that is, her ubiquitous DBMS) to an Internet ticketing server. The ticketing company creates the ordered tickets on her wallet. In this time, only ticketing company can create the tickets. After the purchase, she will want to pass some of the tickets to friends or families . The ticket must be moved atomically between wallets. When she wants to enter into the baseball game stadium, a gating system will automatically examine the ticket stored on her wallet. The ticket wallet must show the examiner only the baseball ticket, while Alice can view all tickets stored on the wallet. Finally, the gating system updates some ticket properties (e.g., used and date) to record gate-in information. In order for a user to prevent getting in repeatedly using the same ticket, this update process must be synchronized atomically with a remote gate database.

The requirements can be summarized as follows.

- **Interoperability**. The objects stored on the DBMS should represent irregular properties of real-world "object", according to the context of commerce applications. The application developers on the other hand should be given a universal means to formulate their demands to create, search, read, update, move and delete objects.
- **Consistency**. Over a set of multiple ubiquitous DBMSs or the class of objects, the number of

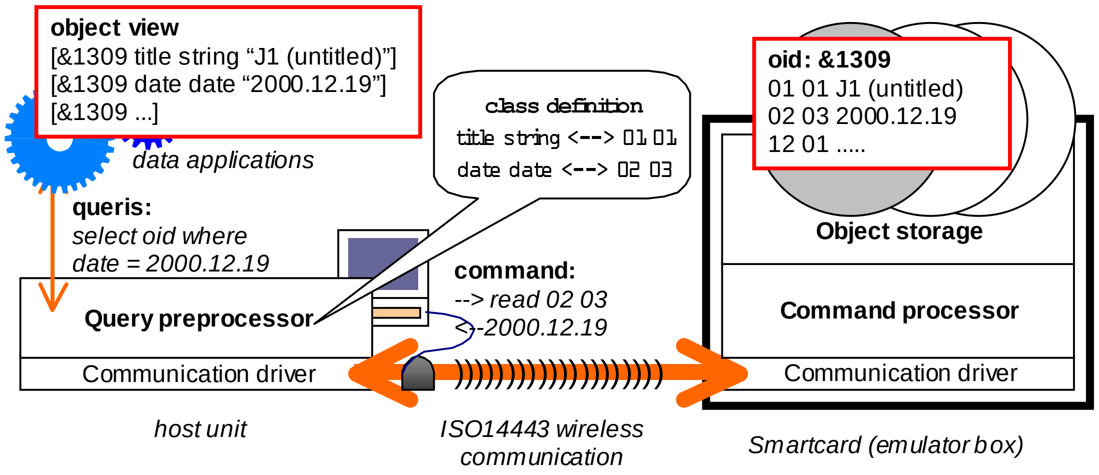

Figure 3: The System Architecture of Ubiquitous DBMS

distributed objects should be controlled. In addition, each object over a ubiquitous database environment is distinct and should be identified uniquely.

- **Concurrency**. The DBMS should ensure harmonious coexistence among different application sessions working at the same object on the DBMS.
- **Privacy**. Each DBMS should authenticate the subjects (that is communicating entities), and then must protect every query operation from unauthorized disclosures and modifications.

4. Design Discussions

This section will report the findings acquired from the experiment, presenting our designs and implementations of the prototype system.

4.1 System Architecture

A first question is the architecture design of ubiquitous DBMS. On the small processor and limited memory resource, it is very difficult to implement the full functionality of rich DBMS. We therefore divided the function of DBMS into two parts on a host PC and a smartcard. The part on the host is a preprocessor that transforms a query language into primitive commands. In addition, it takes charge of the management of schema-based view (4.2) and transaction roll-backing (4.4). On the other hand, the part on the smartcard is a command processor that executes primitive commands that create, read, write, and delete data objects. Also, the permission of the executions is controlled on it (4.5). The both parts are connected through an encrypted wireless communication.

This architecture also makes it easier to implement the query processor by just wrapping a command-based card. However, the increased interactions by a disassembled query become too costly. In the experimentation, we used a contactless smartcard emulator box supporting the ISO14443-based microwave communication. While the movement of 800 bytes of a ticket object took less than 2 seconds (with a single command), the retrieval of 20 properties on the same ticket took more than 20 seconds (with 20 commands). Accordingly, on the smartcard we need more sophisticated commands to select or update multiple properties.

4.2 Data model and query language

To represent a variety of object, we used a simple and flat structure based on a self-describing object. The object consists of multiple properties, each of which is denoted as a tuple of [oid, label, type, value], where oid is an object identifier, the label is a name of the property and the type is a class of the value. With encoding the label-type pair to a numeric code, this structure can be mapped into a Type-Length-Value record in ASN.1 on the smartcard. (Note that we used an original record with an access permission field, instead of the TLV.)

In addition, each object needs some metadata to manage itself. We in advance defined the semantics of several labels: META.CLASS for a class name of the object, META.PASSWORD for the authentication to access, META.LOCK for the concurrency control, and so on. For example, the query preprocessor first refers to a class name in META.CLASS, then retrieves the class definition from the Internet, and ultimately can convert between any labels and encoded numbers. Figure 3 shows this mapping. Using primitive commands (such as read and write) to the meta semantics, we can manage an object. In summary, we could downsize the command processor to a extent in which it supports at most 17 commands.

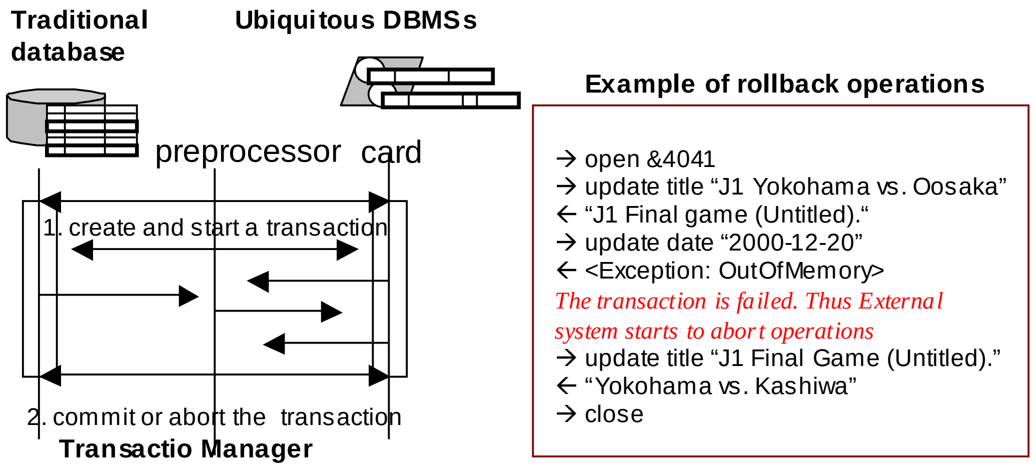
Traditional database
Ubiquitous DBMSs
preprocessor
card
1. create and start a transaction
2. commit or abort the transaction
Transactio Manager
Example of rollback operations
→ open &4041
→ update title "J1 Yokohama vs. Oosaka"
← "J1 Final game (Untitled)."
→ update date "2000-12-20"
← <Exception: OutOfMemory>
The transaction is failed. Thus External system starts to abort operations
→ update title "J1 Final Game (Untitled)."
← "Yokohama vs. Kashiwa"
→ close

Figure 4: 2PC Scheme for Synchronize between Ubiquitous and Traditional Database

4.3 Object Consistency

Our approach to object consistency is based on the central management of object lifecycle. Objects are created centrally at each class, and then distributed to other devices. That is, a ticket can be created only at the DBMS in the ticketing server. The other DBMSs attached to ticket wallets are limited to execute the queries create and delete object. When a user wants to delete an unnecessary ticket, he or her ought to send it to a trash box server.

In addition, we developed several techniques to secure the consistency. These techniques included a digital signature that certificates for the authenticity of object identifier across multiple companies and an atomic transfer without the possibility of creating or destroying it. Here we will omit the details due to space constraints. The paper [7] already reported our atomic transfer protocol between two parties.

4.4 Concurrency Control

To synchronize with a remote database, we have implemented basic two-phase commitment scheme. Since it is difficult to record the full state of a transaction on the card, we designed that the preprocessor controls the commitment and the rollback. To exclude coexistence of multiple transactions, we also introduced the lock mechanism for each object. Figure 4 shows the control of roll backing by combining primitive commands. (Note that old values are stored on the preprocessor.)

The session communicating with a moving ubiquitous DBMS will be often disconnected. (For example, consider the gating examination to a walking passenger.) To recover the interrupted transaction, we need to record a preprocessor id in META.LOCK on object and an object identifier on the preprocessor side. These loggings achieved the isolation of the interrupted transaction.

4.5 Access Control

To control access permissions, we statically defined four roles: ticket creator, ticket organizer, ticket owner and others. All properties in an object are set with the "read/write" permissions for each role. With respect to the authentication of the roles, the creator and the organizer can be authenticated by password in meta properties on each object, while the owner is authenticated by password registered on the ticket wallet.

The retrieval of the full list of objects is granted to only the owner of the wallet. Furthermore, we added password option to the select query for the creator and the organizer to find objects that they manage.

```
select oid by Username=Password
```

This authentication limits the scope of lists by the roles, and therefore enables the minimum privacy for the owners. In the experimentation, we didn't experience performance disadvantage by the authentication option, because the number of tickets stored on the wallet was limited less than eight. The remaining problem is how to design dynamic and flexible roles that model more complicated enterprise relationships.

5. Conclusion

Ubiquitous DBMSs will bring high impact on data application integrations in the context of real world. This paper discussed four properties (interoperability, consistency, concurrency and privacy) required in our digital ticket experimentation. These properties will provide ubiquitous database with the possibility to implement various card applications in today's social systems. The other possibilities include ease of application developments (c.f., a mobile code in MULTOS and JavaCard) and ease of data application integrations. This paper also presented the new style of ubiquitous database, in which information can be shared directly from real world "objects". In the future, developing a very small DBMS on the smartcard itself, we will continue to explore new ubiquitous data applications to refine the functionality of the database.

Acknowledgement

The authors would like to thank Dr. N. Koshizuka and Mr. Y. Imada (University of Tokyo) for useful comments and suggestions that helped in improving an earlier draft of this paper.

Reference

[1] S. Abiteboul, P. Buneman and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers, 1999.

[2] C. Bobineau, L. Bouganim, P. Pucheral, P Valduriez. PicoDMBS: Scaling Down Database Techniques for the Smartcard. In *Proceedings of VLDB 2000*, pages11-20, 2000.

[3] P. Bonnet and P. Seshadri. Device Database Systems. In *Proceedings of 16th ICDE*, pages 194, 2000.

[4] S. Fickas, G. Kortuem, and Z. Segall. Software Organization for Dynamic and Adaptable Wearble Systems. In *Proceedings of* ICWC'97, October 1997.

[5] U. Hansmann eds. *Smart Card Application Development Using Java*. Springer Verlag, 1999.

[6] K. Kangas and J. Roning. Using Code Mobility to Create Ubiquitous and Active Augmented Reality in Mobile Computing. In *Proceeding of Mobicom '99*. pages 48-58, 1999.

[7] K. Kuramitsu, H. Matsuda, T. Murakami and K. Sakamura. TTP: Secure ACID Transfer Protocol for Electronic Ticket between Personal Tamper-Proof Devices. In *Proceedings of IEEE COMPSAC2000*, Oct. 2000.

[8] S. Ortiz. Industry Trends Embedded Databases Come out of Hiding. *IEEE COMPUTER*, pages 16-19, 33(3), 2000.

[9] T. Starner, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R. W. Picard, and A. Pentland. Augmented Reality through Wearable Computing. *Presence*, Vol. 6(4), pages 386-398, 1997.

[10] M. Weiser. The Computer for the 21st Century. *Scientific American*, pages 66-75, September 1991.

[11] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communication of the ACM*, Vol. 36 (7), pages 75-84, 1993.