SMART CARD DATA STREAM MANAGEMENT SYSTEM BASED ON CONTEXT AWARENESS

by

Seda Polat

BSc, in Computer Engineering, Bogazici University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2010

SMART CARD DATA STREAM MANAGEMENT SYSTEM BASED ON CONTEXT AWARENESS

APPROVED BY:

Prof. Taflan İ. Gündem (Thesis Supervisor)	.....
Prof. Mehmet U. Çağlayan	.....
Dr. Ahmet F. Mustaçoğlu	.....

DATE OF APPROVAL: 12.07.2010

ACKNOWLEDGEMENTS

Firstly and the most importantly, I thank my advisor Prof. Taflan Gündem, for his great support during my studies. We had discussions both technically and theorically about this work that were really useful for me.

I thank Orçun Ertuğrul, Ahmet Yıldırım and Ercan Doğan for their support about SCDSMS implementation environment and thesis documentation environment. Their comments on this thesis, friendship and support were very important for me.

I thank Mustafa Başak and all members of AKiS Project for their patience while I was at school instead of being at work with them.

Finally, I thank my mother Şengül Polat, my father Tuncer Polat and my sister Serim Polat for their endless love and support that I can always feel.

ABSTRACT

SMART CARD DATA STREAM MANAGEMENT SYSTEM BASED ON CONTEXT AWARENESS

Smartcard Technology is becoming very popular than before via the extended applications on it. They are widely used in many areas in the world. Health Care Systems, Banking Systems, Mobile Communication, NFC (Near Field Communication) Applications, Travel Payment Systems, National ID Card Systems, Electronic Passport Applications and many other areas can be counted that use smartcards as its main device.

In this project, a new system for this widely used little, portable and useful devices is offered. This system is a "Data Stream Management System based on a Context Awareness on Smart Cards" called Smart Card Data Stream Management System (SCDSMS). SCDSMS is a context aware database management system for smart cards.

There is not any research about data stream management for smart cards. But there are a few database management for smart cards papers. We also examine some papers about context awareness and data stream management.

Some database systems have been designed for smartcards for general or specific usages. All these database management systems are designed as conventional database systems (all data is stored in card and one time query execution applied on it) whereas this paper is opening a new area in smart card researches by using data streaming on smart cards.

Context aware systems are also used by many systems except the smartcards. Smart cards are very important application device for context aware systems because

they hold private data of user that can compose the context data of the person.

In our project there is not a huge database storage need because the data that is queried is stream data coming from a source into the smart card and immediately deleted just after the query execution on the tuple. Only very few the required results are stored in the card after query execution.

In applications, there will be a data stream source which sends stream data continuously to the smart card. Smart card has a query inside, which is taken just before the data streaming from the same source. According to that query, data stream tuples are queried continuously. Queries also use the context data of the user inside the card. These context data may be private and using them in query is a good strategy whereas exporting them outside of the card. While querying tuples, some results according to the query are stored in the card. So, after data streaming is completed, the result file can be exported outside of the card. This result file is very small regarding to the incoming data size.

Consequently, we produced and executed a SCDSMS-beta on a real smart card to evaluate the performance and see the applicability of SCDSMS.

ÖZET

AKILLI KART DURUMDAN HABERDAR AKAN VERİ YÖNETİM SİSTEMİ

Akıllı kart teknolojisi, gelişen uygulamalarıyla artık daha da popüler. Bu kartlar dünyada bir çok alanda kullanılmakta. Bunlardan bazıları şunlardır; Sağlık Sistemleri, Bankacılık Sistemleri, Taşınabilir İletişim Sistemleri, NFC (Near Field Communication - Yakın Alan İletişimi) Uygulamaları, Ulaşım Ödeme Sistemleri, Ulusal Kimlik Kartı Uygulamaları, Elektronik Pasaport Sistemleri.

Bu projede, küçük ,taşınabilir aletler olan akıllı kartlar için yeni bir sistem tasarlanmıştır.Bu sistemin bir Akıllı Kart Durumdan Haberdar Akan Veri Yönetim Sistemi'dir ve adı Akıllı Kart Akan Veri Yönetim Sistemi'nin ingilizce kısaltması olan SCDSMS olarak verilmiştir.

Akıllı kartlarda akan veri yönetimi konusunda literatürde herhangi bir çalışma bulunmamaktadır. Fakat, bu konuyla alakalı olarak akıllı kartlarda veri yönetimi konusunda bazı çalışmalar vardır. Bu çalışmalar ve diğer ilgili konular olan Durumdan Haberdar Olma ve Akan Veri Yönetimi konularında da bazı çalışmalar incelenmiştir.

Şimdiye kadar, akıllı kartlarda özel veya genel uygulamalar için bazı veritabanı yönetim sistemleri tasarlanmıştır. Bu sistemler klasik veritabanlarını yönetmek içindir yani tüm veri kartta hali hazırda saklanmaktadır ve sorgular bu sabit veritabanı üzerinde bir kez koşturulur. Diğer yandan, akan veri yönetim sistemlerinde sabit veritabanı üzerinde sorgu koşturulmaz, bunun yerine gelen her veri üzerinde sorgu koşturulur ve sadece sorgu sonucunda istenen veriler kartta tutulur. Bu bakımdan SCDSMS akıllı kart araştırmalarında yeni bir konu açmaktadır.

Durumdan Haberdar sistemler, akıllı kartlar haricinde birçok uygulamada kullanılmaktadır. Akıllı kartlar durumdan haberdar uygulamalar için çok uygun araçlardır çünkü kişinin özel bilgileri kartta güvenli bir şekilde tutulurken istenen sorguların cevapları için bu özel bilgiler kullanılarak alınabilir, böylece kişisel bilgiler dışarı çıkmadan, kişinin durumu için özel işlem yapılabilir.

Akıllı kartlarda büyük bir sabit veritabanını tutacak alan yoktur bu yüzden bizim sistemimizdeki gibi akan veriyi gelir gelmez önceden girilmiş sorguyla işleme tabi tutup sonra karttan silmek gayet uygulanabilir bir yöntemdir. Sorgu sonucuna göre sadece istenen kriterlerdeki veriler karta sabit olarak kaydedilebilir, diğer veriler daha sonra ihtiyaç duyulmayacak veriler olduğu için silinir.

Uygulamada, SCDSMS içeren kart tek kanaldan akan veri alabilir. Bu veri iletimi başlamadan önce kaynaktan gelen bir komutla SCDSMS başlatılırken, sorgu cümlesi de sisteme bildirilir. SCDSMS bu sorguyu gelen veri üzerinde nasıl işleyeceğini planladıktan sonra veri kaynağına veri aktarımına başlanabilir cevabı döner. Bu durumda veri aktarımı başlatılır ve SCDSMS gelen veriyi planladığı için sorgular. Bu sorgular kart içerisinde daha önceden saklanmış olan kullanıcı bilgilerinden de faydalanarak durumdan haberdarlk özelliğini de gösterebilir. Sorgulama sonucunda gerekli veriler karttaki sonuç dosyasında kalıcı olarak saklanır. Bu sonuç dosyası, sorgulama bittikten sonra okunabilir ve bu dosya karta giren veri büyüklüğüne göre çok küçük boyuttadır. Bu da kartın veri saklama kısıtıyla başa çıkabilmek için önemli bir özelliktir.

Bu çalışmada, SCDSMS olarak adlandırılan Akıllı Kart Akan Veri Yönetim Sistemi'nin ayrıntılı tasarımı anlatılmaktadır. Ayrıca bu sistemin uygulanabilirliğini ve performansını görebilmek için bir de pilot çalışma olan SCDSMS-beta sistemi kart içine kodlanarak çalıştırılmış ve performans sonuçları verilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

P1	Parameter 1
P2	Parameter 2
SW1	Status Word 1
SW2	Status Word 2
*	Address of pointed data

AKiS	Smart Card Operating System
APDU	Application Protocol Data Unit
COS	Card Operating System
DF	Directory File
EEPROM	Electronically Erasable Programmable Read-Only Memory
EF	Elementary File
LC	Length of Command
LE	Length of Expectance
RAM	Random Access Memory
ROM	Read Only Memory
PC	Personal Computer
SCDSMS	Smart Card Data Stream Management System
TUBİTAK	Scientific and Technological Research Council of Turkey
UEKAE	The National Research Institute of Electronics and Cryptology

1. INTRODUCTION

In this thesis, I proposed a system called SCDSMS which is an abbreviation of "Smart Card Data Stream Management System". This system runs completely on smart cards so it has so many restrictions rather than the PC programs. Today's standard smart cards have nearly 160KB ROM, 160 KB EEPROM and 6K RAM. SCDSMS's bottleneck is the tiny RAM of smart card.

In SCDSMS, there is not a huge database storage system because the data queried are stream data coming from a single stream source to the smart card and every incoming tuple is immediately deleted just after the query execution. Only some required results are stored in the card after query execution over them. Up to now, a few database systems are designed for smartcards for general or specific usages. All these database management systems are designed as conventional database systems (all data are stored in card and one time query execution applied on it) whereas this thesis opens a new subject for smart card researches by using data streaming on smart cards.

Context aware systems are also used by many systems except the smartcards. Smart cards are important application devices for context aware systems because smart cards hold private data of user that can compose the context data of the person. SCDSMS uses the Context Awareness issue that can be used by many applications and some of them are described in applications part below.

In applications, there will be a data stream source which sends stream data continuously to the smart card. Smart card has a query inside, which is taken just before the data streaming from the same source. According to that query, data stream tuples are queried continuously. Queries also use the context data of the user inside the card. These context data may be private and using them in query is a good strategy whereas exporting them outside of the card. While querying tuples, some results according to the query are stored in the card. So, after data streaming is completed, the result file can be exported outside of the card. This result file is very

small regarding to the incoming data size. SCDSMS is composed of 3 parts which are Context Manager, Storage Manager, Query Manager.

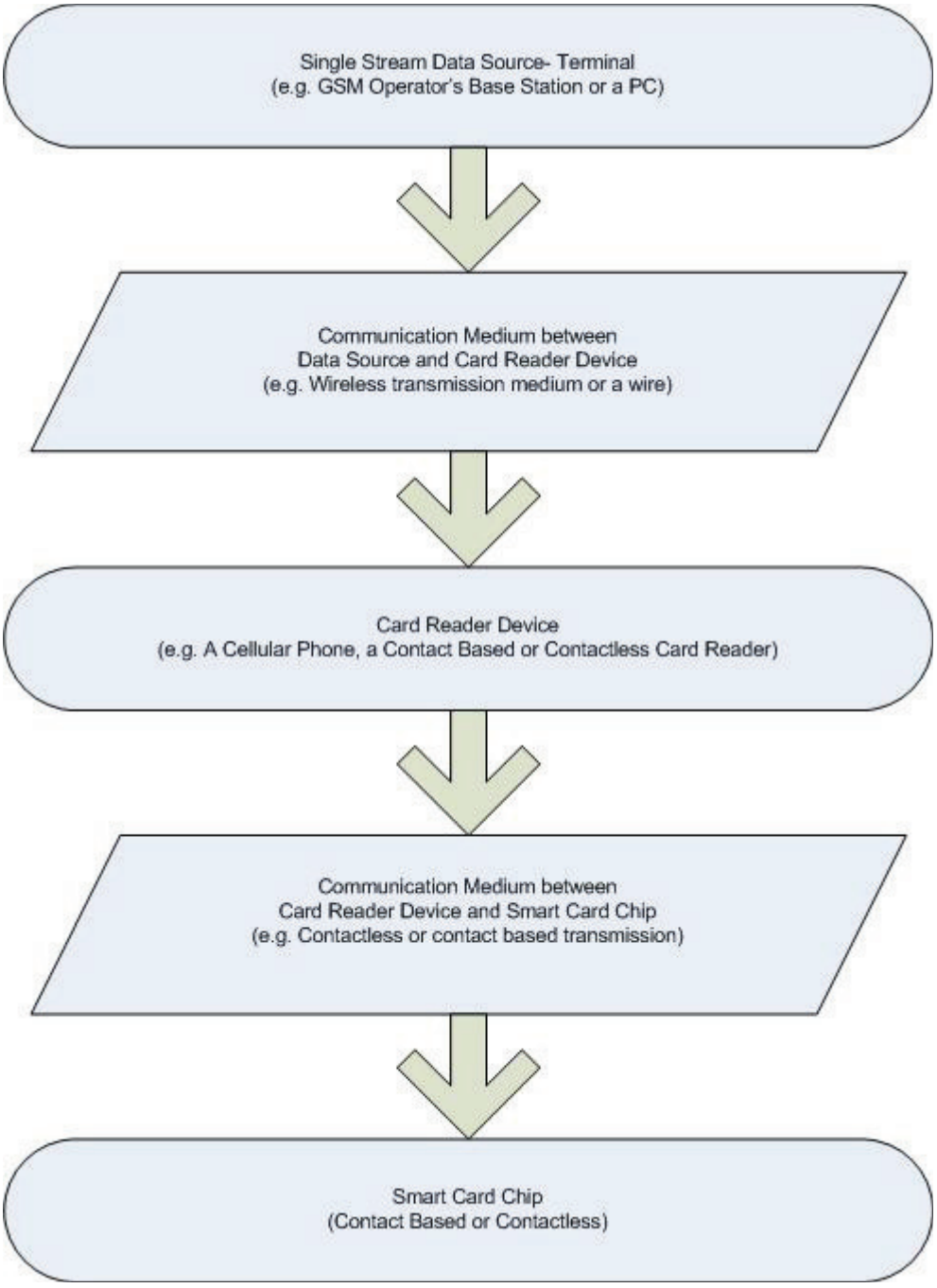

Figure 1.1. Smart Card Communication Overview

Because of the smart cards' limitations, SCDSMS should consider these constraints and be feasible for smart cards. These limitations can be given briefly as here:

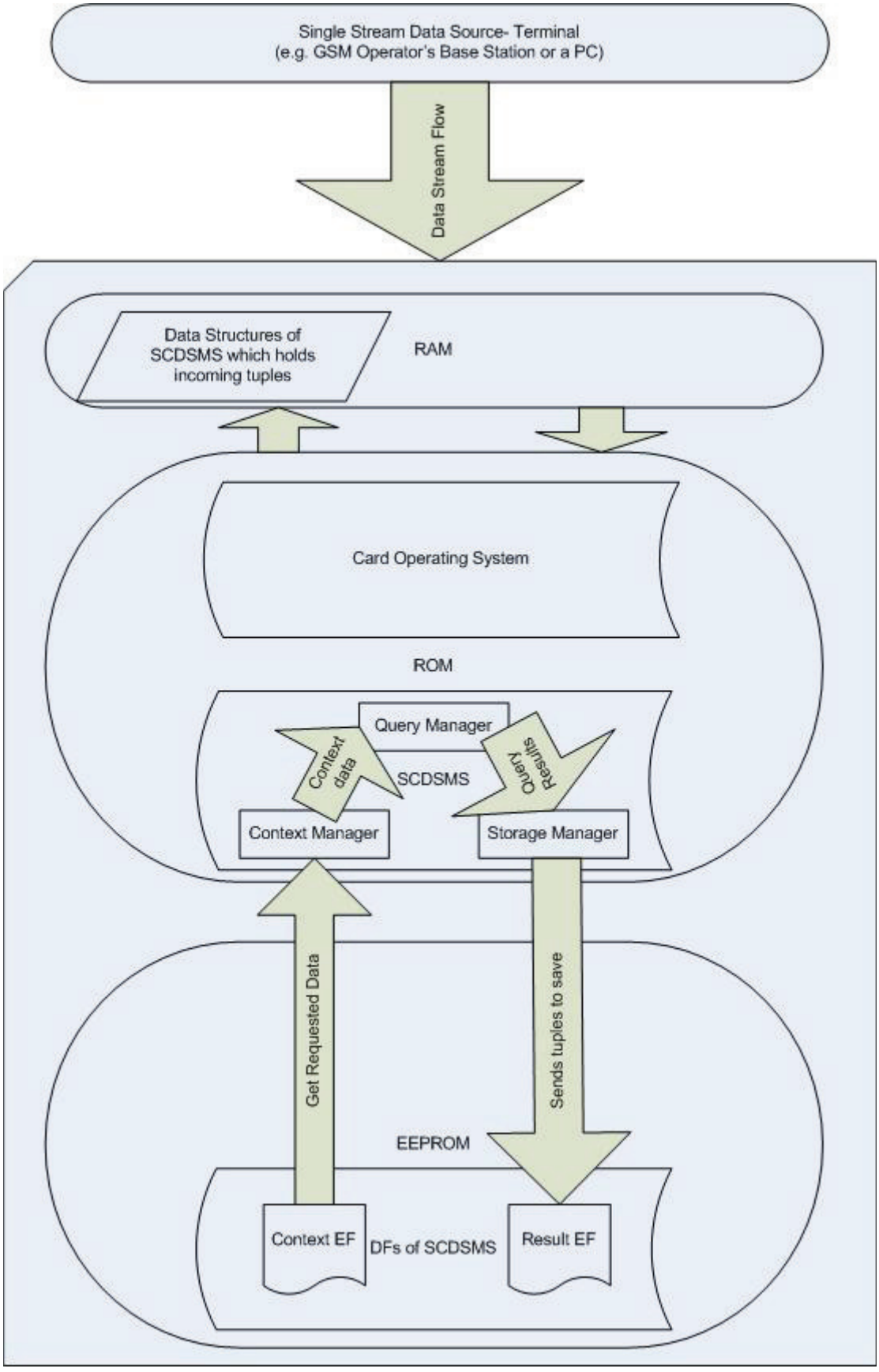

Figure 1.2. SCDSMS Organization in the Card

- RAM limitation (very small RAM compared to PCs, up to 10K)
- EEPROM limitation (very small stable storage compared to PCs, up to 160K)
- EEPROM write limitation (write operation in EEPROM is very costly in time)
- Power limitation (smart cards does not have a power supply inside)
- Command-Response working principle of smart cards
- Limited command size approximately 256 byte

So, for smart cards both database management system and the stream data management system can not be implemented as on the PCs. This thesis aim is to minimize stream data management system functions and optimize them to fit these limitations. Opposite to the the conventional database management systems, we can not use huge data structures like hash tables and big trees in SCDSMS due to the RAM limitation. Besides we can not use some operations that are not easy to run on smart cards like aggregation or join. Also, we should not use some needless operations like view operation in so many conventional database systems , these kind of operations can be implemented in a system that runs on a PC which is connected with the smart card and serves more functionalities than in the smart card. When some data wanted to be viewed with these options, card can be connected to a PC which has a SCDSMS-PC suite and required data can be viewed or some other additional operations can be applied on the data which can not be done only with SCDSMS. For these cases a SCDSMS-PC Suite can be implemented easily thanks to the PC's huge resources as RAM and processor capability.

In this thesis, SCDSMS is described in detail. Its modules, operators, limitations are given in below parts. Also some application examples are explained which can be used in many areas with today's technology. Finally, the challenging algorithms in SCDSMS are implemented as SCDSMS-beta and executed using a native COS called AKiS, which is implemented in TUBITAK-UEKAE institute. The trivial parts of SCDSMS are not implemented but their estimated RAM and EEPROM usages are calculated and it is seen that there is not any obstacle to implement these parts also in smart card. In conclusion, SCDSMS's performance and usability is given, also future works are considered.

2. PRELIMINARY INFORMATION

In this part, some preliminary information about Smart Cards, Data Streaming and Context Awareness are given. This information is important to understand the work done in the thesis which is described in next parts.

2.1. Smart Card Technology

Smartcard Technology is becoming very popular than before via the extended applications on it. They are widely used in many areas in the world. Health Care Systems, Banking Systems, Mobile Communication [1], NFC (Near Field Communication) Applications [2], [3], Travel Payment Systems, National ID Card Systems, Electronic Passport Applications and many other areas as mentioned in papers [4], [5] can be counted that use smartcards as its main device.

Smart Cards are widely used tiny devices. Their features, constraints and advantages are explained in subparts of this section. Before talking about its features, it is better to give a table to show how many people use these cards in different applications during the past years as also mentioned in [6].

The Table 2.1 shows the global smart card usage in different areas in 1996 which is taken from [7]. The Table 2.2 shows the World Wide Chip Market evolution through years which is taken from [7].

Thanks to the new technological expansions, smart cards usage increases through the years with a high acceleration. It is not hard to say that it may be over 10 Billion today. These widely used devices are very important due to its application areas. They should be so secure that it holds private information about users. In next parts of this section its facilities and security features are explained.

Table 2.1. Smart Card Usage Numbers in Applications

Usage Area	Number of Users
Phone Cards	575 Million
GSM Cards	15 Million
Financial	36 Million
Data/ID	30 Million
Pay TV	17 Million
Other	3.8 Million

Table 2.2. Smart Card Usage Numbers in Years

Years	1997	1998	1999	2000	2001	2002
Units (Million)	962.4	1290.7	2015.7	2886.5	3837.2	4716.7

2.1.1. What is Smart Card?

Smart Card which is also called as Chip Card or Integrated Circuit Card is SIM card sized or Credit card sized plastic card with a integrated circuit inside.

There are two types of smart cards:

- Memory Cards
- Microprocessor Cards

Memory Cards only include a non-volatile memory inside, whereas a Microprocessor Card has a microprocessor, non-volatile storage, RAM, and a optional crypto engine in it. So, it is not a mistake to say a microprocessor card is a tiny computer. Besides, cards do not contain batteries; energy is supplied by the card reader. Smarts cards may have up to 8 kilobytes of RAM, 346 kilobytes of ROM, 256 kilobytes of

programmable ROM, and a 16-bit microprocessor.

Figure 2.1. A SIM sized smart card

According to the type and application, smart card sizes may vary. Smart cards can be grouped in two due to their communication type:

- Contact Based Smart Cards
- Contactless Smart Cards

Contact Based Smart Cards can be SIM sized or in credit card size. All contact based smarts card can work in SIM size but because of some usage standards (e.g. Financial Usage : Name and Bank labels on the card) it should be in a bigger size.

Contactless Smart Cards should be bigger than SIM sized card because it should also have an antenna inside layer or the plastic card. So all contactless cards are in credit card size.

Figure 2.2. Credit Card with Chip

- VCC Power supply input.
- RST Reset signal, used to reset the card's communications.
- CLK Provides the card with a clock signal, from which data communications timing is derived.
- GND Ground (reference voltage).

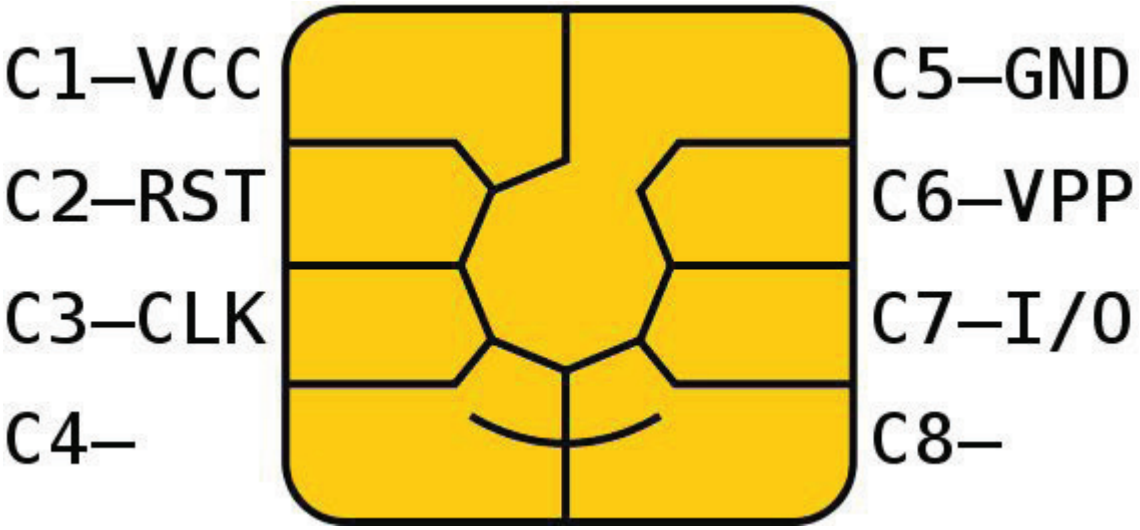

Figure 2.3. Signaling Elements of a Smart Card Chip

- VPP Programming voltage input - originally an input for a higher voltage to program persistent memory (e.g. EEPROM), but now deprecated.
- I/O Serial input and output (half-duplex).
- C4, C8 The two remaining contacts are AUX1 and AUX2 respectively, and used for USB interfaces and other uses.

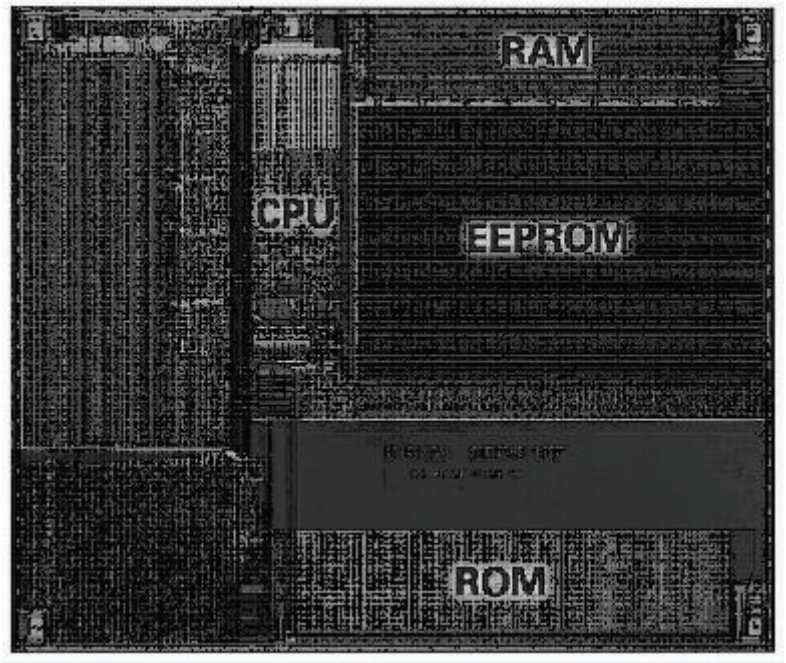

Figure 2.4. Organization in a Smart Card chip

2.1.2. History

Smart Cards' first mass use was as pay-phone cards in France in 1983 whereas its invention is in 1968 by two German engineers Helmut Grttrup and Jrgen Dethloff.

After this invention Memory Cards and Microprocessor cards are invented respectively in 1974 and 1977 by French inventor Roland Moreno and Michel Ugon from Honeywell Bull. In 1978, Bull patented the architecture to program the chip as "SPOM; Self Programmable One-Chip Microcomputer". Motorola used this patent three years later. Bull also has 1200 patent about smart card till 2001. In 2001 Bull sold its all patents to Schumberger which created its own smart card department and Axalto. In 2008, world's first and second biggest smart card companies Axalto and Gemplus are combined as Gemalto.

Besides this information second usage is seen in again France as Carte Bleu Debit Cards in 1992. In this application customers should enter their PIN using POS machines before the transaction begins, except small payments like Highway Tolls which does not require user's PIN verification for transaction.

In mid 1990s, throughout many Europe countries, smart card based electronic purse applications are widely used. In this system, POS machines do not require online connections with banks because debits are stored in the card.

Maybe the most important evolution in smart card history is the SIM cards invention used by GSM phones in Europe in 1990s. Thanks to this area, smart cards became more popular with widely used mobile phones.

Another important event is the co-operation in the banking system cards. In 1993, international payment brands MasterCard, Visa and Europay (EMV) developed and agreed on a standard for smart cards of electronic payment systems. First stable version of this standard is released in 1998. After that year upgrades of these specifications are released with the backward compatibility. Many countries payment systems uses card

in EMV standards to make their payment system interoperable with many countries in the world. Such countries like United States did not use EMV standards so long years. They used some other standards like American Express card. Thanks to the EMV standard all over the European countries, European citizens can use their EMV credit cards in other EMV standard user countries.

By the contactless smart cards some other usage areas arose besides the older usages also upgraded. For example in 2004 a new EMV specification is released for contactless credit and debit cards. Contactless smart cards do not require physical contact with the card reader. This feature makes smartcards useful for some more areas like ticketing application for mass transit and highway tolls.

In recent years, governments also use smartcards as Citizen (E-ID) cards, Driver License cards and Patient Cards. Electronic passport which is standardized by ICAO (International Civil Aviation Organization) is useful to make international transfer verifications in a more secure way.To sum up the history of smart card evolution the Table 2.3 may be useful.

2.1.3. Types

According to the card-reader communication medium, smart cards can be grouped in four:

- Contact based cards
- Contactless cards
- Hybrid cards
- Combi cards

Contact based cards communicate with reader using its contact area which is approximately 1 square centimeter with contact pads. When card is inserted into the reader, these pads provide electrically connectivity. Interoperability of card readers and contact based smart cards are provided using ISO 7810 and ISO 7816 standards. Thanks

Table 2.3. Smart Card Usage Numbers in Years

Year	Usage	Country	Detail
1968	No mass use in this year	Germany	German engineers Helmut Grttrup and Jrgen Dethloff and invented the automated chip card.
1974	No mass use in this year	France	Memory Card is invented.
1977	No mass use in this year	France	Microprocessor Card is invented.
1978	No mass use in this year	France	Bull received the SPOM patent.
1982			Inventors received a patent
1983	Pay Phone Card	France	
1992	Carte Bleu Debit Cards	France	First use fo Debit Cards with smart cards.
1993	Credit Card	Europe	EMV specifications are started to be released.
Mid 1990s	Electronic Purse Applications	Germany	After Germany, many European Countries used this applications.
By 2000s	Citizen Card, Patient Card	Malaysia, Germany	
By 2000s	Electronic Passports	Europe	Passports containing chips to secure the international transfers.
2001			Schlumberger got all the patents from Bull and created Axalto.
2004	Contactless Credit Card	US and Europe	New specification for contactless credit cards is released by EMVco.
2008		France	Biggest Smart Card firms Gemplus and Axalto is merged as Gemalto

to these standards :

- Physical shape and characteristics
- Electrical connector positions and shapes
- Electrical characteristics
- Communications protocols, including commands sent to and responses from the card
- Basic functionality

are implemented by card and reader producers to supply compatibility.

Contactless Smart Cards communicates with readers through RF Induction technology. Cards take energy from contactless reader using RF Induction. They use a inductor to receive radio-frequency signal. Then it rectifies the signal and use to supply power to the smart card's chip.

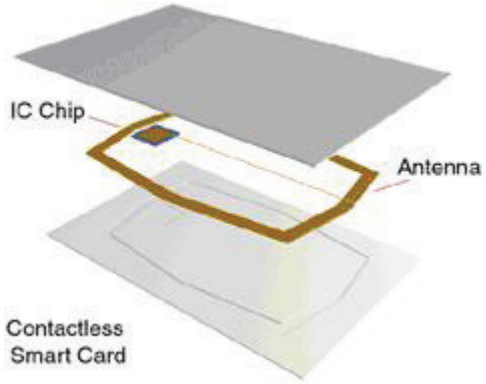

Figure 2.5. Contactless Smart Card Inlayer

Data rates are 106 to 848 kilobits/second. These cards have an antenna inside the plastic card different than the contact based cards. Antennas require close proximity to the readers. These cards are often used for quick and hands free transaction, like paying a mass transit placing wallet near the reader which includes contactless credit card. Another usage area is electronic purse applications. These applications provides user to pay quickly without entering PIN for small amount of payments.

Hybrid cards provide both contact based and contactless interface in one card. It includes two chip on the card that supplies both interfaces according to the ISO 14443 Type B standard. Some applications may need both interfaces in different phases.

Combi Cards provide same interfaces as in Hybrid cards but they have one chip in the card with both contact and contactless interfaces. So in Combi Cards same memory is read using both interfaces.

2.1.4. Standards

So many other examples can be given to imply the importance of interoperability of smart cards. Smart cards like credit card should be interoperable among the countries. Also all SIM card should fit all cellular phones. Here are some important standards used by all smart card , related device and software producers :

- ISO/IEC 7816 Identification cards - Integrated circuit cards
  - (i) 7816-1 Physical characteristics
  - (ii) 7816-2 Cards with contacts - Dimensions and location of the contacts
  - (iii) 7816-3 Cards with contacts - Electrical interface and transmission protocols
  - (iv) 7816-4 Organization, security and commands for interchange
  - (v) 7816-5 Registration of application providers
  - (vi) 7816-6 Interindustry data elements for interchange
  - (vii) 7816-7 Interindustry commands for Structured Card Query Language (SCQL)
  - (viii) 7816-8 Commands for security operations
  - (ix) 7816-9 Commands for card management
  - (x) 7816-10 Electronic signals and answer to reset for synchronous cards
  - (xi) 7816-11 Personal verification through biometric methods
  - (xii) 7816-12 Cards with contacts – USB electrical interface and operating procedures
  - (xiii) 7816-13 Commands for application management in multi-application environment
  - (xiv) 7816-14 Cryptographic information application

- ISO/IEC 14443 Identification cards - Contactless integrated circuit cards - Proximity cards
- ICAO 9303 Machine Readable Travel Documents

2.1.5. Components

Smart cards are used as electronic authentication keys, digital signs, GSM cards and bank cards. Also, they are used as electronic passports and e-government cards such as personal identification and health care cards.

Basically smart card consists of 3 main parts:

- Metallic unit on plastic material which is called plastic module (physical plastic card)
- Silicon chip located in the metallic unit on the plastic module. This chip consists of microprocessor, ROM, RAM, EEPROM and some hardware units (decoders, advanced crypto engine, RNG, MED, etc.)
- Operating system (written in ROM and enables the operation of card functions using hardware units)

Card Operating System (COS) is embedded in ROM during chip manufacturing and can't be changed afterwards. However, data can be written into EEPROM under operating system's control. COS will be located in a smart chip planted to a plastic card. The interface of the card to the outside world is over a smart card reader or access device such as POS (Point of Sale) machine. PC (over the smart card reader) or access device transmits the commands to the smart card. Incoming commands are interpreted by COS and the response is transmitted back to the access device or to the PC over smart card reader as in Figure 2.6.

The smart card has 8 pins according to the IEC/ISO 7816-2 [8] which is shown in Figure 2.3. Smart card communicates with reader via I/O pin. 2 pins are reserved for future use. In the past, smart card's EEPROM was being programmed by Vpp pin

Figure 2.6. COS's environment

which is not used anymore. VCC, GND, RST and CLK pins are used to operate the smartcard.

2.1.6. Communication Protocols

The communication between smart card, card reader and the computer starts when the card is inserted into the card reader. After the card reader senses the smart card, it supplies voltage through the Vcc pin. Afterwards, RST and CLK signals are applied to the smart card and the ATR (Answer to Reset) message is waited from the card. With Vcc, RST and CLK, COS prepares the memory and sends an ATR message. ATR message includes communication data such as protocol name, block waiting time, character waiting time, etc.

In Table 2.5, commands that does not need data exchange can use this format. In Table 2.6, commands that only retrieves data from card can use this format..In Table 2.7, commands that only sends data to card can use this format. In Table 2.8, commands that sends and retrieves data can use this format . If Le = 0, then the number of bytes expected is unspecified and must be provided by the smart card (maximum 256 bytes). In Table 2.9, return value is in ODATA, SW1 and SW2 gives the return code of command.

When ATR is received successfully by the card reader and the computer, COS is ready to accept and run the incoming commands. Smart card receives the incoming commands over I/O pin according to T = 1 protocol. Incoming commands are interpreted by COS and the response is transmitted back to the computer with the same

Table 2.4. Command Fields

CLA	A class of instructions. The values of some class bytes can have a specific meaning to a certain class of commands.
INS	A particular instruction
P1 & P2	The parameters for the instruction.
LC	The number of bytes that the terminal will send to the card.
LE	The number of bytes that the terminal expects to receive from the card.
DATA	Bytes that the terminal send to the card.
SW1 & SW2	First status byte and Second status byte. They are predefined public variables of type byte. Before a command is executed, they have the values &H90 and &H00, which is a standard status code meaning, "Command successfully completed". If it is wanted to return an error code to the caller, SW1 and SW2 are set to the appropriate values before exiting the command.
ODATA	Data returned by command.

Table 2.5. APDU type 1

CLA	INS	P1	P2
00H	01H	01H	01H

Table 2.6. APDU type 2

CLA	INS	P1	P2	LE
00H	02H	01H	02H	40H

Table 2.7. APDU type 3

CLA	INS	P1	P2	LC	Data
00H	02H	01H	02H	01H	99H

protocol over the card reader. The communication is unidirectional and at 9600 baud rate. Smart card communicates with TPDU (Transmission Protocol Data Unit) packets according to the IEC/ISO 7816-4 [9]. The commands are sent to the smart card in the APDU (Application Protocol Data Unit) format. TPDU and APDU structures are shown in Table 2.4. The details of ATR are described in the ISO 7816-3 standard.

Table 2.8. APDU type 4

CLA	INS	P1	P2	LC	Data	LE
00H	01H	01H	02H	02H	0AH 36H	20H

Table 2.9. Response APDU

ODATA	SW1	SW2
01H 57H	90H	00H

2.1.7. File System

Standards related to the smart card file system are determined by ISO-7816-4. Smart card file system has a root that is called the master file (MF). Directories of file system are called dedicated files (DF). Normal files are called elementary files (EF).

Files are referenced by a file identifier (FID), which is two bytes long. There are several kinds of elementary files:

- Transparent files, which are seen as a sequence of bytes.

- Linear fixed files, which are seen as a sequence of fixed-length records.
- Linear variable files, which are seen as a sequence of variable-length records.
- Cyclic files, which are seen as an endless sequence of fixed-size records.

Files in the smart card file system are accessed by means of standard file system commands. File system commands perform some operations in the smart card file system such as file open, read, write etc. The application program running on the terminal can access the files using APDU commands if there is access right required to access those files.

2.1.8. Usage Areas

Smart Cards are widely used devices as mentioned in [6] and some applications that use smart cards are listed and briefly explained below.

2.1.8.1. Financial Services. Many paying systems use smart cards. Banks use credit cards and debit cards which are smart cards. Smart cards can be used as electronic wallet which is loaded with funds to use for small amount and quick payments like payments of transportation applications, vending machines. Cryptographic protocols exchange the data in a secure way.

2.1.8.2. Health Application. In medical services, insurance firms or government provide smart cards to store some critical data about patients and authenticate the patients to system. This system improves security and privacy of patient's information. Thanks to smart card portability feature, medical records or patient information can be carried securely.

2.1.8.3. Identification. In these applications cards authenticate the holder's identity via PIN or finger print. Most used PKI cards includes certificates of user in it, this provides verification of a trusted certificate. Generally, these cards are given by the government as ID cards and may include driver license, emergency medical information,

e-wallet for transportation.

2.1.8.4. Computer Security. Secure network connection can be applied using smart cards. Web browsers, for SSL connections, may need certificates which are stored in the smart card. Operating systems use smart card log-on option for secure single sign on operations.

2.1.8.5. Cellular Phones. All mobile phones use SIM cards which are small size smart cards. Main aim of smart card in cellular phones is to authenticate user to the operator's system securely. Besides operators control user information and upload application independently the mobile phone device's brand or model.

2.2. Data Streaming

This thesis proposed system SCDSMS receives data stream as input. So, it is important to understand data streaming properties and applications. In this section data streaming is explained.

2.2.1. What is Data Streaming?

Data Stream is group of digital data packets which are continuously transmitting from an electronically source to a receiver. The data streaming continuously to the receiver is too big to store all of them in the receiver's storage. Applications that data streaming is used, need a data stream handling system in the receiver side in order to drop processed data. According to the application, data stream handling system may be like a data base management system. Applications of data streaming can be found in finance, web applications, security, networking, and sensor monitoring.

While talking about data stream, first data type comes to the mind is obviously media stream of televisions or radios. These streams are just processed on the receiver side without eliminating some of them. Over these media stream, no queries

are executed like in below applications. So, it is important to understand and express the difference between the media stream and data stream. Handling both streams are different processes. Media streams are just presented without querying on the receiver side, whereas data streams are continuously queried on the receiver computer.

A data stream can be represented as S which includes ”s” and ”t”. Where ”s” represents sequence of tuples and ”t ” represents the packet sending time. ”S” stream is taken by the receiver and processed according to its timestamp ”t”.

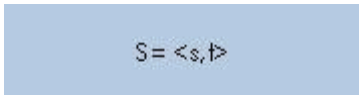

Figure 2.7. Stream Elements

2.2.2. Applications

Stream data can be found in financial applications. As stock tickers and news feeds, there are always new data packets in finance applications. There are many programs that handle the stream financial data and retrieve searched information among the incoming data. Traderbot is a example application that is a web-based financial search engine which executes continuous queries over streaming financial data.

Over network packets, some security application can be used in front of a secure cluster in the network via security programs like iPolicy Networks. This software provides firewall support and intrusion detection over multi-gigabit network packet streams. This kind of programs need complex query processing capabilities including URL filtering based on table lookups and computing the correlation across multiple network traffic flows.

Some web applications also produce data streams. For example large web sites may want to monitor user operations, like click streams to analyze heavily accessed web pages and some logs and statistics about it. Yahoo uses some kind of application on its servers to monitor its real performance.

There are so many applications of stream data in sensor networks. Sensor networks carry continuously measurement data from sensors to servers. Sensors continuously measure something and generate packets including these measurements then send them to the server to be analyzed. These kinds of network flows are data streaming packets and need to be queried on server side just after received by the server.

2.2.3. Data Stream Management Systems

Data streams are different than the conventional databases. So, conventional Data Base Management Systems cannot process stream data. Some differences of data stream then the conventional data are:

- Data Stream packets arrive online.
- There is no control of process order over the stream data.
- Data Streams are potentially unbounded size.
- After processing the tuples of data stream are discarded or archived.

Data Stream Management Systems (DSMS) handle incoming data streams. Data streams are continuously queried on the receiver computer side to search some information or eliminate some tuples among the incoming tuples in stream. Queries are also different from the queries in conventional database management systems (DBMS).

Main distinction is conventional systems use "One-time Queries" whereas DSMSs use "Continuous Queries". One-time queries are evaluated once over a point-in time snapshot of the data set, with the answer returned to the user. On the other hand, Continuous queries are evaluated continuously as data streams continue to arrive. The answer to a continuous query is produced over time, always reflecting the stream data seen so far. Continuous query answers may be stored and updated as new data arrive, or they may be produced as data streams themselves.

Other distinction of queries of DBMSs and DSMSs is predefined and ad-hoc query types. Predefined queries are used in DSMSs where ad hoc queries are used in both

DSMSs and DBMSs. Predefined queries are the queries which are set before data streaming begins and continuously applied over the data while streaming period. Ad-hoc queries are set during the streaming period and it is more complex to handle than the predefined queries. For DBMSs ad-hoc queries are one-time queries so it is not hard to execute. Unfortunately, in DSMSs, ad-hoc queries create complexity because they come during the execution and it is not so easy to rapidly adapt new query over the incoming data without missing not queried data packets.

According to the data rates and applications, some tuples can be dropped without queried. If data rate is so high and application allows tuples dropping when necessary, then an algorithm can decide to drop tuples to manage other tuples easily. Also, buffering is used in DSMSs due to the receiver's storage and processing constraints.

2.3. Context Awareness

Context Awareness issues is used in SCDSMS as user's private data stored in the smart card. Queries of SCDSMS may require this private data sometimes. For example, data stream tuples may be eliminated according to the user's job or age. For financial applications, stock exchange news are taken according to the user's interested lots. The interested lots list is stored in the smart card SCDSMS application directory. When news feeds starts to streaming to the user's smart card (maybe In the mobile phone as SIM card), user's smart card eliminates and processes over the lots that are in the interested list of that user.

In this manner, context awareness issue is very important for SCDSMS. Under this section context awareness issue and context aware systems are explained to make readers understand the context operations and Context Manager in SCDSMS easily.

2.3.1. What is Context Awareness?

Context Awareness is a subject that is widely used in computer based applications which supplies information about the user to make software operate according to the

user's special cases.

Context Aware systems are used by many applications, especially by mobile applications. Due to the changed physical environment and geographical location mobile news feeds change thanks to the context aware systems. For example if you are in Istanbul, your GSM operator sends Istanbul's weather forecast information to you, not Ankara's. This is a kind of simple context aware system that uses the user's location information hold in GSM operator's database.

Common context awareness attributes are:

- Location
- Identity
- Activity
- Time

As the user's activity and location are crucial for many applications, context awareness has been focused more deeply in the research fields of location awareness and activity recognition.

2.3.2. Applications

Context Aware systems generally uses location information and this demand is increasing due to the mobile devices are began to be used widespread. Various tourist guide projects can be given as example for location aware systems. Position or proximity information for these systems can be gathered through the GPS satellites, mobile phone towers, badge proximity detectors, cameras, magnetic card readers, barcode readers, etc. Other examples can be found in Espinoza et al. (2001), Priyantha et al. (2000), Burrell and Gay (2002) and Kerer et al. (2004).

3. RELATED WORK

3.1. Smart Card Database Systems

There are a few numbers of studies about the database systems for smartcards or some other lightweight devices. In this section these papers are briefly explained in the manner of data structures, query processing methods, optimizations and applications.

The most important of all is "Pico DBMS: Scaling Down Database Techniques for the Smartcard" [10]. In this paper a compact database management system is proposed for smartcards. They have stated the smartcard constraints, applications and a possible need for a database management system. PicoDBMS implements the minimum functionality that is strictly needed to manage the data securely in the smartcard. Other components of traditional DBMSs can be implement on the terminal side, which are can be GUI, sort operators and etc.

PicoDBMS implements these parts of DBMS in smartcard:

- Storage manager: Manages the storage of the database and the associated indices.
- Query manager: Processes query plans composed of select, project, join and aggregates.
- Transaction manager: Enforces the ACID properties and participates in distributed transactions.
- Access right manager: Provides access rights on base data and on complex user-defined views.

Details about Pico DBMS's organization are given in next parts.

Another paper that explains logical design issues about a possible the smart card database system is "Logical and Physical Design Issues for Smart Card Databases" [11]. It describes the storage mechanism of database in the smartcard in an effective

way.

The paper "DELite: Database Support for Embedded Lightweight Devices" [12] describes very briefly the lightweight device database design like smart cards. They talk about the storage constraints and RAM constraints of these devices so the selected data structures and algorithms should satisfy these issues.

Also in 2006 a report is published by the PicoDBMS authors called "Smart Card DBMS: where are we now?" [13]which includes the experimental results that are applied on a smart card. So, it gives new results than before due to the experimental results.

ISO 7816-7 [14] document can be another reference for these studies because it expresses the standards of SQL for smart card which is called CSQL (Card SQL). CSQL implementation issues depend on the DBMS application on smart card. Standard only defines the interfaces between application and users. The query language is explained in detail, the inputs and outputs of them are standardized.

To sum up, there are three main researches about smart card DBMS is done and other references can be ISO 7816-7 and update reports like in PicoDBMS report. These 3 papers are handled in next parts to show how the design issues are handled up to today.

3.1.1. Data Storage Models

PicoDBMS gives three types of storage model:

- Flat Storage
- Domain Storage
- Ring Storage

Flat storage is the trivial method where tuples are stored sequentially and attribute values are embedded in the tuples. This method creates a big space consuming.

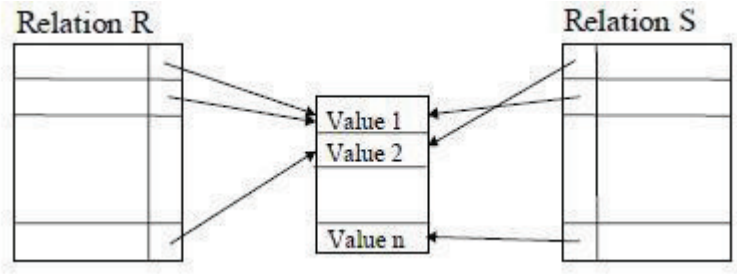

Figure 3.1. Domain Storage

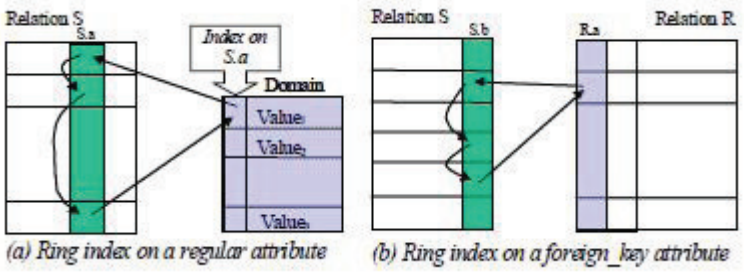

Figure 3.2. Ring Storage

Besides, it is inefficient due to the lack of indexation.

Domain Storage uses pointers to the values that are hold in a pool, and can be pointed from multiple tuple. This method is not useful if attributes are not duplicated.

Ring Storage also uses pointers and proposes a new storage model called Ring Storage for smartcards that combines data and index storage in a single structure. PicoDBMS proposes the Ring Storage method that prevents too many pointers to the value.

3.1.2. Query Plans

PicoDBMS[10] uses a extreme right deep tree for its query execution plan. Other alternatives that are explained in PicoDBMS study are "Left Deep Tree", "Right Deep Tree" and "Bushy Tree". All these tree structures uses RAM are very inefficiently whereas Extreme Right Deep Tree do not use such more space in RAM due to No Materialization in middle steps.

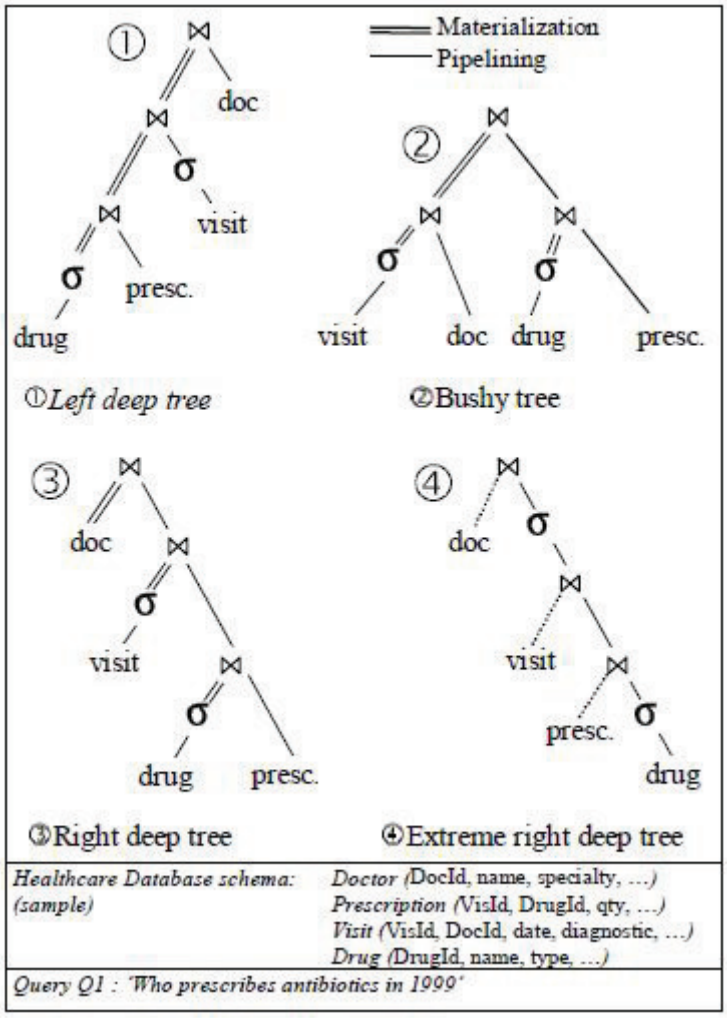

Figure 3.3. Query Execution Plans of PicoDBMS

3.1.3. Optimizations

Some optimization for RAM usage and EEPROM usage are explained in PicoDBMS[10]. The ideas behind these small optimizations are :

- Ring storage type to make EEPROM usage lower.
- Extreme Right Tree query execution plan to make RAM usage lower.

These methods are briefly given above and can be found in more detail in the referenced paper.

3.1.4. Applications

PicoDBMS analyzed the requirements based on a Health care Applications which need patient folders in the card and some database management requirements.

Health cards are used in such countries: France, Germany, USA, Russia, and Korea. The initial idea was to give ID card to each citizen which can hold ID information, insurance information. As the increasing storage capability of smartcards, the information held in the card can be extended to emergency data, surgical operations, doctors and even heavier data like X-Ray examination, scanner images, measurement results which are previously hold in the hospital's servers.

By holding these data in the heal cards, only authenticated users can access the private data whereas public or emergency data can be accessed without authentication. The data in the card can be used by doctor, surgeons, pharmacists, insurance agents and user herself. As there are so many users of the database there should be an database management system on the card because these persons may find out complex relations in this database which requires a complex query. Also, amount of data is getting bigger which need more advanced methods to be queried.

Using DBMS on the card supplies these advantages for the health card case and

also for other usages:

- **Holding database in card:**
  (i) Data will be highly available when needed. (Anywhere, anytime, without asking surgery information to the Hospital to look up its servers)
  (ii) Storing data in servers (of Hospital) may hurt privacy.
  (iii) Maintaining a centralized database (like servers in Hospital) is fairly complex due to the large data variety.

- **Embedding DBMS in card:**
  (i) Provides availability (Card can be queried by any terminal without a software suite need)
  (ii) Provides privacy (Data is queried in the card, so private data is not given outside)

3.2. Data Stream Management Systems

Before 2000s real time data problems are handled in some studies, but after 2000 still today, data streaming is handled in different perspectives by many studies. Such data streaming studies are explained in the section.

The most important paper is certainly "STREAM" project of Stanford University [15]. In this project, a general purpose Data Stream Management System (DSMS) is build. STREAM can handle variable data rates of stream with limited system resources.

Another important stream project is Aurora[16] handles the streams for monitoring applications. Monitoring applications differs from the conventional data processing operations. So, Aurora cover this large network of triggers. Aurora processes and reacts to continuous data from many sources like sensor data.

In paper [17], new requirements and issues on data streaming are expressed and

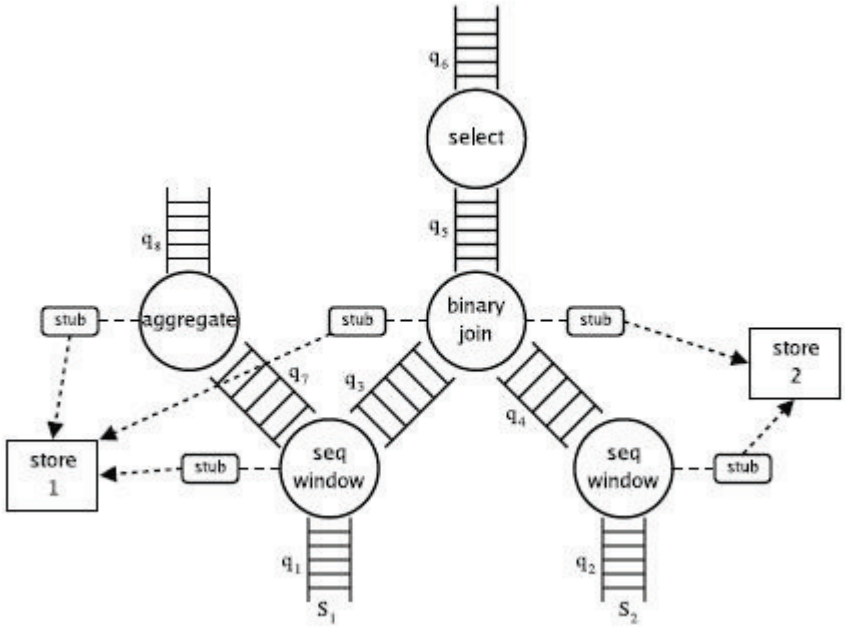

Figure 3.4. Query execution plan of STREAM

models are offered for these subjects.

MedsMan project [18] deals with multimedia streaming. They have proposed a stream management system for the media streams which are queried using digital processing techniques.

Tribeca project [19] is designed to analyze network traffic using stream-oriented DBMS.

3.2.1. Data Storage Models

In Stream project materializes intermediate results in query execution plan. This makes the RAM usage of STREAM project[15] bigger. Query execution plan of STREAM can be shown in Figure 3.4.

Aurora Project [16] also uses operator boxes to show its query execution plan and queues in front of these operator boxes as shown in Figure 3.5. Aurora project queue model can be seen in Figure 3.5. Its queues are designed to drop tuples when

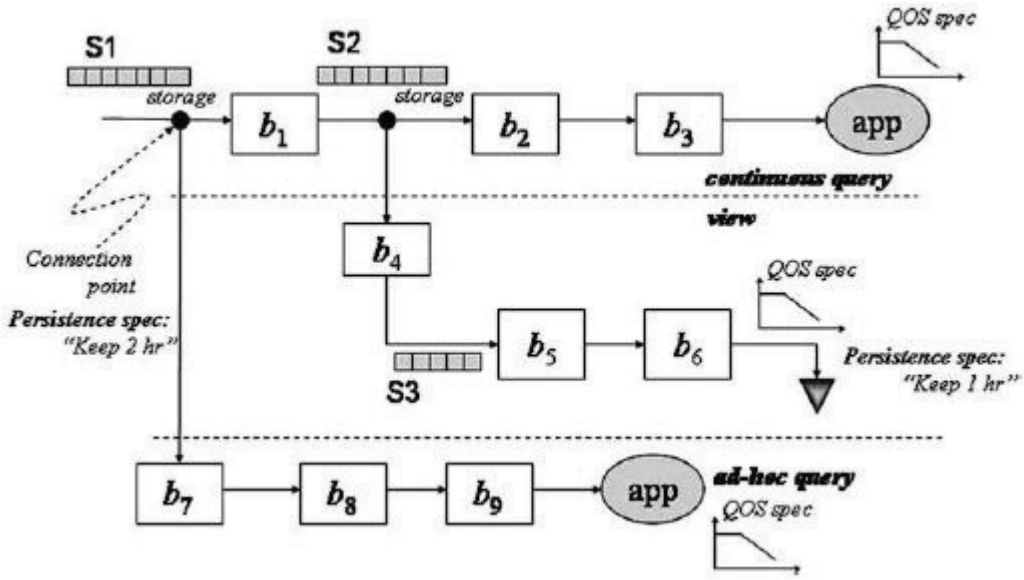

Figure 3.5. Query Execution Plan of Aurora Project

necessary. Dropping method looks the timestamps of tuples.

As explained in paper [17], Sliding Windows principle is an important model in stream data processing. Thanks to this method, optimizations can be done by taking some of the tuples according to their timestamps. The queue model in Figure 3.6 can be considered as a sliding windows data structure.

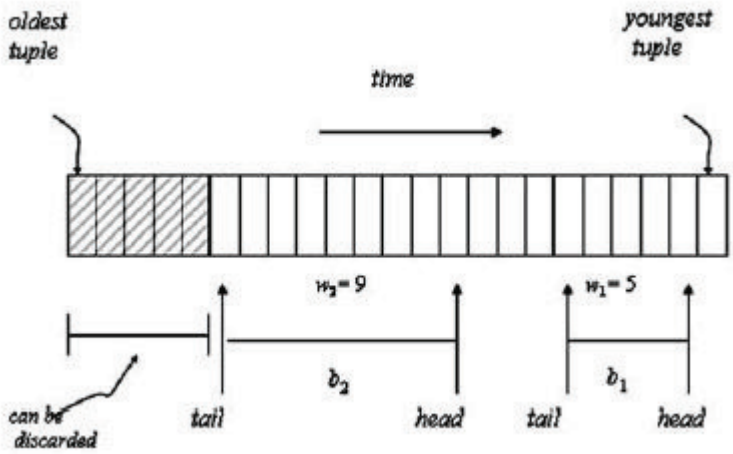

Figure 3.6. Aurora project queue model

3.2.2. Query Plans

As shown in Figure 3.5 and Figure 3.4, query plans are scheduled in tree-like structures with tuple queues and operator boxes. These structures use huge computing spaces.

3.2.3. Optimizations

Main optimizing used in these projects is "Tuple Dropping Method". Tuple Dropping can be done by using Sliding Window principle. Dropping operation can be done by looking tuples' timestamps.

3.2.4. Applications

MedsMan project is designed to be used in media streaming applications. The media streams can be queried in order to get some information about the media by using advance image processing techniques.

Aurora project [16] is designed for monitoring applications, for example data stream coming from sensors. Other examples of Aurora project's applications can be military applications that monitor readings from sensors worn by soldiers (e.g., blood pressure, heart rate, position), financial analysis applications that monitor streams of stock data reported from various stock exchanges, and tracking applications that monitor the locations of large numbers of objects for which they are responsible (e.g., audiovisual departments that must monitor the location of borrowed equipment).

3.3. Context Aware Systems

Many systems, especially in mobile applications can use context awareness to make application specialize for the users. Context-aware systems' advantage is their ability to adapt their operations to the current context without explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental

context into account.

The paper [20] holds the overall applications and researches about this issue. There are some systems using context awareness but main usage area is location-aware systems. Due to widespread usage of mobile devices, demand for location-aware systems is increased in recent years. For example, there are some tourist guide projects using location-awareness.

Other context aware system may detect more than one situation like location. For example, time and condition information can be combined with location information to do something independently the user's triggering. These context data can be gathered via some sensors, servers or software. For example, when a person stays somewhere at sometime, his mobile device can send message to some servers or other mobile devices using location and time information which can be gathered via GSM operator and combined with the personal data stored in his SIM card or phone.

4. SMART CARD DSMS BASED ON CONTEXT AWARENESS

Smart Card Data Stream Management System (SCDSMS) is a data stream management system which is designed for smart cards. It is also a context aware system that can recognize user's private information stored in the card.

A single stream source sends stream data to the smart card via using a kind of communication way of smart card according to the application. These communication devices can be contact based card reader or contactless based card reader. A contact based card reader may receive data via wired (e.g. Credit card machine with cable) or wireless medium (mobile phones with SIM cards). A contactless smart card can communicate with the terminal via contactless readers (e.g. Pay-Pass system of MasterCard) or other devices such as NFC device or NFC integrated mobile phones.

SCDSMS is loaded in EEPROM or ROM of a smart card (depends on card type: Native or Code Downloadable Cards) and runs using RAM and EEPROM of this card. SCDSMS also uses "Context" tables located in EEPROM. These context tables are EFs that contain relational tuples in TLV format and contains private information about user that are used by queries during query execution and not given outside. In the figure below 4.1 a overview of SCDSMS structure in Card can be seen clearly.

SCDSMS receives relational tuples and also saves results as relational data format into the EFs in smart card.

SCDSMS composed of these parts:

- Query Manager
- Context Manager
- Storage Manager

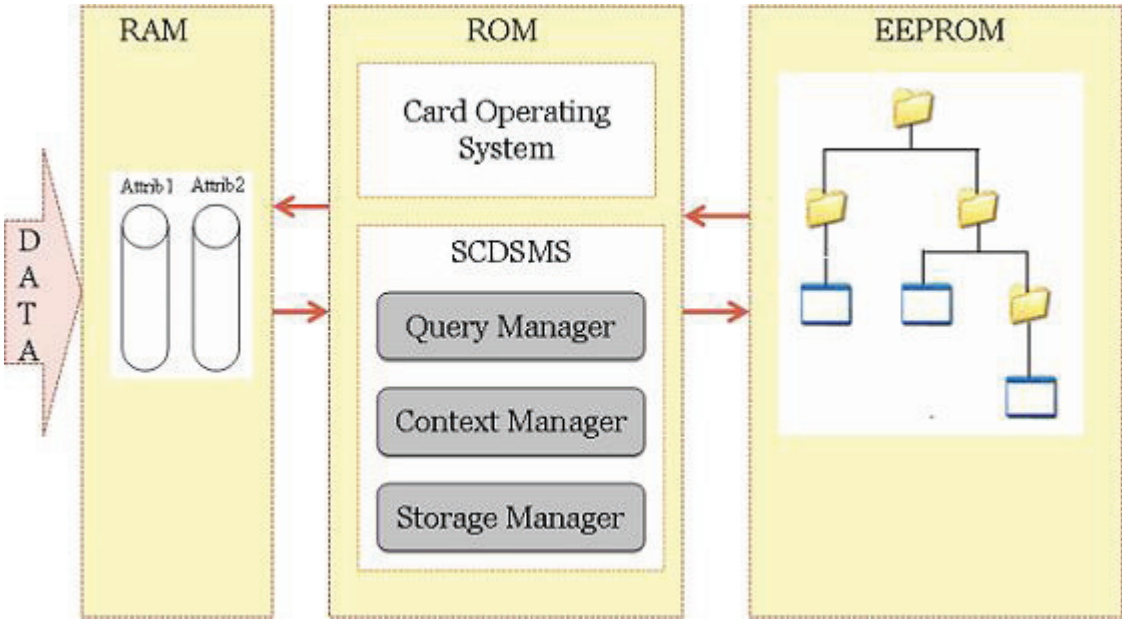

Figure 4.1. SCDSMS Structure in Card

Query Manager handles the query, parses it and then prepares the required data structures in RAM and EEPROM and organizes the query plan and query execution. Query Manager also communicates with the Context Manager in order to gather the context data from card's EEPROM. When query is executed (if there occurs a result) it sends the result to the Storage Manager to be stored in the result file in EEPROM of smart card.

Context Manager communicates with Query Manager and sends required data from smart card's EEPROM by applying a search algorithm to find it. Context Manager also use some tricks to find data easily in EEPROM.

Storage Manager is responsible to write results into the smart card according to the SCDSMS format. It also sends the result table to outside of the card when requested.

Many traditional DBMS has also some other parts as Access Right Manager or Transaction Manager. We assume that the Card Operating System handles Access rights of folders and files, so this is enough for Access right management in a smart card. We do not want to pump the application code size and its RAM requirements

due to the restrictions of EEPROM size in smart card.

For example, in our experimental environment, AKiS (a native kind COS), has PIN/PUK protection for every DFs. So, when reading or writing a EF under a PIN protected DF it is required to enter the PIN of related DF at the beginning of the streaming.

4.1. System Overview

As described above, SCDSMS is a data stream management system which is designed for smart cards based on context awareness that provides to recognize user's private information in the card and this feature can be used by the queries.

The single stream data source may buffer the tuples until SCDSMS is ready for the next stream. In this case the timestamps put by the source as "sent time stamp" and "creation time stamp" may be important if there exists a WSINCE command in the query. WSINCE can eliminates the unwanted old stream by checking the difference of timestamps with the time given in query. If the stream or tuple is older than given upper limit minute value, then it will be discarded.

SCDSMS uses relational tuples. It works with single data stream. It is a simple query manager used to query data stream using the private data generally called as context data which are located in the smart card's EEPROM. SCDSMS commands can be located in native code of COS or maybe downloaded to the card (if the COS supports downloadable code). In this paper, SCDSMS is embedded into a native operating system called AKiS. It is described in experiment part.

SCDSMS uses related EFs in smart card to make system context aware. EFs, that contain private data about card owner, include data in relational tuple format.

In SCDSMS, query is submitted before streaming. This query is taken by the first command that starts SCDSMS. After that query is submitted into the card, a

single stream data flows into this smart card to be queried. Query Manager parses the query and prepares data structures in RAM and EEPROM according to the query plan and waits for data stream.

Query may require context information stored in card not come with stream. At this time Context Manager is called by Query Manager to prepare easy access data structures in EEPROM which helps to search attributes during query execution.

Below some descriptive schemas of SCDSMS working principle in a smart card are shown. According to the system SCDSMS can be embedded in to the native code of operating system or can be downloaded as an application into the card works over the operating system.

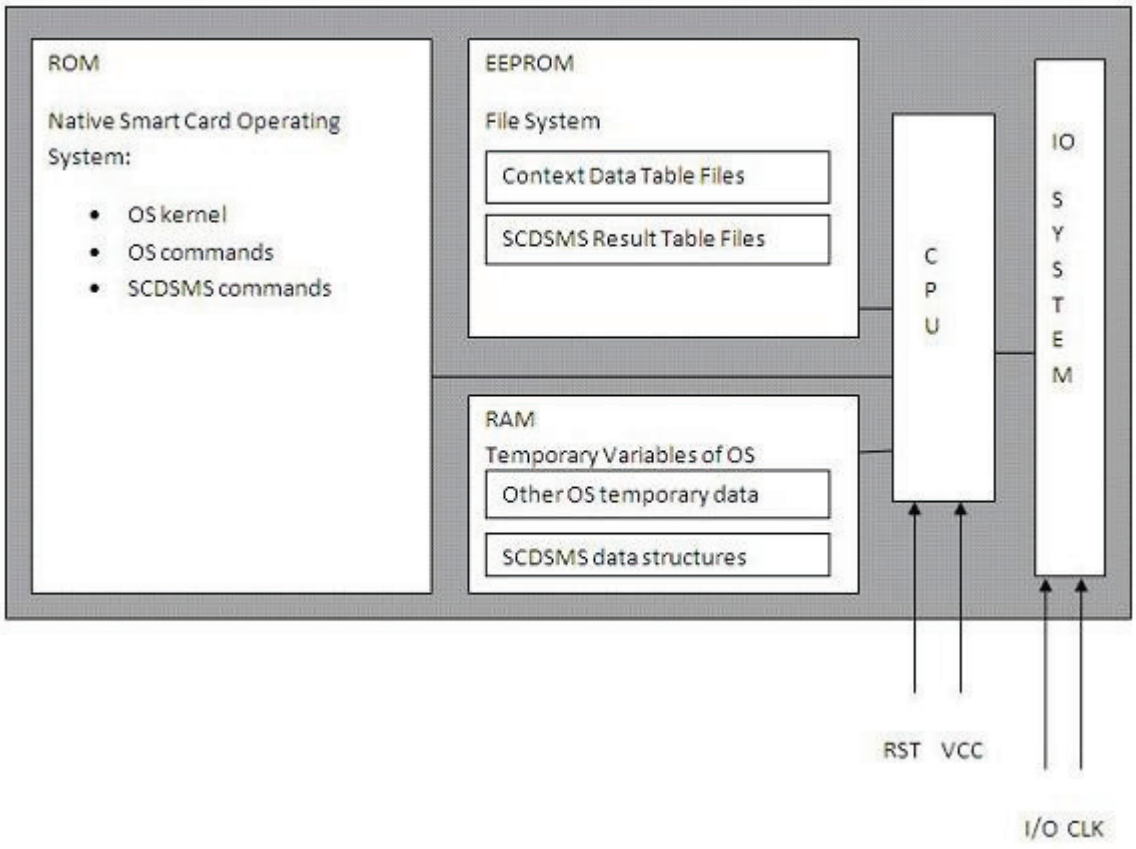

Figure 4.2. SCDSMS embedded in a Native Smart Card Operating System

In this document, SCDSMS is considered being embedded in the Native Card Operating System so all examples and scenarios are given through this way. Experiments are also done with a Native Card Operating System AKiS which is developed

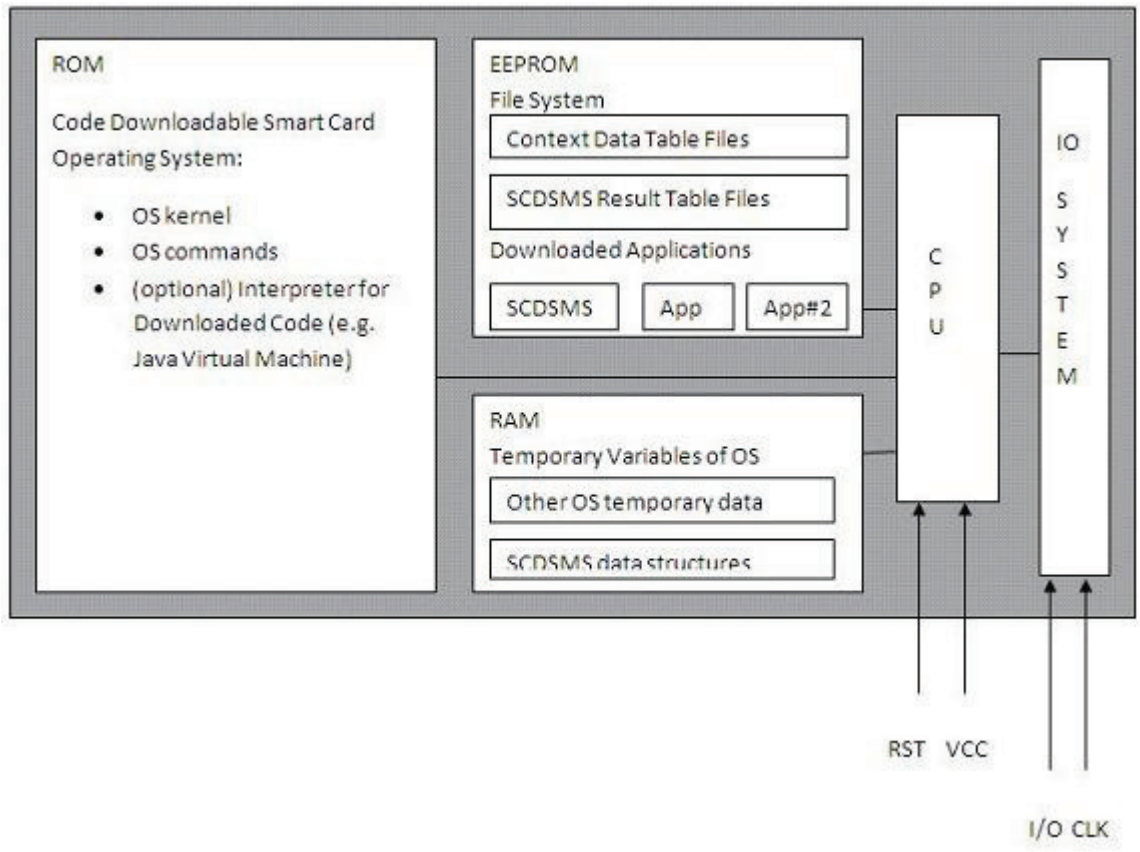

Figure 4.3. SCDSMS downloaded in a Code Downloadable Smart Card Operating System

by TUBITAK-UEKAE in Turkey.

Whenever a query result is obtained it is recorded into related EF in smart card as a relational tuple.

As shown in Figure 4.4, data streaming starts with special command sent to the smart card. When a data stream is started by a source, card receives the command which includes attribute list of data stream, result table name, and query. Smart Card operating system (COS) interprets the command and forwards this command to SCDSMS. By this way SCDSMS is started by COS and a session is created for this data streaming. SCDSMS takes the starting command data and builds a query plan, creates data structures into the RAM and EEPROM. Now SCDSMS is ready to take data stream and sends a response to source to start streaming. Source starts streaming by this response from card. Relational tuples are sent to card in command's data space. Every command can hold one stream which may include one or more tuples inside. As a result of streaming, a result EF is created in smart card. During query execution whenever a tuple which supplies the wanted criterias is found the required attributes of it are written into the result EF. When user want to get the results, he/she can call the read command of COS by giving the result EFs FID as parameter.In the next sections these processes are described in detail.

4.2. Problem Statement

There are some constraints to implement a DSMS for smart cards. These constraints can be listed as:

- RAM limitation (very small RAM compared to PCs, up to 10K)
- EEPROM limitation (very small stable storage compared to PCs, up to 160K)
- EEPROM write limitation (write operation in EEPROM is very costly in time)
- Power limitation (smart cards does not have a power supply inside)
- Command-Response working principle of smart cards
- Limited command size approximately 256 byte

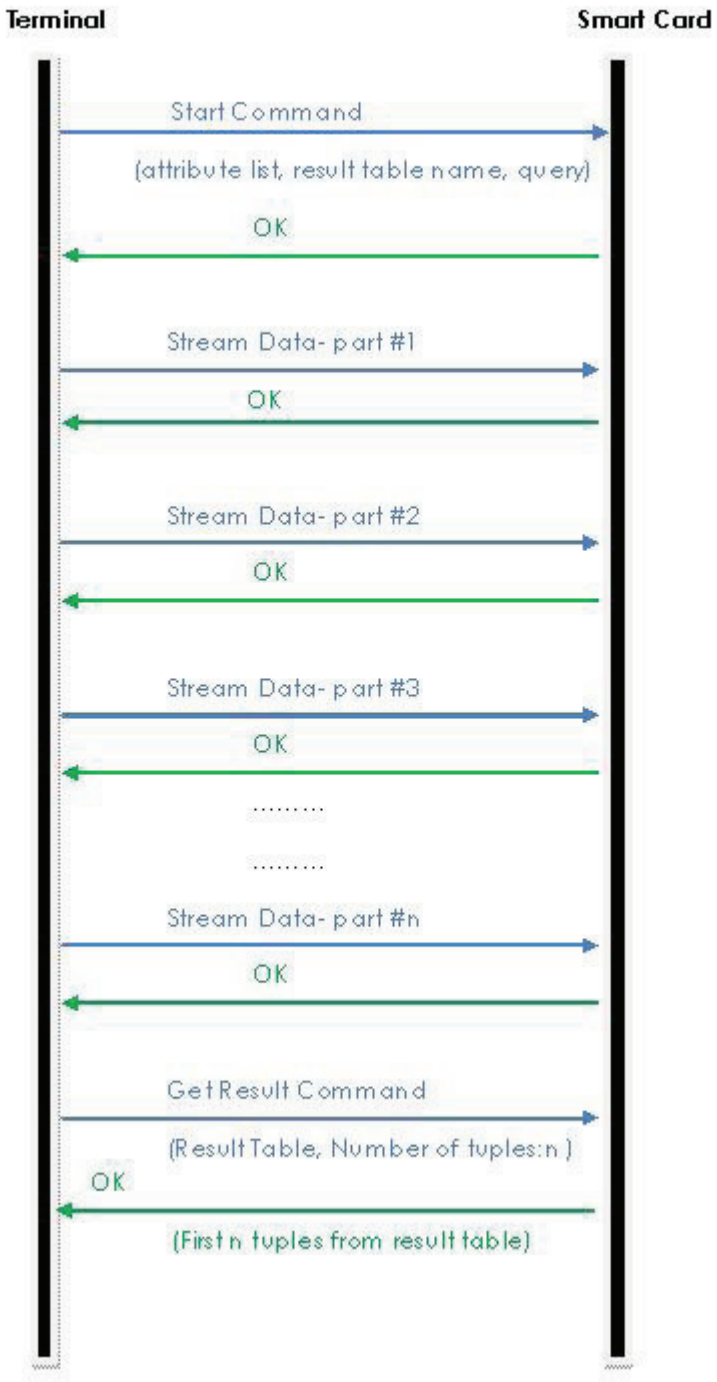

Figure 4.4. SCDSMS command-response flow

SCDSMS is designed according to these points. Data stream is taken into the RAM, RAM is small and useless data should be deleted and query should be executed immediately to take new stream into RAM. Results of query are recorded into the result file in EEPROM. EEPROM is also a limited area, so this point should be considered. EEPROM writing time is a little bit long, so it should be carefully considered to write in EEPROM.

Smart cards does not have its own power supply unit inside, so it can do processes needed while connected to a card reader which supplies power to the card via contact based or contactless technologies. Data stream management requires data stream buffering while executing query on previous stream tuples. Smart cards working principle, the Command-Response principle is a restriction to buffering new stream. Because, the new stream can only be sent after receiving response of previous command from the smart card. One command includes one stream and one stream may include more then one tuples. The restriction of command size is generally 256 byte and one stream can be up to 256 bytes.

SCDSMS is designed considering these restrictions. A conventional DBMS for smart cards can not handle data stream, due to the lack of the continuous query execution feature. A DSMS designed for PCs can not be implemented to smart cards due to the smart card limitations like RAM space. Consequently, SCDSMS is built to cover these issues with an additional feature called context awareness.

SCDSMS is embedded into the chip and provides minimal functionality of a stream data management system. It also uses context awareness issue which uses the private data of user stored in the smart card.

Some other features like view operation of conventional DBMSs, can be implemented on the terminal side because of the smart card's limitations. A PC-Suite of SCDSMS can be designed for terminal side to compensate the other feature that maybe useful for the users. The PC-Suite of SCDSMS is not mentioned in detail in this paper, this may be covered in an update report of SCDSMS in the future.

4.3. Query Manager

Query Manager handles query parsing, planning and execution operations. SCDSMS has its own simple query language which is specially designed for smart card restrictions. This query language, query planning methodology and detailed operations are described below.

4.3.1. Query Language

Query Language of SCDSMS is Smart Card Stream Query Language (SCSQL). Commands of SCSQL are:

- SELECT
- WHERE
- WSINCE
- WLAST

The brief of functionalities of these commands can be listed as :

- SELECT command selects the given attributes from the stream tuples.
- WHERE command applies elimination over stream due to the operations given.
- WSINCE command eliminates the tuples or streams due to their sent and creation time stamp, that are put in the stream source by comparing the minute value given.
- WLAST command selects the first n tuple in stream and discards others.

SELECT command corresponds to the project operation of relational algebra of database concept. WHERE command corresponds to the select operation of relational algebra of database concept. WSINCE command corresponds to a special kind of select operation of relational algebra of database concept. WLAST command corresponds to a special kind of select operation of relational algebra of database concept. So as in the relational algebra concept, SCDSMS has two kind of operations, these are Select

and Project. Also there are some operators used in WHERE clause:

- =
- !=
- <
- >
- AND
- OR

These operators are used with WHERE clause. Their functionalities are listed here:

- = Equality operator checks the equality of numerical or string values.
- != "Non-equality" operator checks whether the numerical or string values are not same.
- < "Smaller Than" operator checks whether the first numerical value is smaller than the second.
- > "Bigger Than" operator checks whether the first numerical value is bigger than the second.
- AND Logical "AND" operator checks whether the conditions are satisfied together in the statement.
- OR Logical "OR" operator checks whether one of the conditions is satisfied together in the statement.

Some example queries of SCDSMS are:

- SELECT CONFERENCEID
  WHERE (LENGTH<2hr) AND
  (CONFERENCEID ! = ATTENDEDCONFERENCES.ID)
  WSINCE (60sec)
- SELECT HEARTBEATRATE
  WHERE (BLOODPRESSURE > HEALTHINFO.AVGBLOODPRESSURE)
  AND (HEARTBEATRATE > 100)

4.3.2. Query Plan

In SCDSMS, there are 4 commands. There can be 3 commands in a statement in maximum. A statement in the maximum length (3) may be in "SELECT, WHERE, WSINCE" or "SELECT, WHERE, WLAST" format. Other statements may include "SELECT, WHERE" or only "SELECT" command. Thanks to this feature SCDSMS uses very small space in EEPROM while querying the stream data. So, rather than the query planning algorithm, data structures used in this algorithm is more important in our study. Here are the steps of query planner:

- Parsing Step: Parse the query according to the commands
- Data Structures Step: According to the parsing results, a group of data structures is prepared in EEPROM. These data structures are used to execute query easily and use minimum RAM and EEPROM while executing.
- PROJECT-1: Consider the attributes in query that are taken from data stream. If there are more attributes in stream than the required number of attributes, then do not create useless queues in RAM and drop down these attributes from incoming data stream using PROJECT operation.
- SELECT-1: (Windowing operations, WSINCE and WLAST, are also a kind of SELECT operation)If there exists one of the WSINCE or WLAST commands in the query, then apply them to drop only the required part of stream immediately to clean the EEPROM from the useless data.
- SELECT-2: If there exists a WHERE command, then search whether there exists any data required to gather from EFs in smart card (Context Awareness Property). If there exist one or more context attribute required in query, inform the Context Manager about required data to make Context Manager ready to give context data immediately.
- PROJECT-2: If result table contains less number of attributes than the number of attributes in current data stream table, then apply PROJECT operation

In these steps, operations used are Relational Algebra operations. So, it can be said that SCDSMS uses the SELECT/PROJECT part of the Relational Algebra. JOIN, NATURAL JOIN operations of relational algebra are not required in SCDSMS because there is only one table information in a data stream. So, single stream applications do not need join or aggregation operations. This makes SCDSMS works better in tiny RAM of smart card without pumping the used space.

Query Manager takes the incoming attributes from data stream into related queues in RAM. All attributes in data stream are hold in different queues. This helps SCDSMS to eliminate needless columns easily.

4.3.3. Operations

SCDSMS performs its operations in RAM and EEPROM. So, it is important to use tiny smart card RAM efficiently. In order to remember the smart card constraints again we can summarize the points to be taken into consideration while producing a smart card application. These points are:

- RAM limitation (very small RAM compared to PCs, up to 10K)
- EEPROM limitation (very small stable storage compared to PCs, up to 160K)
- EEPROM write limitation (write operation in EEPROM is very costly in time)
- Power limitation (smart cards does not have a power supply inside)
- Command-Response working principle of smart cards
- Limited command size approximately 256 byte

SCDSMS operations are designed by taking these points into consideration. Query Manager Operations are optimized to use very smaller RAM area than the Query Planners in other Data Stream Management Systems for PC-like devices. Other points are related with the Context Manager and Storage Manager of SCDSMS. Example query to be executed:

- SELECT CONFERENCEID WHERE (LENGTH<2hr) AND (CONFERENCEID

!= ATTENDEDCONFERENCES.ID) WSINCE (60sec)

Query Manager's first job is parsing the query according to the commands of SCSQL. Query Manager puts the parts of split query into a length 4 which is Command Queue. The Command Queue includes the command list of SCDSMS in execution order. At first all elements of this queue is equals NULL. When a command is seen in query then a link is created from the correspondent command bucket in the Command Queue to a new bucket in the Parameters Queue. Parameters Queue consists of the link of the parameters that the commands have in the query. The buckets in the Parameters Queue can be linked to these structures:

- Binary Operator array with size 3
- Required Attributes List from Incoming Stream
- Constant List
- Context List

When the parameters of command include Binary Operators of SCDSMS then a Binary Operator Array is created and it can be nested if there is more than one binary operator in the parameter. Required Attributes List is created to see which columns of incoming table are needed. It is also used to link the attributes in query and so repetition of attribute names has been avoided. Constant List consists of constant values used in the query. These values may be string or numerical values. Context List consists of links to the Context Manager's result list in EEPROM.

So, parsing is done as shown in the Figure 4.5. According to the parsing results, a query structure is prepared in EEPROM. By this way, only the incoming tuples are taken into the RAM, structures in Figure 4.5 are located in EEPROM.

In Figure 4.6 Query Execution Structures are shown which are located in RAM and used to carry tuples in a stream.

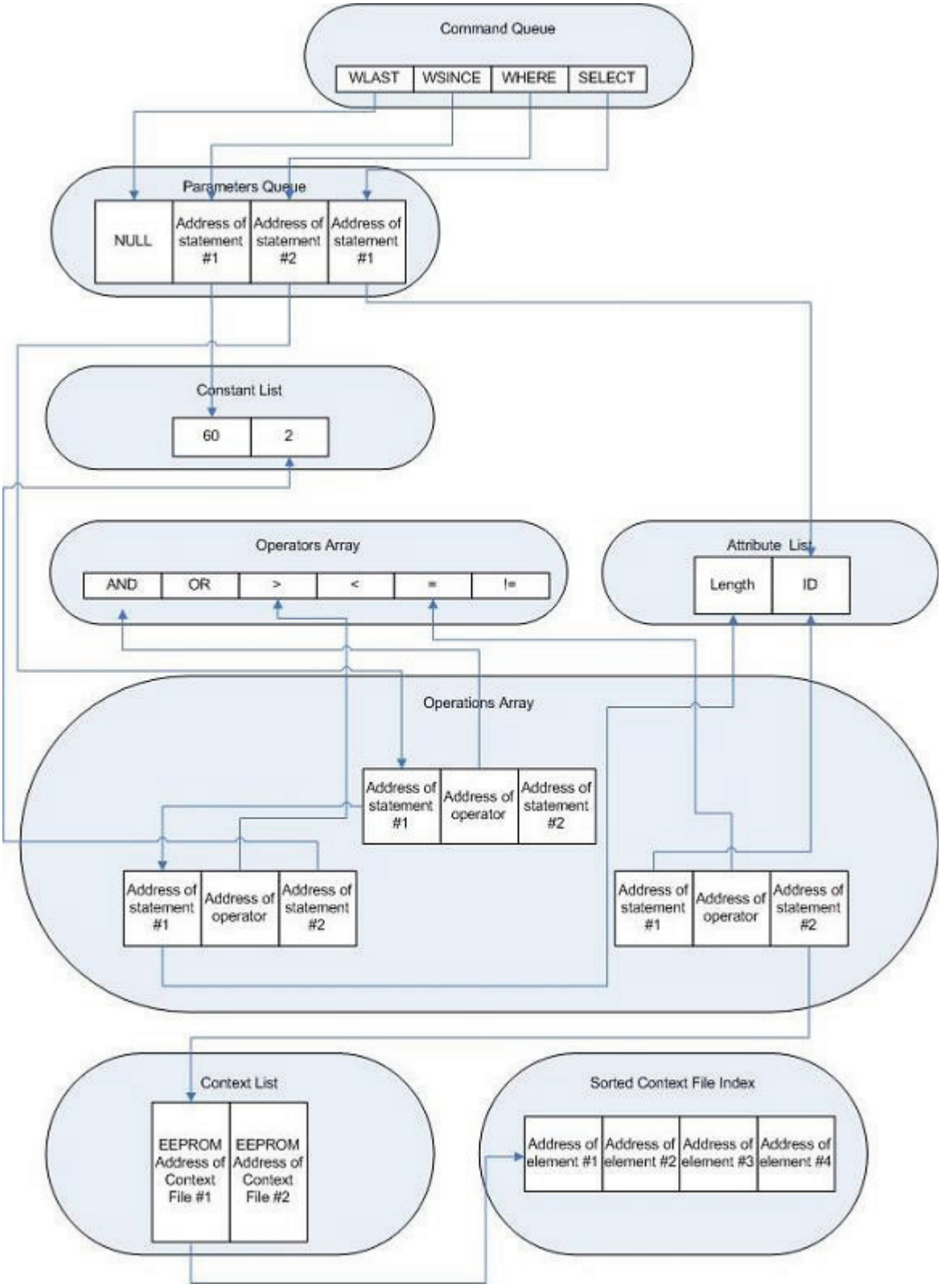

Figure 4.5. Query Manager Data Structures in EEPROM

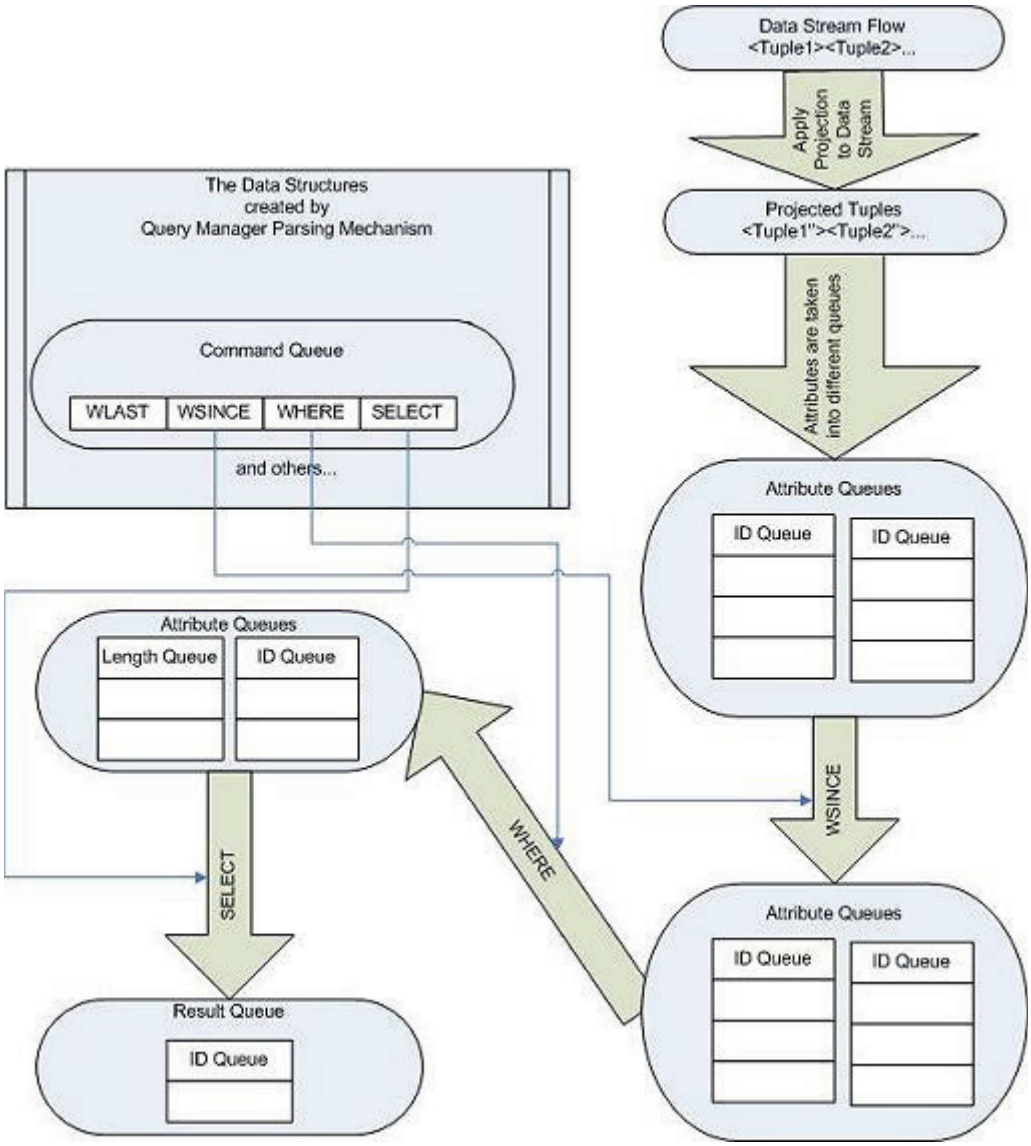

Figure 4.6. Query Manager Execution Structures

After creating data structures, it is time to apply step-3 in query plan which is dropping useless attributes. When a relational tuple comes to SCDSMS, it may be dropped according to the needless attributes in it. After query parsing it can be seen whether there are needless attributes. Then these attributes are dropped from the tuple before sending it to the data structures of first command in the Command Queue.

In our example the first command is WSINCE, so the modified relational tuple is reshaped for he data structures of WSINCE. After WSINCE operation is done, results are sent to the next command's data structure. By this way, after last command's operation is done, a group of data structures consist of results is obtained.

The sequence of executing the commands are always same. But if there are NULL values in Commands Queue, these NULL valued commands are not executed. This is because these commands are not called in the query and so has no parameter in the Parameters Queue.

As we can see, Query Manager optimizes RAM and EEPROM usage by its features. These features can be summarized as below:

- PROJECT operation applied in step-3
- Applying Windowing operations
- Using different queues to store attributes

By first feature of SCDSMS Query Manager, some RAM space is gained by eliminating useless attributes of input stream. Besides, by second feature, needless tuples are eliminated at the begging of the operations. If they are eliminated at the last step of query execution, they would be carried in the data structures and executed by the commands in the Commands Queue. This would cause redundant consumption in space and time.

Also using different data structures for different attribute groups provide to eliminate non required ones easily. This saves time and as a result the space.

4.3.4. Data Structures

4.3.4.1. Command Queue. Command list of SCDSMS is a length 5 array which holds pointers to Parameters Queue. If there does not exist the related command in the query, then its bucket contains NULL value. In our example, WLAST command's bucket includes a NULL value whereas the other elements contain pointers. This table puts the command execution in order and does not cause big space consumption, it only holds address values if there exists the related command in the query, and else it puts just a null value like a hexadecimal "FF" value in one byte.

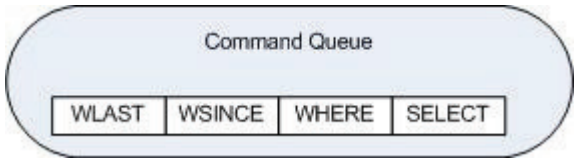

Figure 4.7. Command Queue

4.3.4.2. Parameters Queues. Parameters Queue includes addresses that point to the parameter structures of the related command in the Command Queue. Holding addresses makes space consumption lower and using pointer structures makes query plan obvious and easy to be executed.

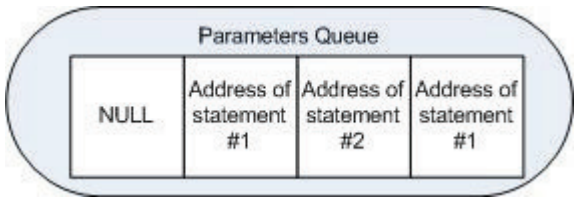

Figure 4.8. Parameters Queue

4.3.4.3. Attribute Queues. Attributes of incoming data stream are stored in the QUEUEs for each attribute in RAM. These queues are first in first out (FIFO) queues. For each command in the query there exist queues to hold the attributes for this command ex-

ecution. This data structure provides an easy project operation during the execution. If an attribute queue is not necessary for the rest of the execution, just this attribute queue is deleted.

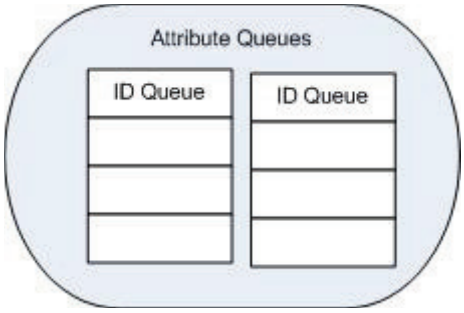

Figure 4.9. Attribute Queue

4.3.4.4. Attributes List. In order to prevent data redundancy in EEPROM, Attributes List holds the attribute names used in the query.

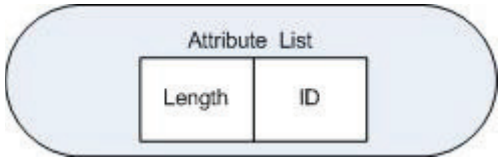

Figure 4.10. Attributes List

4.3.4.5. Constant List. In order to prevent data redundancy in EEPROM, Constant List holds the constants used in the query.

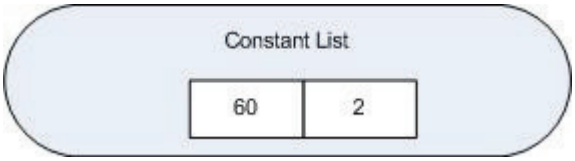

Figure 4.11. Constant List

4.3.4.6. Operators Array. In order to prevent data redundancy in EEPROM, Operators Array holds the operator list of SCDSMS.

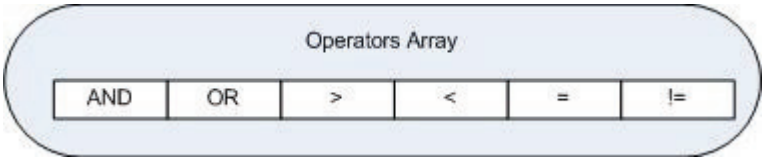

Figure 4.12. Operators Array

4.3.4.7. Context List. Context List holds addresses that point the context data in EEPROM. These addresses are put into this structure by Context Manager. Context List elements may be used by Parameters Queue or Operation Array. Holding addresses makes space consumption lower and using pointer structures makes query plan obvious and easy to be executed.

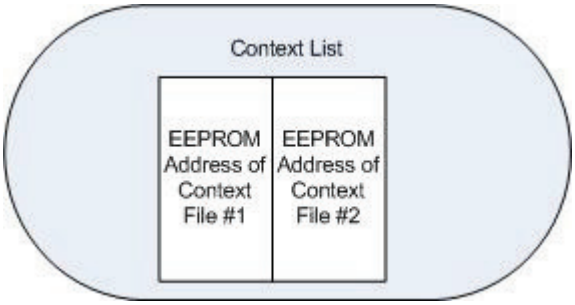

Figure 4.13. Context List

4.3.4.8. Operation Array. Operation Array holds the addresses of the operator and the two values that will be used by that operator. Always second element holds the operator pointer. Element orders are important, because the first element is the first parameter for the operator which is placed in second order. Last element is the second parameter of the operator. Holding addresses makes space consumption lower and using pointer structures makes query plan obvious and easy to be executed.

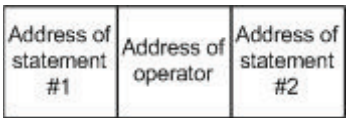

Figure 4.14. Operation Array

4.4. Storage Manager

SCDSMS has a simple storage manager which provides to store results into the smart card's EEPROM in desired format. After taking the results from Query Manager, Storage Manager converts these results into relational tuples and stores them into related EFs.

4.4.1. Operations

Storage Manager has a function named "Save" which destination takes EF name and list of result data structure's pointers in the correct order to save in EF as a tuple. The EFs are stored under the SCDSMS application's directory. By this way, it can store the data by its own format and optimization indexations and methods.

4.4.2. Data Structures

Storage Manager takes the result attributes from result data structures. It does not need another storage structure for them. The write function of Storage Manager places the result attributes as in relational tuples in EFs. In EFs, these tuples are saved in TLV format. Thanks to saving the results in TLV format, avoids space consumption in EEPROM. If we store the result as records, we should define fixed spaces for records; as a result there would be so many non-used areas in EFs.

The values shown in the EFs are in hexadecimal form. As seen in Table 4.1, there are so many wastes space in EF #1 whereas the EF#2 uses the space efficiently. Reading operation of EF#1 may result a little faster than reading the EF#2, but this difference does not worth so much space consumption. Besides, reading speed of EEPROM is very good than writing on it.

Every EF has an header part which includes the information about how many tuples and attributes it has. This header part in TLV type EF is seen as in Table **??**.

Table 4.1. EF#1 - Record type result, EF#2 - TLV type result

<2><2> <127><unused space for Attribute1> <13245>< unused space for Attribute2> <12564><unused space for Attribute1> <9845>< unused space for Attribute2>
<90><6><80><1><2><70> <1><2><60><2><F1,F2> <F1><3><127><F2><5><13245> <F1><4><1256><F2><4><9845>

Table 4.2. TLV type EF Header

<90><6><80><1><2><70><1><2><60><2><F1,F2>

The meaning of bytes are respectively given below:

- 90: Header TAG
- 6: Total length of data in header information (in byte)
- 80: (T) TAG which means "Number of Attributes"
- 1: (L) number of attributes
- 2: (V) number of attributes in the EF
- 70: TAG which means "Number of Tuples" in the EF
- 1: length number of tuples (in byte)
- 2: number of tuples in the EF
- 60: (T) TAG which means "TAGs of Attributes in order" in the EF
- 2: (L) number of tuples (in byte)
- F1,F2: (V) TAG of attribute #1 and #2

4.5. Context Manager

Context Manager provides a context awareness feature to the SCDSMS. User's private information (like previous health measurements) are stored in the card's EEP-

ROM as context tables. When a query includes a statement that uses the context data of user, Context Manager is called by Query Manager to gather the information.

For example, if the query includes such a statement: "'Record the bloodpressure if user's new heartbeat rate is greater than his previous hearthbeatrates"', the previous hearthbeat rate information is queried using Context Manager among the context tables of the user hold in EEPROM. By this way, user's private information is used by queries without giving them outside explicitly or holding them in an unsecure outside storage(like hospital's servers). Using Context Tables supplies more accurate results by using personal information.

Query Manager calls Context Manager to get required context data from card inside. Context Manager reads the data which are probably stored before by Storage Manager. Context Manager creates a pointer index in EEPROM in order to reach the context data in context files in EEPROM easily.

4.5.1. Operations

Context Manager is used only by the Query Manager before the execution and during the execution. Before taking stream data, Query Manager takes the query into the RAM and prepares its data structures and query plan. If query has a comparison statement which needs context data, then Query Manager calls the Context Manager to prepare index structures.

When Context Manager is called by Query Manager, it searches the related attribute in given table EF. Context Manager puts a pointer index structure into the EEPROM which helps it to find and search data when it is needed by Query Manager. This index structure is sorted so searching on it is very easier than searching on context file itself which also includes other attributes inside.

4.5.2. Data Structures

Context Manager uses the data in EEPROM which is stored by the Storage Manager. Context Manager creates an easy access index in EEPROM which includes addresses of related attribute in all tuples of the table. By this way, finding related attribute becomes very easy. Another improvement to find the value in attributes of table is sorting the table according to this attribute. Applying a sort algorithm on this attribute in table and then rewriting table in EEPROM is very hard to achieve due to the writing restrictions of EEPROM in smart cards. It should not be rewritten in EEPROM. So, Context Manager solves this problem by creating a semi-sorted address table in RAM. Context Manager creates a sorted index list in order to make its search easier.

Table 4.3. Conference Query Example

SELECT CONFERENCEID WHERE (LENGTH<2hr) AND (CONFERENCEID != ATTENDEDCONFERENCES.ID) WSINCE (60sec)

Here ATTENDEDCONFERENCES.ID is an element of context data. So, Context Manager puts a pointer into the Context List, the value of these data or a linkage to its sorted index list shown is 4.5.2. There is a pointer of ATTENDEDCONFERENCES.ID in Context List. This pointer links to the address list of ID attributes in AttendedConferences table in EEPROM. In AttendedConferences.ID table includes a sorted list which includes addresses of attributes.

4.6. Algorithms and Methods of SCDSMS

SCDSMS contains 3 parts:

(i) Query Manager

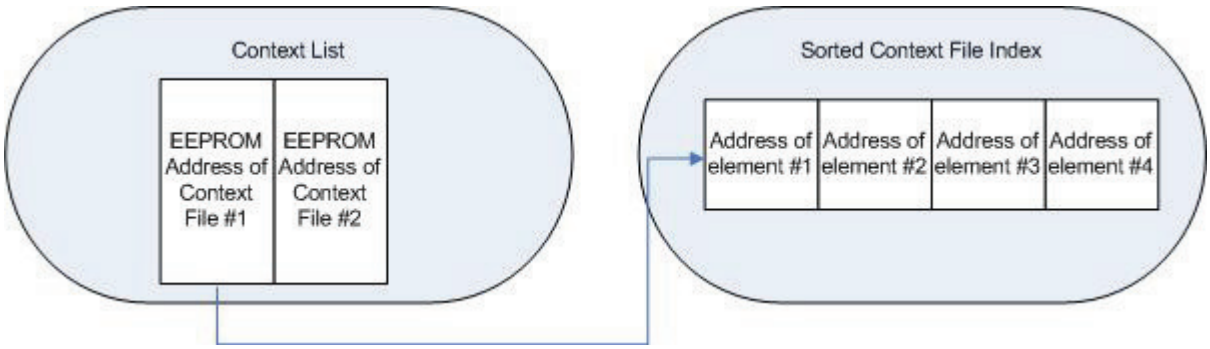

Figure 4.15. Sorted Index File

(ii) Storage Manager

(iii) Context Manager

Query Manager uses Parsing and Planning algorithms. Storage Manager uses Storing algorithm. Context Manager uses Sorting and Searching algorithms. In this part, methods of the 3 parts will be given. Also in these methods, mentioned algorithms are used.

4.6.1. Query Manager's Method

Here is the step by step methodology that Query Manager applies:

- Query Manager takes query from COMMUNICATION MANAGER and applies PARSING ALGORITHM on it.
- QM applies PLANNING ALGORITHM using the data gathered by 1st step. According to the query, CONTEXT MANAGER may be called in this step. By the 1st and 2nd steps, QM data structures have been prepared in EEPROM and RAM. Only data stream holder structures are created in RAM, other easy access or command holder structures are created in EEPROM.
- Now it starts to get stream data from single stream data source. Puts the data into the EXECUTION ALGORITHM. If required results can be taken from tuple, then the result is stored in smart card by STORAGE MANAGER.
- Result file can be read after streaming.

4.6.1.1. QM Parsing Algorithm. Here is a step by step brief of Parsing Algorithm:

- Take query into initialQueryString.
- Search for the commands in the initialQueryString.
- Create Command Queue list in EEPROM and put NULL value if the related command does not exist in the query.
- Create Parameters Queue according to the number of commands in the query.
- Put links in Command Queue to the Parameters Queue.
- Put related parameters in tempParameters list as String.
- Parse tempParameters according to the OPERATORS of SCDSMS. Use Operators Array to point which operator is used in statements. These statements are hold in Operations Array.
- Attributes used in query are hold in Attribute List and they are linked as shown in Figure 4.5.
- Constants used in query are hold in Constant List and they are linked as shown in Figure 4.5.
- Context data related attributes in query are taken from Context Manager and Context List holds the links to the easy access tables for context data in RAM. These index tables that are used for easy access are created by Context Manager in the RAM.

4.6.1.2. QM Planning Algorithm. Here is a step by step brief of Planning Algorithm:

- During query execution,QM takes tuples from data stream queue and apply PROJECT operation if necessary. PROJECT operation is necessary when required attributes by query is less than the data stream has. In this case,query planner prepares a needless attributes list that are going to be dropped from the incoming tuple and putted into the next list called "Projected Tuples" during execution.
- During query execution, from "Projected Tuples" list, attributes are selected and put into different queues called "Attribute Queues -1", so query planner also

prepares the "Attribute Queues -1" before execution.

- During execution, if there exists a WSINCE or WLAST command in query, this WSINCE or WLAST command is executed with the data of "Attribute Queues-1". New attributes are put into the next "Attribute Queues-2" structures, so query planner also prepares the "Attribute Queues -2" before execution.
- During execution, , if there exists a WHERE command in query then the attributes in "Attribute Queues-2" are taken one by one and WHERE command is executed by the output to "Attribute Queues-3", so query planner also prepares the "Attribute Queues -2" before execution. If Context Data are required in query then Context Manager is also informed about that to prepare the sorted index file in EEPROM and put the address of that file in RAM. If more than one index file is required according to the attribute required in query, then in RAM context address holder includes more then one index file addresses.
- During execution, if there exists a SELECT command in query then the attributes in "Attribute Queues-3" are eliminated as to "Attribute Queues-4" which included less attributes than before the SELECT command execution, so query planner also prepares the "Attribute Queues -4" before execution.

4.6.1.3. QM Execution Algorithm. Here is a step by step brief of Execution Algorithm:

- All steps in query planning algorithm are traced using the prepared data structure in EEPROM and RAM.

4.6.2. Storage Manager's Method

Storage Manager stores when there occurs a result after a tuple execution.

- Takes result from QUERY MANAGER.
- Execute STORING ALGORITHM on the data.

4.6.3. Storage Manager's Method

Storage Manager stores when there occurs a result after a tuple execution.

- Takes result from QUERY MANAGER.
- Execute STORING ALGORITHM on the data.

4.6.3.1. SM Storing Algorithm.

- Takes the input from STORAGE MANAGER in the format: EF NAME, LIST OF RESULT STRUCTURES' POINTERS
- Searches the EF in the SCDSMS directory. If the EF exists then STORING ALGORITHM appends data to it, else it creates a new EF and puts a header at first.

  Header is in this format like in the example above:

  `<90><6><80><1><2><70><1><2><60><2><F1,F2>`

  (i) 90 : Header TAG
  (ii) 6 : total length of data in header information (in byte)
  (iii) 80: (T) TAG which means "Number of Attributes"
  (iv) 1 : (L) number of attributes
  (v) 2 : (V) number of attributes in the EF
  (vi) 70 : TAG which means "Number of Tuples" in the EF
  (vii) 1: : length number of tuples (in byte)
  (viii) 2 : number of tuples in the EF
  (ix) 60 : (T) TAG which means "TAGs of Tuples in order" in the EF
  (x) 2 : (L) number of tuples (in byte)
  (xi) F1 , F2: (V) TAG of attribute #1 & #2
- If EF exists, before appending the tuple, the attribute number in the header of file is incremented. Takes result from QUERY MANAGER.
- According to the given order in the list of structures' pointers, attributes are read from result data structures in the RAM and written in the EF as shown in the

example :

<F1><3><127><F2><5><13245>

<F1><4><1256><F2><4><9845>

Here F1 and F2 values are the TAGs of attributes.

4.6.4. Context Manager's Method

Context Manager finds and retrieves the context data from SCDSMS directory in EEPROM.

- Take input from Query Manager. Input is the first element in the Context List. Here "AttendedConferences" is the EF name and "ID" is the required attribute name.
- Find context EF in the SCDSMS directory using COS functions.
- Create SORTED CONTEXT DATA LIST in the EEPROM with the step 2's result EF . SORTED CONTEXT DATA LIST holds the addresses of found attributes in EF in sorted order using CM Sorting Algorithm.
- Using EF pointer SORTING ALGORITHM is called. As a consequence of this execution SORTED CONTEXT DATA LIST is filled with the order numbers of attributes.
- When data streaming is started, Context Manager is subject to find required attribute by its SEARCHING ALGORITHM. SEARCHING ALGORITHM takes 2 input. First is the operation pointer and second is the attribute's pointer.
- After SEARCHING ALGORITHM is executed, a result a list of found element addresses is returned to the QUERY MANAGER.

4.6.4.1. CM Sorting Algorithm.

- Attributes of EF are semi-sorted according to their first 5 character. Semi-sorting is better than full-sorting respect to the time consumption.
- Used sorting algorithm is Bubble Sort due to its very small space consumption

than others. Space complexity of Bubble Sort is O(1).

- As the result of 2nd step, semi-sorted list is created.

4.6.4.2. CM Searching Algorithm.

- It takes operation array's pointer as input . Where operators list includes 6 operators as : AND, OR , <, >, != ,=
- If operator is 'AND', 'OR', '!=' or '=' the exact value as the attribute is searched in the EF. Else if the operator is '<' or '>' the bigger or smaller values are searched in the EF.
- Used searching algorithm is Divide and Conquer algorithm. According to the semi-sorted list, the value is searched with this algorithm. If the exact value is searched, then the found same valued element orders are hold in an array and returned as a result of Searching Algorithm. Else , the bigger or smaller values are searched in the semi-sorted list, by the Divide and Conquer Algorithm the value is searched by making smaller the searching space thanks to the algorithm. When divide and conquer algorithm completes its search space, and there is not exist the searched value in the list, then the if the search type is for finding the exact value then NULL is returned, else the search type is BIGGER THAN or SMALLER THAN types, then the last order is returned as result. According to the search type, higher or lower values are taken from the list.
- For all cases return value is an array contains addresses of found elements.
  For AND, OR,!= ,= type searches :
  (i) If array is NULL, it means no element contains the searched value.
  (ii) If array is not NULL, then it contains the found elements' addresses.
  For >,< type searches :
  (i) If array is NULL, it means no element found ">" or "<" than the searched value.
  (ii) If array is not NULL, then it contains the found element's address.

4.7. Optimizations

Due to the smart card restrictions as:

- RAM limitation (very small RAM compared to PCs, up to 10K)
- EEPROM limitation (very small stable storage compared to PCs, up to 160K)
- EEPROM write limitation (write operation in EEPROM is very costly in time)
- Power limitation (smart cards does not have a power supply inside)
- Command-Response working principle of smart cards
- Limited command size approximately 256 byte

Algorithms and methods should be carefully designed according to them. Briefly, we have 4 methods and 6 algorithms used in SCDSMS:

- Query Manager Method
- QM Parsing Algorithm
- QM Planning Algorithm
- QM Execution Algorithm
- Storage Manager Method
- SM Storing Algorithm
- Context Manager Method
- CM Sorting Algorithm
- CM Searching Algorithm

One by one, the optimizations in those algorithms and methods are revised in this part. Query Manager Method uses Parsing, Planning and Execution Algorithm. It shares the variables with other parts of SCDSMS so RAM space is effectively used. Query Manager optimizes RAM and EEPROM usage by its features. These features can be summarized as below:

- PROJECT operation applied in step-3
- Applying Windowing operations

- Using different queue-like structures to different store attributes

By first feature of SCDSMS Query Manager, some RAM space is gained by eliminating useless attributes of input stream. Besides, by second feature, needless tuples are eliminated at the begging of the operations. If they are eliminated at the last step of query execution, they would be carried in the data structures and executed by the commands in the Commands Queue. This would cause redundant consumption in space and time.

Also using different data structures for different attribute groups provide to eliminate non required ones easily. This saves time and as a result the space. Parsing Algorithm uses pointer lists instead of using value lists. This pointer lists saves space in RAM of the smart card. Planning Algorithm's projection operation is a serious optimization that drops the useless data in the RAM. It also uses different lists for all attributes that makes projection algorithm very trivial.

Execution Algorithm uses the data structures prepared by query planner. So, it executes efficiently. During execution only the data stream is taken into the RAM area, other address holder structures are hold in EEPROM because they are not modified during execution, they are only read. Holding these read-only files in EEPROM makes RAM area big enough to carry stream tuples.

Storage Manager's method is not more than calling Storing Algorithm. Storage Manager uses the space efficiently using TLV structure while writing result tuples to the result EF. Storing Algorithm uses TLV format to use EEPROM space efficiently. EFs have headers which contains:

- Number of Attributes
- Number of Tuples
- TAGs of Tuples in order

This information in header makes some works easier. For example if some method needs the tuple number in an EF, it can only read the header of that EF. Context Manager's method takes the input from the Context List. So, it does not recreate a data structure. Sorting Algorithm's output is a semi sorted list of order numbers in the related EF in EEPROM. Used sorting algorithm uses small RAM space than others. Searching Algorithm uses Divide and Conquer algorithm to make use of the semi sorted list's advantage.

4.8. SCDSMS-beta

SCDSMS-beta is implemented in AKiS which is a native Card Operating System developed by TUBITAK-UEKAE. SCDSMS-beta is a simple version of SCDSMS. SCDSMS-beta is implemented to show SCDSMS is an applicable system. Also, using SCDSMS-beta performance of this system is measured. SCDSMS-beta has all complex operations of SCDSMS inside. Some features of SCDSMS are not implemented but the vital and performance related features are implemented in SCDSMS-beta. 2 commands are implemented to make experiments, called "SubmitQuery" and "Stream" is explained in this section.

SCDSMS-beta holds context data files under MF. They can be selected via their SFI given in P1 parameter as one byte. SCDSMS-beta can use the operator "!=", "==", ">" and "<". Other operators of SCDSMS like "AND" and "OR" are not implemented but they are not related directly with the performance. SCDSMS-beta can use the query format as shown in Table 4.5.

Table 4.4. Query format of SCDSMS-beta

SELECT	TAG1	WHERE	TAG2	Operator	TAG3

SCDSMS-beta does not contain extra command to read Result EF. AKiS command "ReadBinary" can be used to get results from the EEPROM of the card after streaming is completed.

Table 4.5. tags and operators of SCDSMS-beta

TAG1	An attribute Tag in the tuples of the stream. These attributes are written into the result file if the Where clause is satisfied.
TAG2	An attribute Tag in the tuples of the stream. These attributes are compares with TAG3 attribute of the selected context file.
Operator	Operators can be : != 01H , == 02H , < 03H , > 04H ,
TAG3	An attribute Tag in the tuples of the context file. These attributes are compares with TAG2 attribute of the tuples in stream.

4.8.1. SubmitQuery

SubmitQuery should be sent before streaming. SubmitQuery command starts the SCDSMS-beta with sending Context File's SFI, attribute TAGs and Operator that are used by the query. Submit Query can execute one type of query that is shown above in the Table 4.6. Fields of the SubmitQuery is shown in the Table 4.7. In experiments, SubmitQuery command's examples are given.

P1 : Context Table's SFI

P2 : Result Table's SFI

DATA: Tags of attributes and the operator indicator are given. This field should include 4 bytes.

Table 4.6. SubmitQuery Command

CLA	INS	P1	P2	LC	DATA
00H	01H	00H	00H	04H	

Table 4.7. Data Fied of SubmitQuery Command

Byte1	Byte2	Byte3	Byte4
TAG1	TAG2	Operator	TAG3

4.8.2. Stream

Before first Stream command is sent SubmitQuery should be sent to prepare query plan. One Stream command include one stream. One stream may include one or more tuples. LC: Length of the stream data DATA: Includes the stream which contains one or more tuple inside in TLV form.

Table 4.8. Stream Command

CLA	INS	P1	P2	LC	DATA
00H	02H	00H	00H		

5. EXAMPLE APPLICATIONS

Here are 3 main applicable areas we have proposed. SCDSMS can be used in Health Cards , SIM Cards and smart cards used in Congress Centers or malls. In Figure 5.1 the overview of the general infrastructure of these applications that use SCDSMS efficiently.

Figure 5.1. Infrastructure for SCDSMS Applications

5.1. Health Cards

Up to now, health cards are used only to store emergency information and small sized measurements of patients. Other information like huge measurement history are hold in hospital's servers or only patient's folders as printed copies.

We offer a new system, in which health cards include more information about patients and include a data management system called SCDSMS which is used to record and query measurements. By this way, patient's private measurements are not hold a unsecure hospital server. On the other hand, at any terminal patient or doctor can access patient's measurements. Embedding SCDSMS to card also makes these operations terminal independent.

In Figure 5.2, SCDSMS Health Card Application Components are shown. In this system patient's measurements are recorded into the card during the measurement pe-

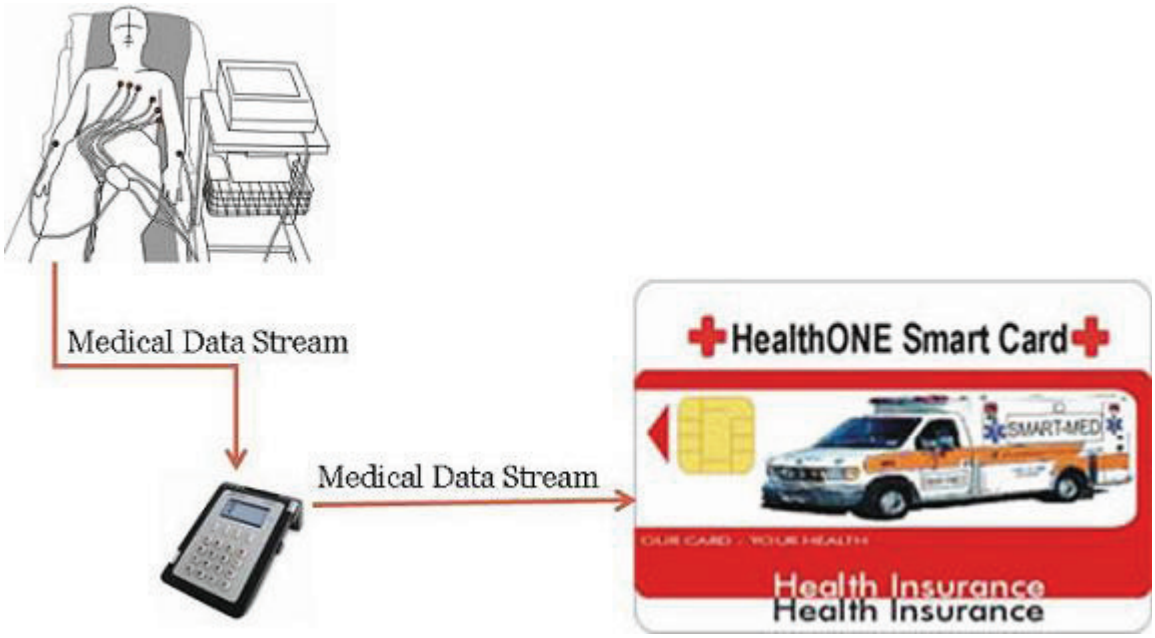

Figure 5.2. Health Card Application Components

riod by connecting the health card to the medical device. But here occurs a problem. The measurement data are too big to store in a smart card because they include some long-time measurements like EEG device outputs which include 15 min-long measurements or some devices may give one-day-long outputs. So, at this point SCDSMS handles the problem by recording only the important parts of the streaming data from the medical device connected to the patient and stores a very small part of the output into the health card. The important parts of the streaming data are decided by the query set by the doctor before medical device's streaming starts. After measurements are completed, the doctor can query the results of the patient's measurements. Carrying these measurements in the card makes observability on the patient's history easier and holds their information in a more secure way than today's health card applications.

As an example to the Health Card applications the query in Table 5.1 can be used. In this case, heart beat rate of patient is recorded if the condition in where clause is satisfied. In WHERE clause two comparisons are done. First one is the comparison of heart beat rate in stream tuple with static value 100. The other one is the comparison of average blood pressure in context file with blood pressure in stream tuple.

In Table 5.1, S1 is HEARTBEATRATE attribute TAG in incoming tuples. S2

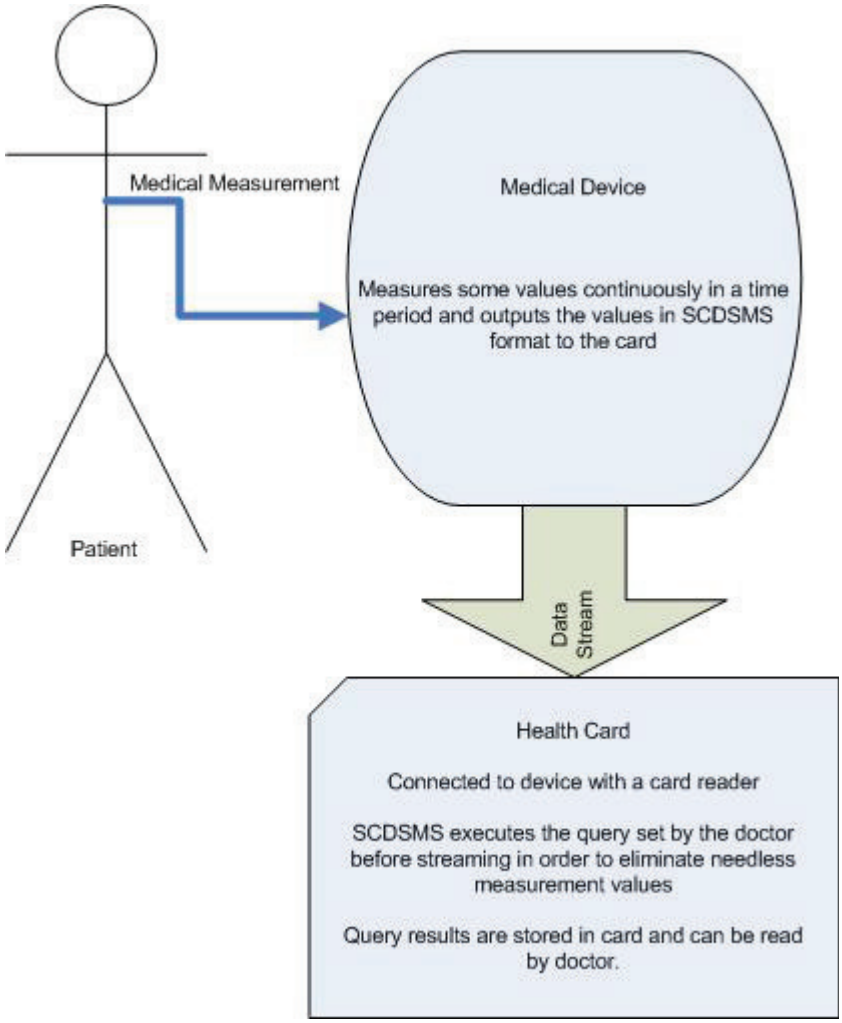

Figure 5.3. Health Card Example

Table 5.1. Query of Health Card Example

SELECT S1 WHERE (S2 > 0x06.C2) AND (S1 >100)

is BLOODPRESSURE attribute TAG in incoming tuples. C2 is AVGBLOODPRESSURE attribute TAG in context file. 0x06 is SFI of the Context File.

The Context File used in this example can be as shown in Table 5.2. In this figure the context file with 3 attributes is shown. These attribute TAGs are C1, C2 and C3. In query in Figure KK only attributes C1 and C2 are used. So before streaming the sorted index according to the C1 attribute is created in EEPROM by context manager. While streaming the C1 and C2 values are checked using this index file comparing with the S1 and S2 attributes in stream tuples. C2 attribute is also use the index file because C2 Is checked in the same tuple of the compared C1 in the tuple.

Table 5.2. Context File of Health Card Example

<Header of Context file>
C1 03 04 05 45 C2 05 34 35 56 56 56 C3 03 23 43 56
C1 03 04 05 89 C2 05 34 35 56 34 34 C3 03 23 67 98
C1 03 04 05 65 C2 03 34 35 98 C3 02 23 34

In Table 5.3, one data stream can be seen which includes 5 tuples. One tuple in the stream includes 3 attributes as S1, S2, and S3 where S1 is HEARTBEATRATE attribute and S2 is the BLOODPRESSURE attribute used in the query. S3 attribute is not used in the query, so it will be dropped immediately when tuples are taken into the card to be processed.

Table 5.3. Stream of Health Card Example

S1 03 04 05 48 S2 03 34 35 56 S3 03 23 43 56
S1 03 04 09 76 S2 05 34 35 56 34 56 S3 03 23 67 98
S1 02 04 97 S2 03 34 35 56 78 S3 02 23 34
S1 02 04 45 S2 05 34 35 56 43 76 S3 03 23 67 98
S1 03 04 05 98 S2 03 34 35 47 S3 01 46

The streaming data from the medical device which sends measurements of patient continuously in a time interval (e.g. 15 minutes), are queried with this query and the HEARTBEATRATEs are selected as the query set by the doctor. When patient is disconnected from the medical measurement device, doctor can view the results EF which includes desired HEARTBEATRATEs.

5.2. Congress Centers or Malls

SCDSMS also can be used by Congress Centers or Malls to send news or advertisements (sales news) according to the user's context information.

In Congress Centers, visitors generally finds the related conferences or informations from the booklets or some announcements in the exhibition centers. Our system offers a context aware system that makes user's know the conference or event news that they probably interested in and want to attend.

In Congress Centers example, there are 2 cases. In one case, a NFC device may be given to the visitors which can communicate the Vendor Machines around the Center. By this way, user puts his NFC device with his smart card, including his personal interest and job information in it, and gets the conference time table stream from the server machine which can communicate with the NFC device. Streaming table is queried by the SCDSMS. By this way, whole table is not recorded to the smart card (which may be impossible), and user can easily access the related event list from his smart card in NFC device.

In another case, mobile phones or PDAs can be used. In this case, SIM cards with SCDSMS in these mobile devices receive messages about new conference times or subjects, and via the SCDSMS running in their smart card, users will be acknowledged about only the related conference information messages. The Figure 5.5 which shows the Mall case example is so similar to this case in Congress Center example.

In Congress Center example the query in Table 5.4 can be used. In this case, the

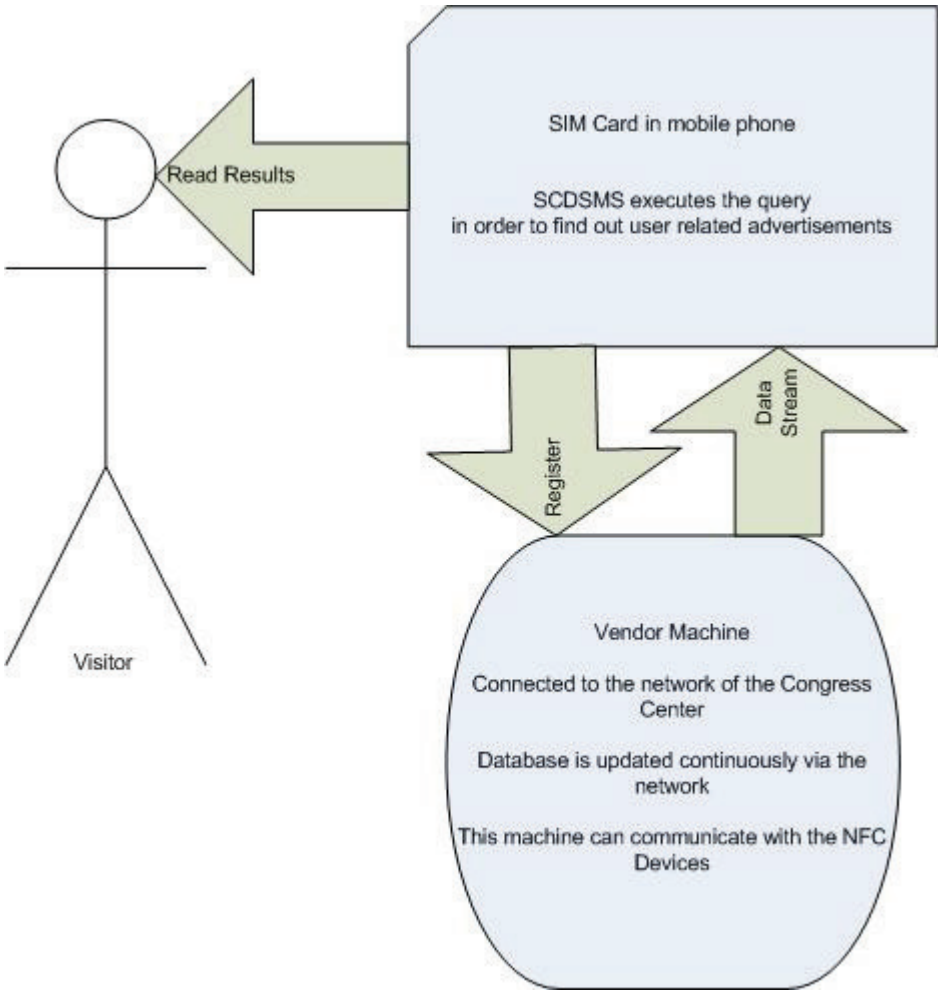

Figure 5.4. Congress Center- NFC example

identification numbers of the conferences which has duration smaller than 2 hours and its news has come in 60 seconds, is selected from the stream and recorded to the result EF in the user's smart card. In Table 5.4, S1 is HEARTBEATRATE attribute TAG in incoming tuples. S2 is BLOODPRESSURE attribute TAG in incoming tuples. C2 is AVGBLOODPRESSURE attribute TAG in context file. 0x06 is SFI of the Context File.

Table 5.4. Query of NFC Example

SELECT S1
WHERE (S2<2hr) AND (S1 ! = 0x05.A1)
WSINCE (60sec)

The Context File used in this example can be as shown in Figure **??**. In this figure the context file with 3 attributes is shown. These attribute TAGs are A1, A2 and A3. In query in Table 5.5 only attribute A1 is used which is Conference ID TAG. So before streaming a sorted index according to the A1 attribute is created in EEPROM by context manager. While streaming the A1 values are checked using this index file comparing with the S1 in tuple.

Table 5.5. Context File of NFC Example

<Header of Context file>
A1 04 31 34 31 37 A2 01 05 A3 01 09
A1 04 31 34 31 39 A2 03 04 05 06 A3 01 56
A1 04 31 34 31 39 A2 02 04 06 A3 01 34
A1 04 31 34 31 39 A2 04 31 32 35 36 A3 01 45
A1 04 31 34 31 39 A2 03 32 36 37 A3 01 98

In Figure **??**, one data stream can be seen which includes 3 tuples. One tuple in the stream includes 4 attributes as S1, S2, S3, S4 where S1 is Conference ID attribute and S2 is the Duration attribute used in the query. S3 and S4 attributes are not used

in the query, so they will be dropped immediately when tuples are taken into the card to be processed. The streaming data from the NFC communicable Machine are queried with this query and the conference IDs are selected as user's wish. When user puts off his NFC device from the source machine, he can view the results EF which includes desired conferece IDs.

Table 5.6. Stream of NFC Example

S1 04 31 34 31 37 S2 03 S3 01 04 S4 03 45 56 56
S1 04 31 34 36 36 S2 01 S3 03 02 06 07 S4 02 86 86
S1 04 31 34 31 33 S2 02 S3 04 02 03 05 06 S4 97 95

On the other hand, in Malls, people see advertisements in booklets, on the billboards or hear from some announcements in Markets or Malls. Our system offers a context aware system that informs users about the discount or advertisement news that they probably interested in and want to buy.

In Malls, users with SCDSMS in their smart cards can take the news and query them easily according to their preferences and context information. So, advertisements can find the correct users who are really interested in these sales. For example, if the

In Figure 5.5, a Mall example is given where a person in Mall registers the Mall's data source when he/she enters the building via a reply to a message sent to the user's mobile phone. So, Mall Server now knows that user wants to receive the advertisements and sales news while He/she is walking around in the Mall. He/She sets the query before streaming according to his/her needs or interests. While streaming in the Mall, the SCDSMS in user's SIM card elects the news via the query. As a result a alert can be shown to the user when a related sales is announced or he/she can get the results from the Result EF in his/her SIM card when he/she wants to look.

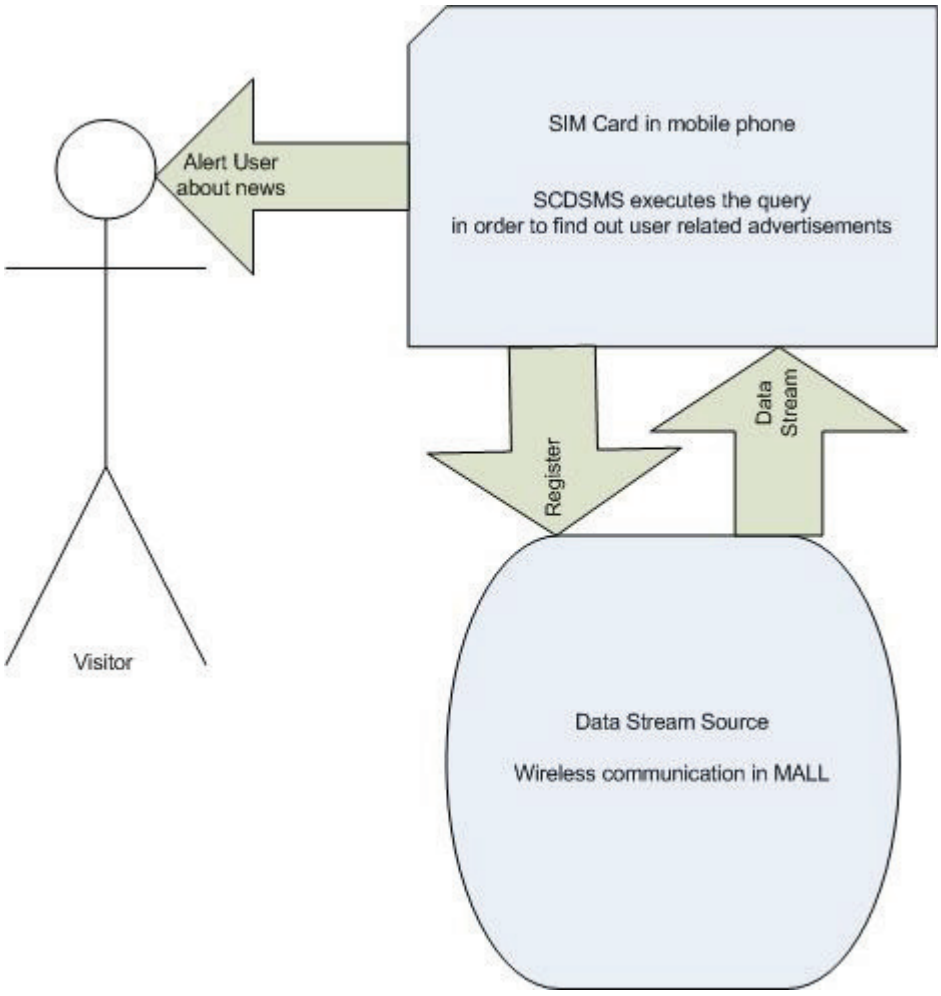

Figure 5.5. Mall Example

5.3. SIM Cards

GSM operator may also use SCDSMS by implementing a java card applet of SCDSMS for java card (GSM operators use generally java cards to be able to implement applets on it). By this way, operators broadcast messages can be eliminated according to the user's context data.

Today, GSM operators sends some information messages to users but they are not context aware messages or they are not streaming data flows. Thanks to the SCDSMS, GSM operators can send streaming data to users and make them be informed about their interests. In Figure 5.6, SCDSMS Health Card Application Components are shown.

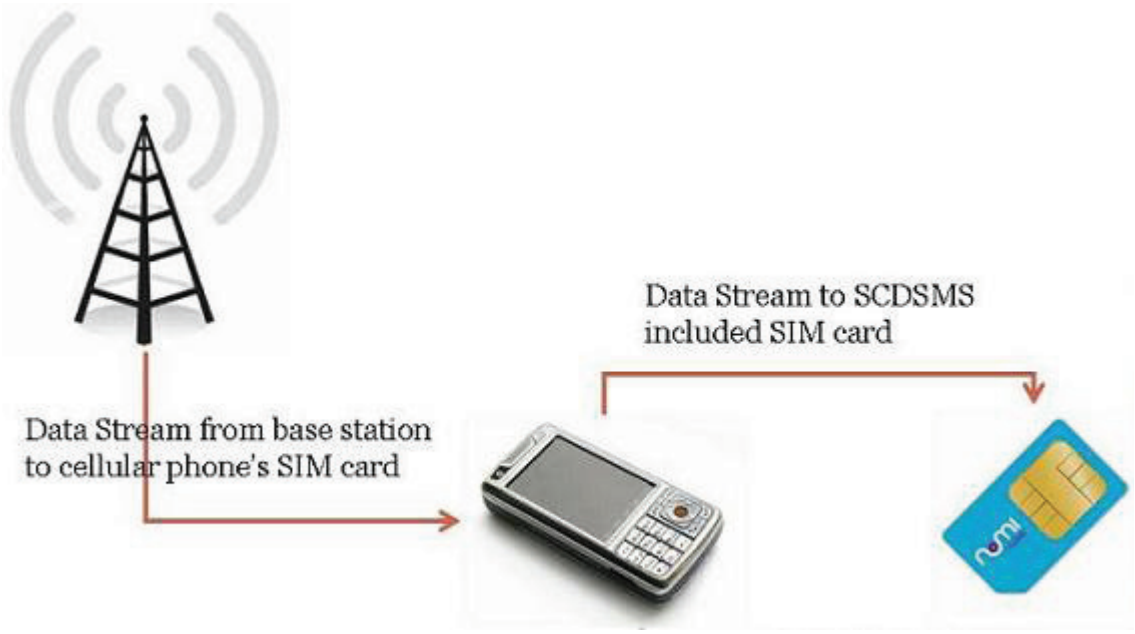

Figure 5.6. SIM Card Application Components

For example, a GSM operator may sends data streams of stock exchange updates to the users to inform them about their interested stocks. SCDSMS queries the data stream using the query set by the user which takes user's context data from the SIM card. All streams are meaningless for the user generally. User may want to know his stocks' situation or highly decreased/increased stocks.

As an example to the SIM Card applications the query in Figure 5.7 can be used.

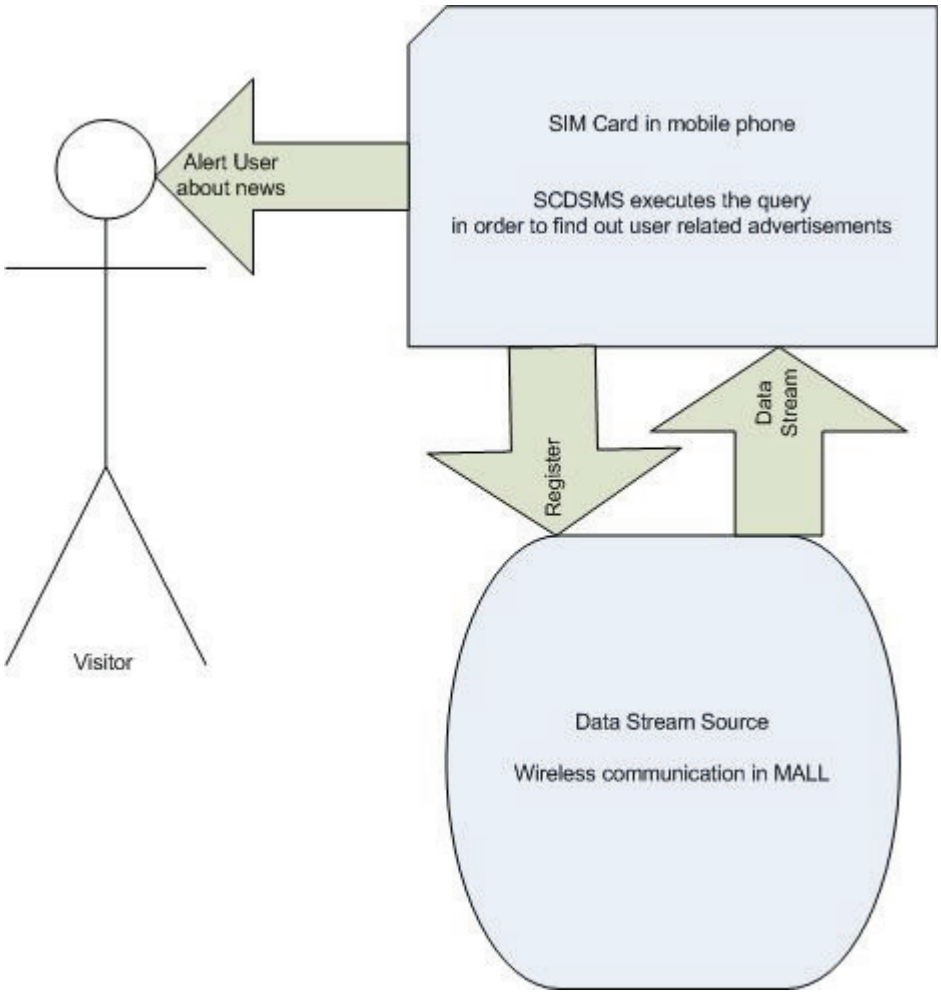

Figure 5.7. SIM Card Example

In this case, the stock id attribute is taken from the stream tuples if the condition in WHERE clause is satisfied. In WHERE clause there are 2 comparisons. First one is the comparison of stock id in incoming stream and the stock id in the context file. Second one is the comparison of stock value in the stream tuples and stock value in context file in the same tuple in previous comparison.

In Table 5.4, S1 is STOCKID attribute TAG in incoming tuples. S2 is STOCKVALUE attribute TAG in incoming tuples. C1 is STOCKID attribute TAG in context file. C2 is STOCKVALUE attribute TAG in context file. 0x07 is SFI of the Context File including interested stocks' information.

Table 5.7. Query of SIM Card Example

SELECT S1 WHERE (S1 == 0x07.C1) AND (S2 >0x07.C2)

The Context File used in this example can be as shown in Table 5.8. In Table 5.8 the context file with 3 attributes is shown. These attribute TAGs are C1, C2 and C3. In query in Figure KK only attributes C1 and C2 are used which are stock id and stock value TAGs. So before streaming one sorted index to the C1 attribute is created in EEPROM by context manager. While streaming the C1 and C2 values are checked using this index file comparing with the S1 and S2 attributes in stream tuples. There is no need to second index file for C2 attribute because always the same tuples C1 and C2 attributes are checked.

Table 5.8. Context File of SIM Card Example

<A Header of Context file>
C1 02 04 05 C2 03 53 27 56 C3 03 23 43 56 C1 02 09 65 C2 03 93 23 49
C3 03 23 67 98 C1 02 04 87 C2 03 12 63 98 C3 02 23 34

In Table 5.9, one data stream can be seen which includes 6 tuples. One tuple in the stream includes 2 attributes as S1ans S2 where S1 is stock id attribute and S2 is

the stock value attribute used in the query. The streaming data from the base station of the GSM operator which sends updates of stock exchange values continuously all day long, are queried with this query and the stock ids are selected as the query set by the user. When user want to check the condition of the results he can view the result EF or system can be designed to alert user when a searched situation is found.

Table 5.9. StreamofSIMCardExample

S1 02 04 45 S2 03 13 35 56 S1 02 04 36 S2 03 47 35 56 S1 02 04 87 S2 03 46 35 98 S1 02 04 23 S2 03 93 35 56 S1 02 06 07 S2 03 95 35 56 S1 02 09 56 S2 03 53 45 79

6. EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

6.1. Experiment Environment

Some challenging parts of SCDSMS are implemented as SCDSMS-beta and run on simulator and also a smart card. Used tools and methods are described in this part.

Table 6.1. Smart Card Usage Numbers in Applications

Used Softwares	Used Hardwares
KEIL - microVision3 IDE (OS development environment for smart cards with C compiler)	Infineon Smart Mask (Test cards with updatable OS feature)
Infineon CC Top (Card Reader simulator software)	Omnikey Card Reader familiy (contact base, contactless and usb card readers)
Raisonance ProxiLAB3 Smart Card Tester (Contactless Smart Card testing software)	Raisonance ProxiLAB3 Emulator (Contactless Smart Card and Reader Emulation Device)
AKiS (Native Card Operating System Developed by TUBITAK-UEKAE)	

After AKiS compilation in KEIL, AKiS with SCDSMS algorithms are uploaded into the KEIL Card simulator. Using CC Top software data files are uploaded into same KEIL card simulator. CC Top software uploads data files as shown snapshot view in Figure 6.1. In Figure 6.1 Raisonance Test tool snapshot can be seen while measuring the execution time of the Stream command.

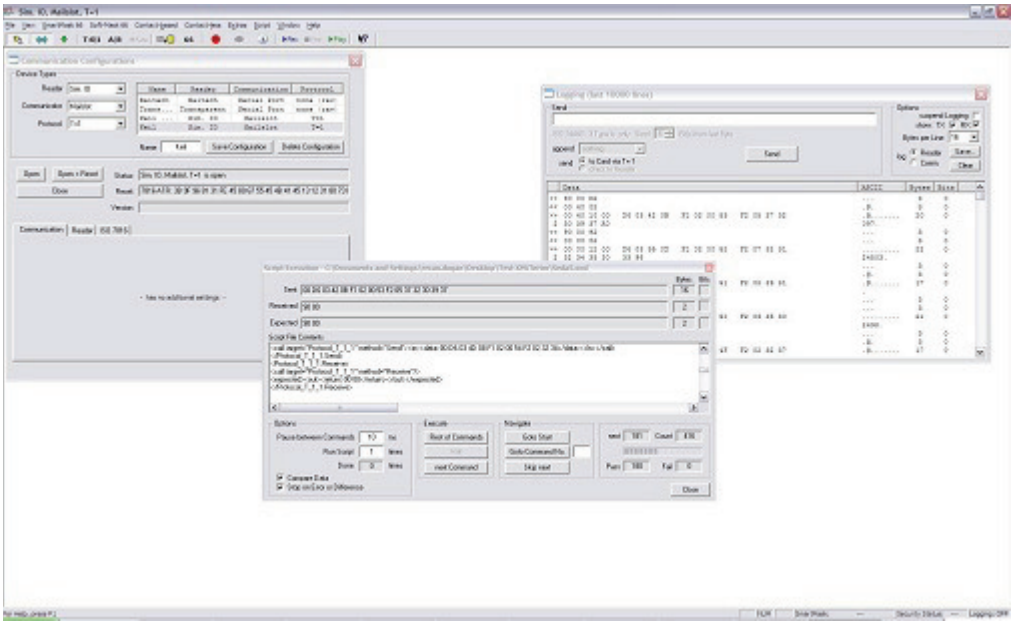

Figure 6.1. CC Top Software

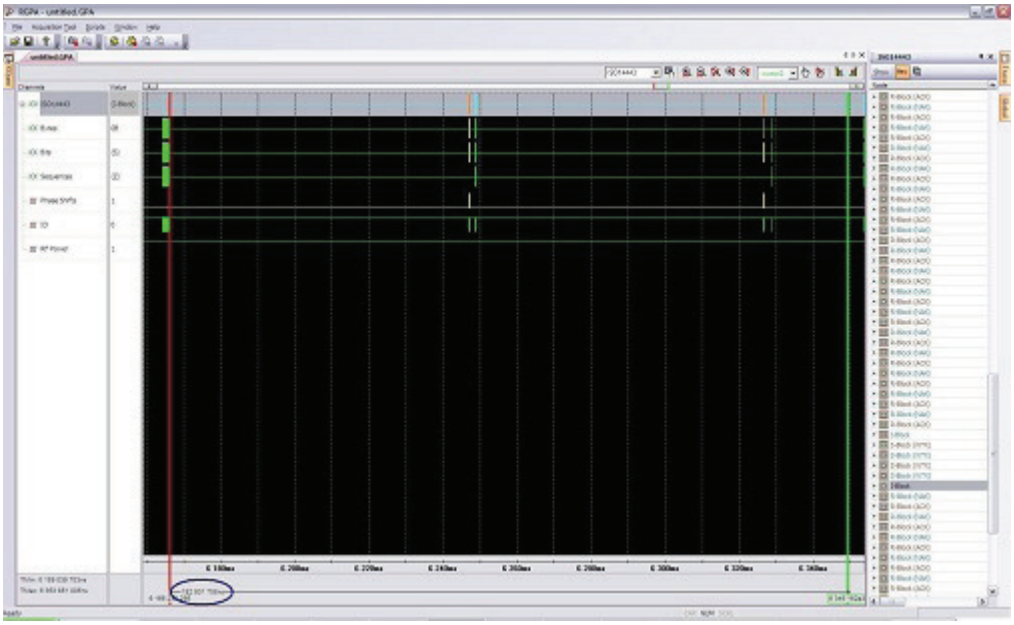

Figure 6.2. Raisonance Test Tool

AKiS is a native card operating system which is developed by TUBITAK-UEKAE (National Cryptology Institute of Turkey). It implements almost all standards of 7816 standards. It can store data, process data, encrypts and decrypts data. It can communicate encrypted or not according to the CLA selection in the command.

It has both contact based and contactless interfaces both can run in same chip sharing same EEPROM and RAM. Its command set can be found in the Appendix-1. AKiS communicates with the terminal using only I/O and CLK as the communication protocol defined in ISO 7816-4.

AKiS card has a life cycle to differentiate the authorities and application from each others. The life cycle does not support going backward phase. This prevents the card to be reinitialized by unauthorized people. This life cycle of AKiS and some more detail about it can be seen in Appendix 1.

AKiS has DF and EF structure to hold data. The root is MF as described in ISO 7816 standard set. A file structure example of AKiS can be seen in Appendix-1.

6.2. Compared Algorithms and Data Structures

SCDSMS has 2 challenging algorithms for smart cards. These are Sorting and Searching algorithms. It is important to use RAM efficiently and complete the execution in a reasonable time. Due to these considerations, Sort and Search algorithms are implemented as functions in AKiS and executed to test limitations, execution durations with different data sizes.

Also used data structures of both these algorithms and other parts of SCDSMS are considered to use less RAM area. Used data structures of these algorithms are tested and seen the expected results. By this way, we can conclude other parts also use the expected RAM areas and all RAM usages are calculated even though some of them are not implemented in card but it is proved that they will use the expected size when implemented.

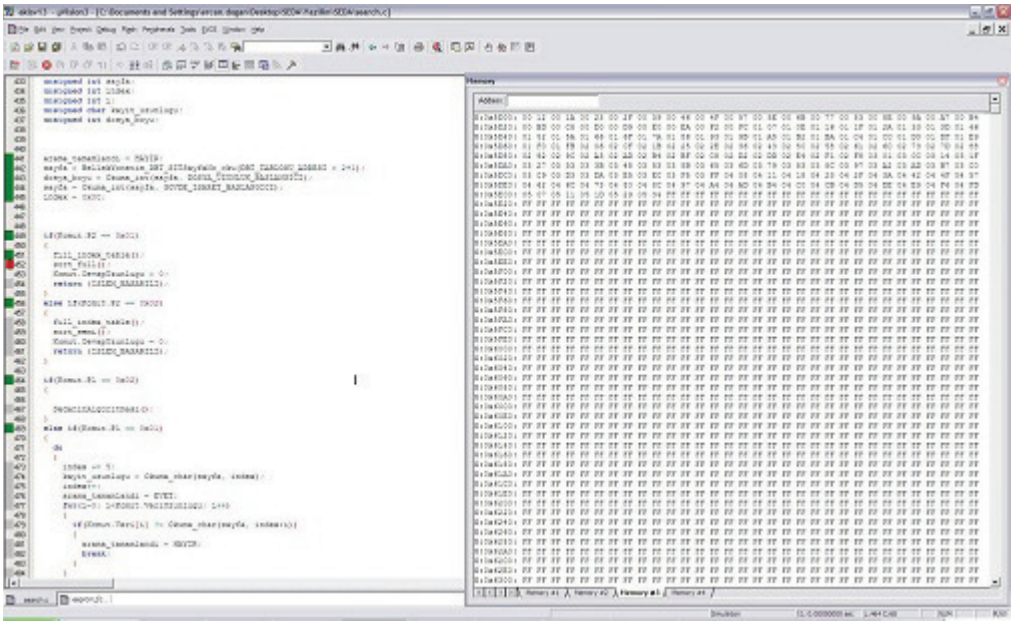

Figure 6.3. KEIL microVision3 IDE

Searching and Sorting operations are implemented in different ways and with different data structures in order to see how much efficiency is obtained via using the selected algorithms and data structures.

6.2.1. Sorting Algorithms and Data Structures

Two different Sorting methods are implemented. Both methods use an index data as input and creates another index data as output. The output index data can be Full-Sorted or Semi-Sorted according to the method applied. Both methods use Bubble Sort Algorithm due to its very small space consumption than others. Space complexity of Bubble Sort is O(1).

In these pseudo codes comparison details are not shown. "isBigger" function compares strings character by character. Get Pointed Value function gets the data according to the index. So, as seen, in Semi-Sort method, comparisons are limited up to 3 character comparisons. Both sorting methods take an index file from EEPROM as input. This index file includes 2 byte addresses as shown in the KEIL microVision3 IDE snapshot picture in Figure 6.2.1.

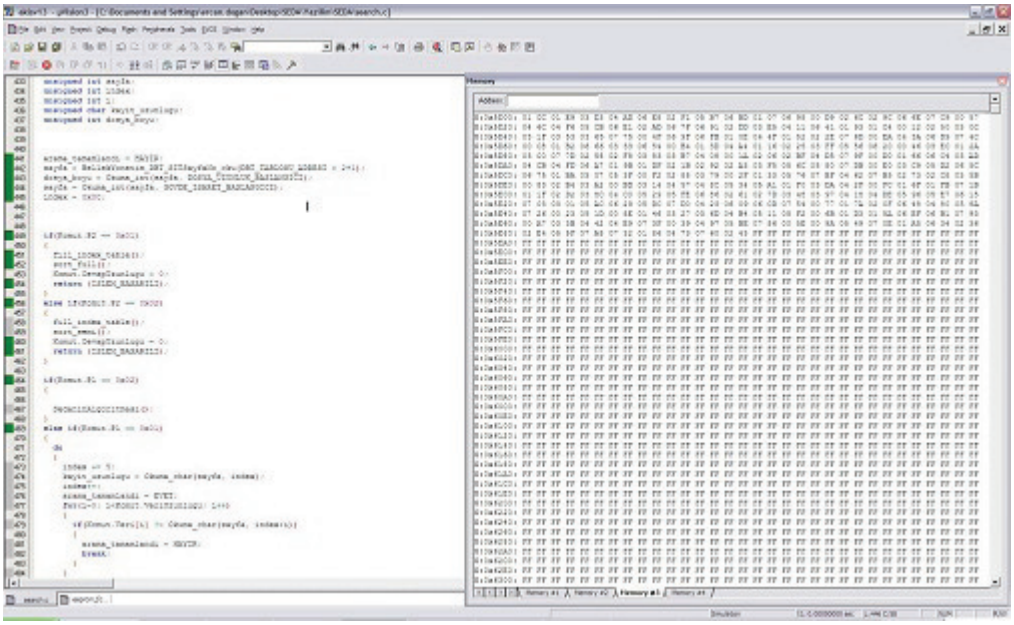

Figure 6.4. After FULLSORT operation sorted index file

After FULLSORT operation sorted index file in EEPROM of smart card looks as in Figure 6.2.1

As shown in the pictures of EEPROM view of smart card, first element of unsorted index file is 00 12 in first picture whereas in second picture sorted index file first element is 01 CC. This means the value in data file which is located in 01 CC address of the data file is the smallest value in the data file.

6.2.2. Searching Algorithms and Data Structures

Searching methods implements 2 different algorithms. One is Divide and Conquer Algorithm and other one is the Linear Search Algorithm. Linear Search Algorithm is the trivial one, which searches the file elements one by one. Divide and Conquer Algorithm uses a sorted index file which points the elements of data file in sorted order.

First 12 bytes are header information of data EF. Next bytes are TLV form tuples: One tuple is:

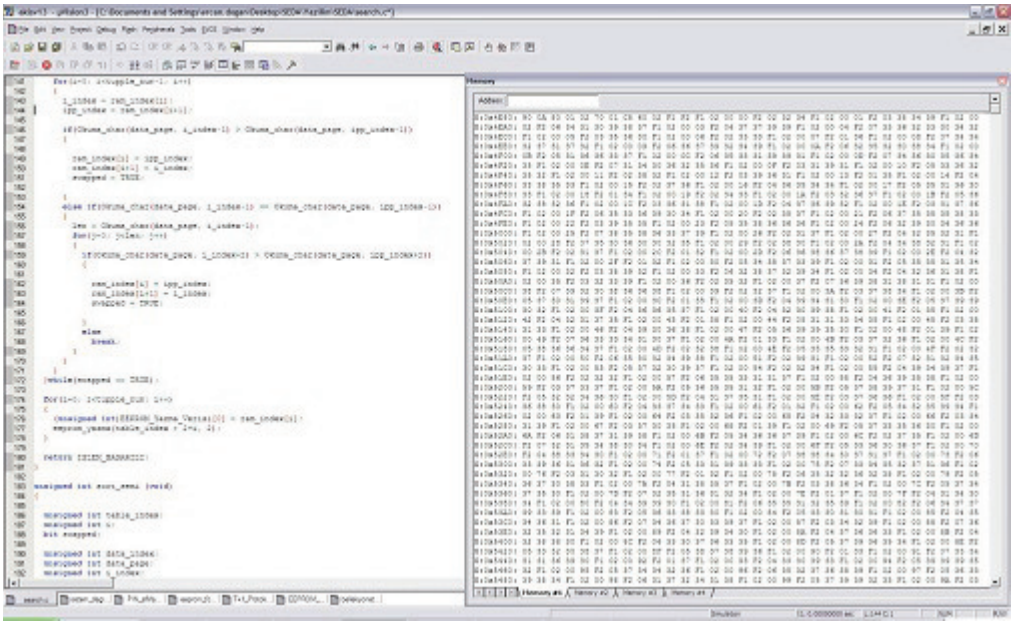

Figure 6.5. Context File view in EEPROM

F1 02 00 02 F2 06 34 31 30 39 38 37

in which F1 and F2 are attribute TAGs. After TAGs first bytes are length values which are followed by the values. In this tuple first attribute value is "02" and second attribute value is "34 31 30 39 38 37". All representations are in hexadecimal form.

6.3. SCDSMS-beta Test Cases

In test cases 4 types of Context files, 4 types of Streams, 2 types of Queries are used. Their brief features are given in Figures 6.3, 6.3 and Table 6.2. The context files can be found in Appendix-2. These Context Files, Streams and Queries are executed in their combinations and results are seen in Results section.

Table 6.2. Queries

Query 1	SELECT F1 WHERE F1 != ContextFile.F1
Query 2	SELECT F1 WHERE F1 > ContextFile.F1

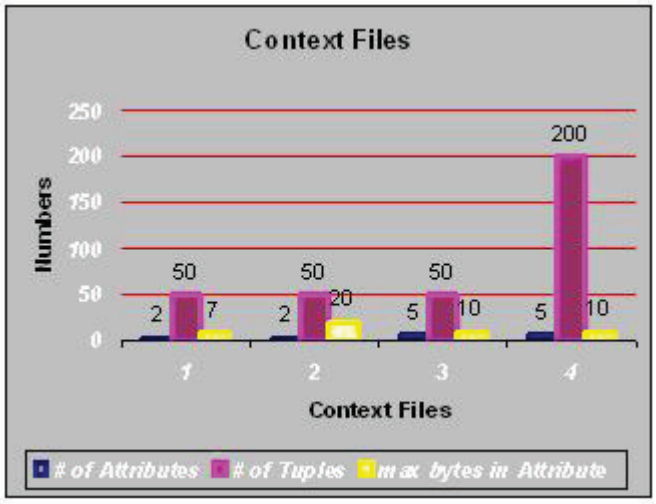

Figure 6.6. Context Files

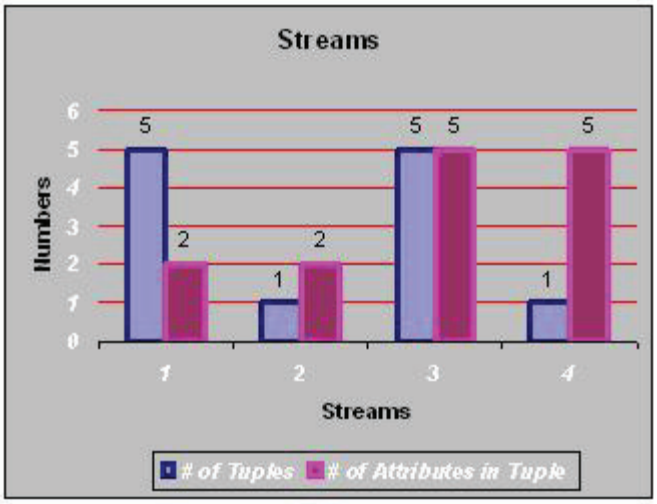

Figure 6.7. Streams

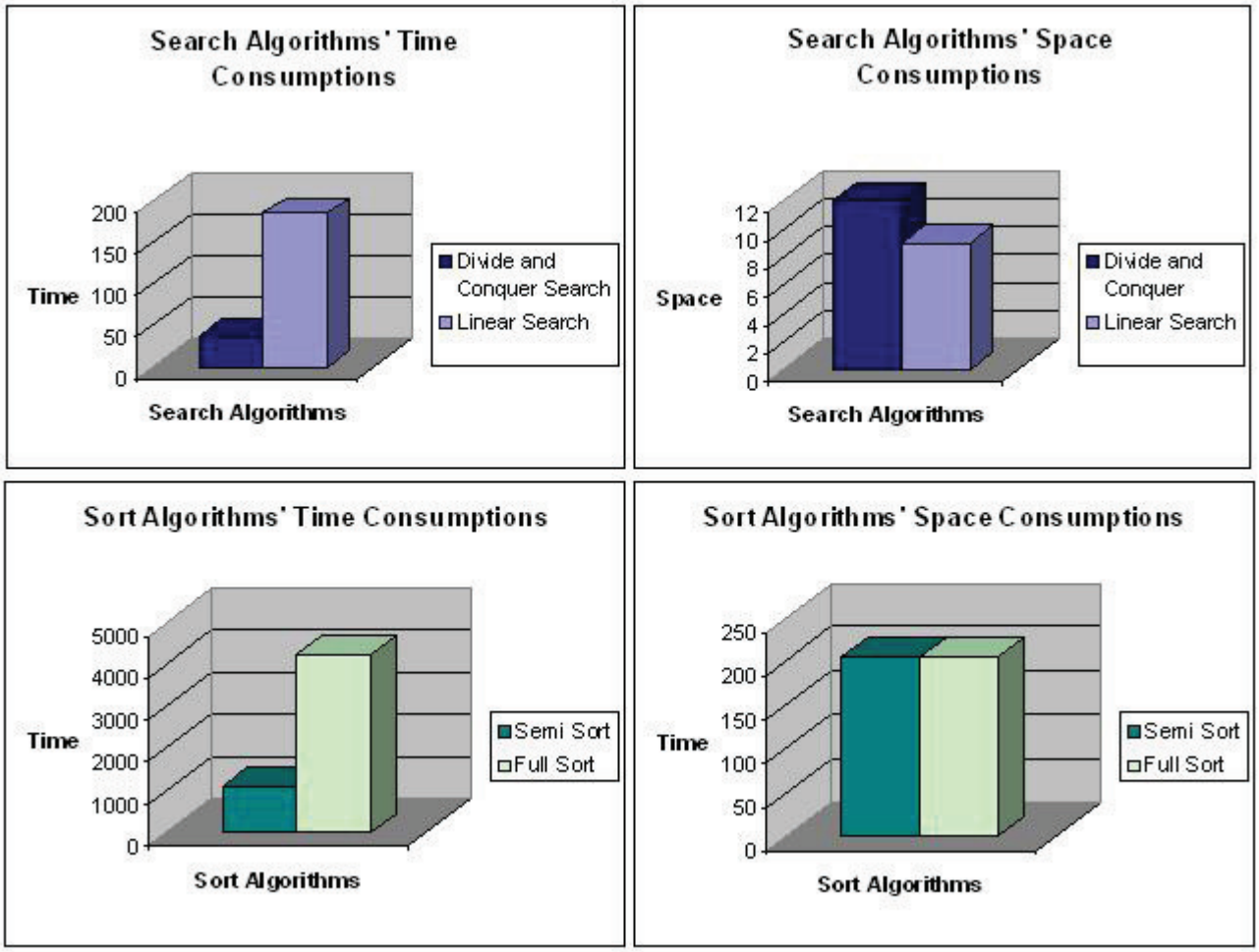

Figure 6.8. Comparisons of Algorithms

6.4. Results

2 kinds of experiments are done, which can be grouped in two as "Algorithm Comparisons" and "SCDSMS-beta tests ". Time results are obtained using Raisonance Test tool. The Raisonance Test tool's measurements IDE can be seen in Experiment Environment.

First experiments is done to use appropriate algorithms and data structures in SCDSMS-beta. Sorting and Searching algorithms are compared using mentioned data structures which are generally address arrays. Algorithm Comparisons results can be seen in Figure 6.4. Over a data file which includes 2 attributes, 200 tuples and a 3K size these results are obtained.

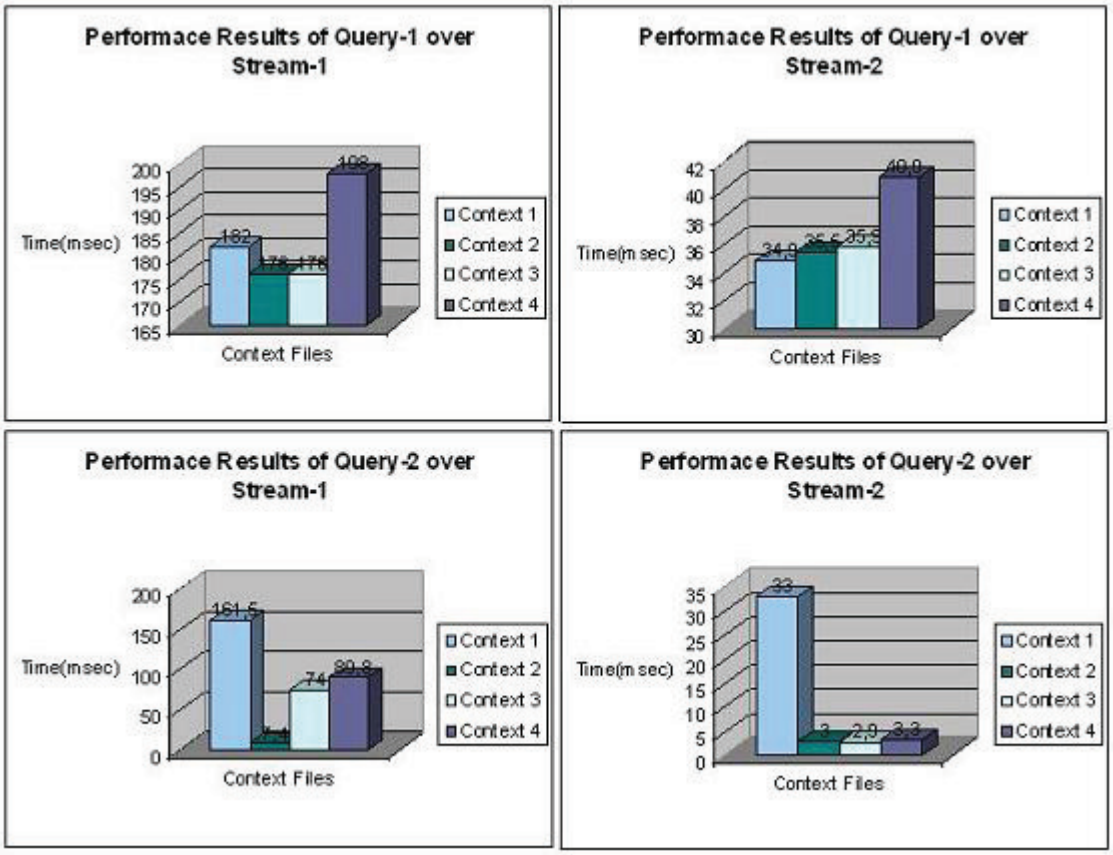

Figure 6.9. Performance Results of Stream Command

Due to the results of Algorithm Comparisons experiment, Divide and Conquer Search Algorithm and Semi-Sort Algorithm are used in SCDSMS-beta. Also by this experiments, it was seen that using address arrays makes RAM area usage lower.

Using 4 type Context File, 2 type Query , 4 type Stream as mentioned above section, the results in Table 6.4 are obtained.

SubmitQuery command executes sorting algorithm for the context file's selected attribute over the address list of them. Result of SubmitQuery command execution is seen in Figure 6.4.

According to the experimental results of SCDSMS-beta, it is seen that SCDSMS is an applicable system for smart cards especially for the applications mentioned like SIM cards, Health cards and Conference cards. Smart cards have very small RAM and EEPROM areas that make data storage capability lower. So, it is better to delete data

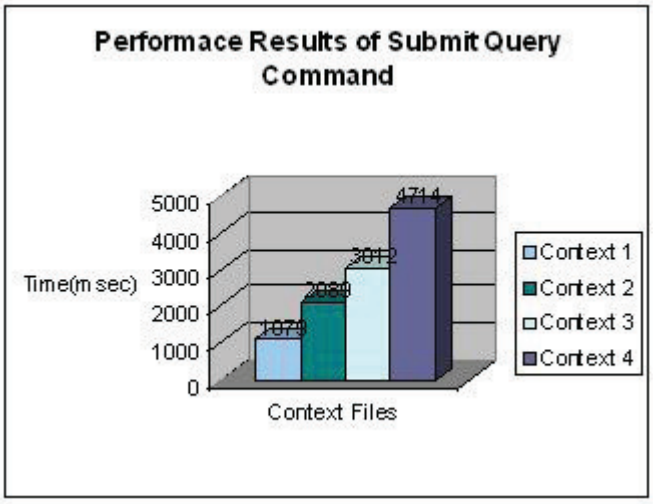

Figure 6.10. Performance Results of Submit Command

after queried rather than storing it in little EEPROM. Data Streaming is appropriate for smart cards because the data in stream are not stored in these systems. Processing speed of smart card is fast enough to calculate the SCDSMS operations while streaming. SCDSMS uses RAM space efficiently and can handle streaming data.

Consequently, we have seen in the experimental results that the processing time, RAM usage and EEPROM usage of SCDSMS-beta is appropriate for data streaming applications mentioned.

7. CONCLUSIONS

By SCDSMS, a new research area is opened for smart cards. SCDSMS is a smart card data stream management system which is based on context awareness. In experiments of the pilot application SCDSMS-beta, AKiS is used. AKiS is a native card operating system which is developed by TUBITAK-UEKAE.

Database systems are forced to support data streaming by the new generation applications. Data streaming is used in much more areas than before. 3G applications, financial data

There are some constraints to implement a DSMS for smart cards. These constraints can be listed as:

- RAM limitation (very small RAM compared to PCs, up to 10K)
- EEPROM limitation (very small stable storage compared to PCs, up to 160K)
- EEPROM write limitation (write operation in EEPROM is very costly in time)
- Power limitation (smart cards does not have a power supply inside)
- Command-Response working principle of smart cards
- Limited command size, approximately 256 byte

SCDSMS is designed according to these constraints. Data stream is taken into the RAM, RAM is small and useless data should be deleted and query should be executed immediately to take new stream into RAM.

To the best of our knowledge, there are not any study about stream data management on smart cards, so SCDSMS project opens a new area in smart card applications. Some may wonder why to hold the part of database in card or why to implement SCDSMS in card, or why to need a stream data manager in card. These reasons can answer the questions:

- **Why to hold database in the card?:**
  (i) Data will be highly available when needed. (Anywhere, anytime, without asking surgery information to the Hospital to look up its servers for health card case)
  (ii) Storing data in servers (in health card case in Hospital Servers) may hurt privacy.
  (iii) Maintaining a centralized database (in health card case in Hospital Servers) is fairly complex due to the large data variety.

- **Why to embed SCDSMS in the card?:**
  (i) Provides availability (Card can be queried by any terminal without a software suite need)
  (ii) Provides privacy (Data are queried in the card, so private data are not given outside)

- **Why to use data streaming to the card?:**
  (i) Avoids EEPROM overflow by continuous query execution principle of SCDSMS
  (ii) Only needed data are stored so data mining among the stored data are easier

Some important and challenging parts of the SCDSMS is implemented on AKiS called SCDSMS-beta to prove project's applicability and evaluate its performance. Two kinds of sorting and searching algorithms are implemented and compared to select the optimum algorithms to use in SCDSMS-beta. Rather than writing the information in EEPROM again into the tables in EEPROM, address tables are used to reduce data redundancy in the EEPROM. Also, sorted index tables which hold the addresses of the data in EEPROM are used to reach searched data easily by using appropriate searching algorithm over these sorted index tables. These index tables are sorted rather than the sorting the EF itself according to the searched item in the EF. This provides efficiency in time and space consumption.

According to the experimental results of SCDSMS-beta, it is seen that SCDSMS is an applicable system for smart cards especially for the applications mentioned like

SIM cards, Health cards and Conference cards. Smart cards have very small RAM and EEPROM areas that make data storage capability lower. So, it is better to delete data after queried rather than storing it in little EEPROM. Data Streaming is appropriate for smart cards because the data in stream are not stored in these systems. Processing speed of smart card is fast enough to calculate the SCDSMS operations while streaming. SCDSMS uses RAM space efficiently and can handle streaming data.

8. FUTURE WORK

SCDSMS is a simple stream data management system for smart cards and it's pilot version SCDSMS-beta works successfully on AKiS. It includes SELECT and PROJECT operations of Relational Algebra.

Other important functionalities like JOIN can be implemented if it can be reduced to use the RAM space efficiently. For our example cases there exists single data stream source, so JOIN operation is not needed. For, some cases this operator maybe useful but also forces the boundaries of the card's RAM and EEPROM. Some algorithms can be designed to apply JOIN command in the SCSQL. Besides JOIN operation, also AGGREGATION or VIEW operations can be implemented for some different cases. All relational algebra operations should be optimized (as we have done over the subset of it) if they wanted to be implemented for smart cards. At this point, the important question is "Does it worth to apply them if they decreases the performance and usability of SCDSMS?". For another study, implementing all these relational algebra operations can be taken into consideration with taking the all constraints of smart card into account. By this study, a CSQL like standard can be published as a kind of 7816 standard as SCSQL (Smart Card Stream Query Language).

On the other hand, the context awareness on smart card can be considered in detail for more application scenarios. This part of SCDSMS is not covered in detail because the first important aim was to prove the card's stream data management capability. So, context awareness is a valuable issue that should be covered in another study.

Besides, we have not tested proposed applications in a real environment using real data. For example, proposed health card application can be tested in a real hospital using real medical device output data. As a result, applicability of the system and the usability by doctors and patients can be observed.

APPENDIX A: AKiS

AKiS is an abbreviation of "Akilli Kart Isletim Sistemi" which is a product of TUBITAK-UEKAE. It is a Native Card Operating System. AKiS components are:

- Memory Manager
- File Manager
- Command Interpreter
- Communication Handler

Message is received by UART which is managed by communication handler in Card Operating System(COS). The message comes in TPDU format which is mentioned above. Incoming TPDU packet is analyzed and block type decision is made by the communication handler. TPDU can include 3 different types of block, named R, S and I block. R and S blocks are used to control the protocol. I block carries the command which is transmitted to the command interpreter and executed in COS. When command execution is finished, communication handler sends the answer to the reader via UART. If the command is related with the file system, command interpreter calls the file manager. File manager is responsible for the operations in the file field which is in the EEPROM. Memory manager is used to open new file, close file, delete page and attach new page.

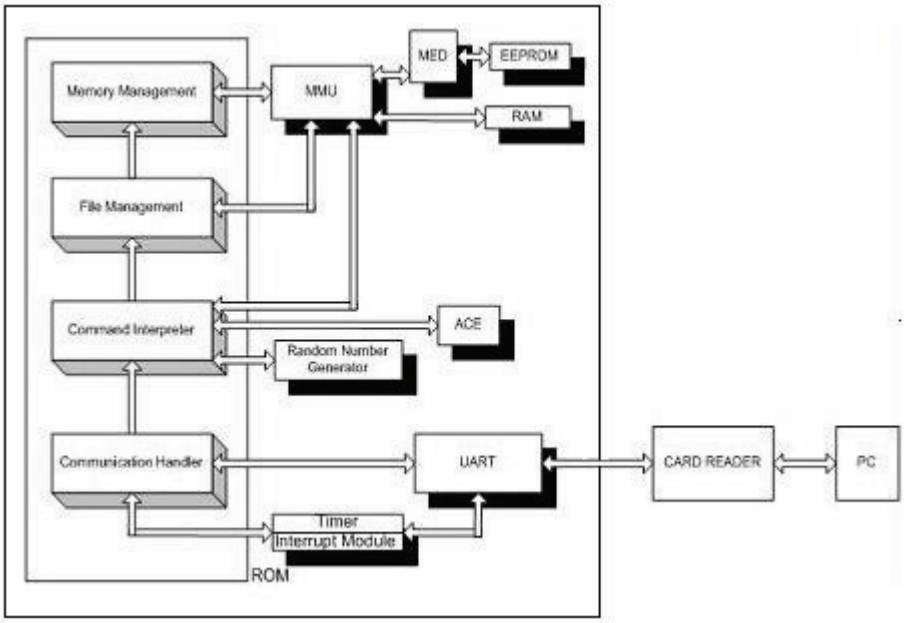

Figure A.1. AKiS components and environment

APPENDIX B: SAMPLE TEST FILES

In SCDSMS-beta tests, four Context Files are used. The Context File-1 is given below, other files can be found in the attached CD. Also, in tests four types of streams are used and Stream-1 can be seen below.

Context File-1

11 22 33 44 55 66 77 88 99 AA BB CC

F1 03 39 35 33 F2 02 34 32

F1 05 34 38 38 31 39 F2 03 38 30 31

F1 05 34 32 31 33 36 F2 01 38

F1 06 39 39 32 35 36 36 F2 05 37 36 37 39 36

F1 04 36 35 32 38 F2 05 36 36 39 35 32

F1 02 33 32 F2 04 37 30 39 38

F1 06 37 32 39 32 34 30 F2 04 36 34 39 30

F1 04 39 31 39 33 F2 02 39 33

F1 03 35 34 31 F2 05 36 36 35 37 31

F1 04 38 31 31 36 F2 03 36 37 30

F1 02 39 39 F2 04 36 35 35 35

F1 03 32 37 30 F2 06 32 39 37 33 33 34

F1 06 33 37 31 37 39 37 F2 05 38 33 37 32 38

F1 04 31 39 31 36 F2 02 37 34

F1 01 37 F2 01 39

F1 05 33 30 36 32 33 F2 03 36 37 36

F1 01 34 F2 03 38 35 34

F1 05 32 38 38 34 36 F2 02 39 32

F1 05 38 30 34 34 32 F2 02 31 35

F1 01 38 F2 04 34 30 37 30

F1 06 32 31 32 37 33 31 F2 01 32

F1 03 33 30 36 F2 02 31 38

F1 03 33 34 32 F2 03 31 32 36

F1 02 34 39 F2 03 38 36 38

F1 05 31 31 36 39 32 F2 02 37 39

F1 04 31 38 35 33 F2 02 39 39

F1 02 39 34 F2 03 39 39 38

F1 05 35 34 33 37 34 F2 03 34 30 30

F1 02 34 30 F2 01 35

F1 02 32 30 F2 05 35 37 31 37 31

F1 01 33 F2 03 32 30 36

F1 05 32 39 35 33 39 F2 05 31 35 33 32 37

F1 04 37 36 36 37 F2 03 31 33 36

F1 03 38 37 37 F2 02 39 34

F1 03 32 32 35 F2 05 32 35 37 36 33

F1 06 37 37 37 33 34 33 F2 02 31 38

F1 02 39 31 F2 01 37

F1 02 38 32 F2 01 35

F1 01 39 F2 02 33 37

F1 02 31 31 F2 01 31

F1 05 33 35 32 33 39 F2 04 37 31 34 30

F1 03 31 31 35 F2 02 33 34

F1 03 36 38 31 F2 06 39 30 39 31 37 34

F1 05 37 38 30 35 35 F2 01 32

F1 06 33 33 34 38 35 38 F2 05 34 32 33 39 34

F1 02 31 38 F2 05 33 32 39 32 35

F1 03 33 37 37 F2 02 37 37

F1 04 36 39 30 34 F2 01 39

F1 01 33 F2 05 38 35 34 33 34

F1 01 39 F2 03 39 34 39

Stream-1

00 02 00 00 55 F1 08 31 31 31 31 31 31 31 31 F2 05 32 32 32 32 32 F1 07 31 31 31 31 31 31 31 F2 05 33 33 33 33 33 F1 09 31 31 31 31 31 31 31 31 31 F2 05 34 34 34 34 34 F1 0A 31 31 31 31 31 31 31 31 31 31 F2 05 35 35 35 35 35 F1 06 31 31 31 31 31 31 F2 05 36 36 36 36 36

REFERENCES

1. Chan, A. T. S., "Mobile cookies management on a smart card", *Commun. ACM*, Vol. 48, No. 11, pp. 38–43, 2005.

2. Geven, A., P. Strassl, B. Ferro, M. Tscheligi, and H. Schwab, "Experiencing real-world interaction: results from a NFC user experience field trial", *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pp. 234–237, ACM, New York, NY, USA, 2007.

3. Chavira, G., S. W. Nava, R. Hervás, V. Villarreal, J. Bravo, S. Martín, and M. Castro, "Services through NFC technology in AmI environment", *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pp. 666–669, ACM, New York, NY, USA, 2008.

4. Omar, S. and H. Djuhari, "Multi-purpose student card system using smart card technology", *Information Technology Based Higher Education and Training, 2004. ITHET 2004. Proceedings of the FIfth International Conference on*, pp. 527 – 532, 31 2004.

5. Pleunis, J. and M. Stala, "Multi-application smartcard user interface", *EUSAI '04: Proceedings of the 2nd European Union symposium on Ambient intelligence*, pp. 29–30, New York, NY, USA, 2004.

6. Vrancart, A., "Global demand for cards continues", *Card Technology Today*, Vol. 19, No. 3, pp. 10 – 11, 2007, `http://www.sciencedirect.com/science/article/B6W6X-4NN6N6K-K/2/cd3260248536413ed82f38047901686a`.

7. "*Global Smart Card Usage Statistics*", `www.cardlogix.com`, Extracted: June 2010, `www.cardlogix.com`.

8. 7816-2:1999, I., *Identification cards Integrated circuit(s) cards with contacts Part 2: Cards with contacts - Dimensions and location of the contacts*, International Organization for Standardization, Geneva, Switzerland, 1999.

9. 7816-4:1999, I., *Identification cards Integrated circuit(s) cards with contacts Part 4:Organization, security and commands for interchange*, International Organization for Standardization, Geneva, Switzerland, 1999.

10. Pucheral, P., L. Bouganim, P. Valduriez, and C. Bobineau, "PicoDBMS: Scaling down database techniques for the smartcard", *VLDB J.*, Vol. 10, No. 2-3, pp. 120–132, 2001.

11. Bolchini, C., F. Salice, F. A. Schreiber, and L. Tanca, "Logical and physical design issues for smart card databases", *ACM Trans. Inf. Syst.*, Vol. 21, No. 3, pp. 254–285, 2003.

12. Ramamritham, K. and R. Sen, "DELite: database support for embedded lightweight devices", *EMSOFT*, pp. 3–4, 2004.

13. Anciaux, N., L. Bouganim, and P. Pucheral, "Smart Card DBMS: where are we now?", Research report, Prism Laboratory, 2006, `http://hal.inria.fr/inria-00080840/en/`.

14. 7816-7:1999, I., *Identification cards Integrated circuit(s) cards with contacts Part 7:Interindustry commands for Structured Card Query Language (SCQL)*, International Organization for Standardization, Geneva, Switzerland, 1999.

15. Arasu, A., B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, and J. Widom, "STREAM: The Stanford Stream Data Manager", *IEEE Data Eng. Bull.*, Vol. 26, No. 1, pp. 19–26, 2003.

16. Abadi, D. J., D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management", *The VLDB Journal*, Vol. 12, No. 2, pp. 120–139, 2003, `http://portal.acm.org/citation.cfm?id=950485`.

17. Babcock, B., S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems", *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 1–16, ACM, New York, NY, USA, 2002.

18. Liu, B., A. Gupta, and R. Jain, "MedSMan: a streaming data management system over live multimedia", *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pp. 171–180, ACM, New York, NY, USA, 2005.

19. Sullivan, M. and A. Heybey, "Tribeca: a system for managing large databases of network traffic", *ATEC '98: Proceedings of the annual conference on USENIX Annual Technical Conference*, pp. 2–2, USENIX Association, Berkeley, CA, USA, 1998.

20. Baldauf, M., S. Dustdar, and F. Rosenberg, "A survey on context-aware systems", *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 2, No. 4, pp. 263–277, June 2007, `http://dx.doi.org/10.1504/IJAHUC.2007.014070`.