A Personal and Portable Database Server : the CQL Card

Pierre Paradinas[1&3] and Jean-Jacques Vandewalle[1&2]

[1] Rd2p, Recherche et Développement Dossier Portable, CHRU Calmette, rue du Pr. J. Leclerc, 59 037 Lille Cédex, France, tel. : (33) 20 44 60 44, fax : (33) 20 44 60 45, emails : pierre@rd2p.lifl.fr - jeanjac@rd2p.lifl.fr

[2] Université Laval, Dép. d'Informatique, Québec, Canada, G1K 7P4, tel. : (1) 418 656 2580, fax : (1) 418 656 2324, email : jeanjac@iad.ift.ulaval.ca

[3] Gemplus, B.P. 100, 13 881 Gémenos, France, tel. : (33) 42 32 50 30, fax : (33) 42 32 50 44, email : pierre@gemplus.fr

Abstract. Database applications and technologies are of central importance in many information systems a person may encounter. To obtain services, the end-users are required a smart card (plastic card containing a microcomputer), which is a device providing information about the user's identity and some related personal data. It can be updated and loaded with new data that will be used during further sessions. Moreover the data contained into the smart card can be used by other information systems, the data are carried away from a site to another. The individual mobility increases the need for a person to carry information about himself anywhere and at any time. For services providers, such as health professionals, it is essential to access to this information stored on several information systems. In many applicative areas, to provide different information systems linked and networked is a real challenge. Based on personal information about the bearer, the smart card is a key to access to different information systems and a mean to share and interchange data. The smart cards are evolving towards personal database functions. We briefly present the technology of smart cards, then we introduce a new approach : the CQL card (for Card Query Language). This card integrates the concepts of the Database Management Systems. Database engine is carried out by the card microcomputer, the card is a new database machine. It manages "users" entities which handle different "objects" according to their "privileges". CQL, a subset of SQL, is used to communicate with the card. Views enable sharing data among information systems. Access rights and privileges guarantee the data privacy. To ease the integration of this portable database we have implemented an ODBC driver enabling smart card connectivity with many applications and DBMS's. The smart card as a personal and mobile data server is a new support for databases, it involves new applications, such as health care cards or administrative document cards, and new ways of carrying and interchanging information.

Keywords. Personal Database, Smart Card, Database Interchange, SQL, ODBC.

1 The Database Card : a Personal and Portable Server

Considering the end user, information systems are becoming more and more important. They store a growing amount of personal information and they implement computing operations growing in complexity. Their importance in our life is obvious (in administrations, corporations, banking operations, shopping, travels, *etc.*). Some people take fright about the use of these information : may I know the data that an organization have on me? These data are they correct? How are they process? Are these data interchanged with other organizations? Is the data privacy guaranteed?... This paper addresses the problem of the interchange of data among different information systems. The key points we want to underline are :

- Data interchange is necessary. The end result of many computing processes depends on a good integration of information emerging from different information systems. For example, in medical information systems, such as in many other applications areas, patient data are physically and logically scattered into different information systems (public and private hospitals or clinics, general practitioners, pathology laboratories) and they may be required in order to make a diagnosis or a medical act. The privacy constraints of each system imply that it is very difficult to access to all information sources. It may exist a lot of information about the same person, but it is not physically gathered on the same spot or not reachable by all intervening parties [BP91]
- Each information system stores a lot of data about persons which are often similar (e.g. names, addresses, occupations, etc.) with many discrepancies. For example, in an information system, Mrs Smith is tagged as a medical student at the Laval University, whereas in another she is registered only as a student (because the speciality and the place are not relevant for that system). Databases are heterogeneous base on hardware (running on different computers), and also by their software (implementation, organization, human interface, command language). But they are also distinct by the kind of information they store, even if the information concerns the same person (or the same object)
- When you have to link information systems the above problems of heterogeneousness may become very difficult. The physical connection is not always possible due to the too important number of gateways between two sites, the communication difficulties and costs. The logical exchange of data may be shackled by different representations of them or different access languages [Bro93]. Finally, the data may be similar but not identical. In order to avoid data duplication or inconsistency of the end result, unification mechanisms have to be carried out. The unification concerns the data locations (in tables or files or others), the data types and also the data semantics (see the above example of the student Mrs Smith). Practical solutions are often based on a global view providing a general vision of all systems to be connected
- Information systems need to communicate and to interchange data for performing tasks and providing services based on multiple sources. Often it is not feasible

because the information systems do not provide the mechanisms for exchanging or sharing their data. Moving Trans-European goods is an example of that problem : at each border, customs officers have to control the provenance of the lorry, its rights for crossing the border, its conformity to laws and regulations of the country, its route, the nature of goods, etc. The European Community has launched a project for computerizing these control operations [Inca92]. But the main problem in such a project is a political one. Politicians of each country oppose to interconnect their information systems. More than the very hard technical problem due to the number of countries and the heterogeneousness of their systems, they fear that confidentiality and security of their system may be jeopardized

In this paper we propose a way to overcome some of these obstacles. The solution consists on a portable equipment that contains the data to interchange. These data generally concern a person or an object (vehicle, good) characterized by a relatively small quantity of information, storable in a small device and carried where they are required. Such a device is likely a portable and personal database server. It should be sufficiently small to take place in a pocket, and should assume security controls and management operations on its data. It is a way to bypass the difficulties of connecting different information systems. The data travel with the person or the object, then they are available any time as soon as an information system needs them. Moreover, the information is timely available to a mobile computer, there is no time lag due to connection and communication with a remote database. An onboard system can directly access to the data stored in the portable device. Data are stored in a single device, they have a unique representation; problems for connecting the database, sending request and unifying different data semantics are reduced to the knowledge of the portable data organization. The security is reinforced by the portable device itself, designed for resisting to any physical attack and offering high level access controls

Smart card is a technological device solution for storing and managing personal data in a single chip microcomputer with CPU, program and data memories. The card chip is embedded into a plastic card. Thanks to its own stand-alone computing capabilities, its internal security logic and its memory capacities, the smart cards are used efficiently by many applications. For example, the electronic purse application use the card like cash flow. Money is made available through the card and may be cash in for any transaction where a card reader is available. Clubs use the smart card to hold member information, club privileges, payment records, *etc*. Smart cards are also used as subscriber card for Pan-European digital cellular radio telephone network, with pay telephones, by healthcare facilities, as security key to access buildings and computers, *etc.*

It is a portable device which protects itself and manages access controls on the data it contains. Therefore, it is applicable as a portable data carrier, it provides a technology for identification support and is able to authenticate any data exchange during transactions. But this is not sufficient. Because smart cards contain sensitive

issuer's information and have to interact with many information systems, specific requirements should be implemented :

- The smart card should provide a way of sharing data among dif ferent information systems, and should maintain their privacy. A card users may only access to the card data for which he has rights. It is the question of sharing data and maintaining privacy
- Along its life cycle, the smart card structure should be able to interact with new users and new data. It is the question of card application flexibility
- Smart card has to be quickly and easily attached to different information systems whatever the computers, the operating systems and the applications. It is the question of smart card integration into the information systems

With these goals in mind, we propose a new smart card approach : the CQL card. In this card we have implemented and downsized the concepts of DBMS (Database Management System). A subset of SQL, CQL is used to communicate with the card and gain access to its data.

2 The CQL Card : A Technical Solution

The database card requires portability, intelligent functions, and confidential features. Smart cards offer these well-known features which have been successfully tested upon time. Smart cards are a convenient and highly secure medium for interaction with personal computer, communication devices, transactional terminals, and information systems in general. Nevertheless, smart cards still have difficulty in managing multiple application data interactions with other computer systems mainly traced down to their low level interface. In order to overcome these problems, a smart card based on database concepts, called CQL for Card Query Language has been developed with the following features.

2.1 Integrated Circuit Card (IC Card) and Portable Data file

The IC Card (usually called "smart card") results from the embedding of a single integrated circuit (a computer chip) into a plastic card (the size of a credit card) [Cri90]. The computer chip is programmed to perform specific functions such as updating and changing the information held on its memory. The key components of the chip architecture are as follows (*cf.* figure 1) :

- 8 bits microprocessor (*e.g.* 6805, 8051)
- input/output management module
- security block that assumes physical security and memory access controls
- ROM containing the operating system of the smart card (size from 2 to 16 KB)
- RAM used as a working memory by the microprocessor (size from 40 to 256 bytes)

- Non volatile memory like EEPROM (Electrically Erasable Programmable Read-Only Memory) used for storing the applicative data (size from 1 to 8 KB)

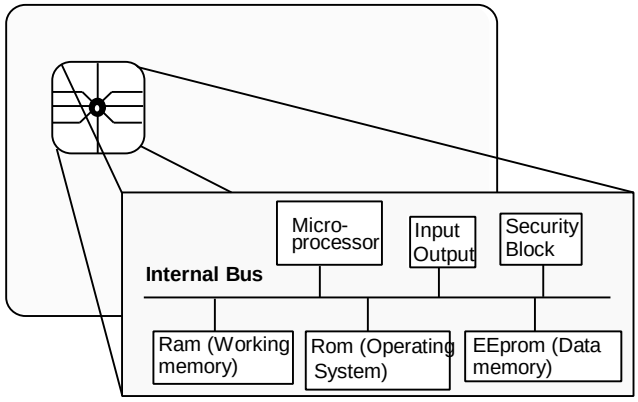

Fig. 1. Integrated circuit card architecture

Smart cards have to comply with the physical constraints defined by the ISO 7816 part 1 (physical characteristics) and part 2 (contact location) standards [Iso1] [Iso2]. The components of the chip have little dimensions, and the total chip area is about 25 mm^2; a larger chip might crack when the card is flexed. The main breakthrough we can expect is an increase in memory capacities (in the same dimensions), mainly with non volatile memory type Flash EEPROM (particular EEPROM memory that reduces the area of a memory cell by erasing a block of bytes rather than a single byte). Bigger memory capacities [Pat90] and new smart card microcontroller chips [Pey94] could be implemented for many applications, (for example student cards or patient cards) and could make possible the use of smart cards as multi-purpose portable and individual database servers.

Today, the operating systems for smart cards are generally based on file management systems. The privacy constraints for each card users imply that it is very difficult to access shared data stored in files. Many card users access to part of the data stored in the same file with different access right (read/write/update). But each access right enables accessing to all records of a file. If some particular records have to be protected they have to be stored in an another file. And so doing same data could be scattered in a multitude of file, resulting in difficulties for maintaining the information coherence. For solving this problem, we propose an operating system based on database concepts. We call a personal and portable database, a smart card used to gather information about the carrier in its database. Each information system may see the portable database as an adjunct source of data. By using DBMS mechanisms, each user, depending on its role, accesses to the data through specific views on the database. The authorized actions on the database are managed by the card. The location of the operating system (the database engine for us) inside the ROM memory of the card microcontroller is the one of the most important features of the smart card. Like that, the card insures its integrity

(physical and logical protection of the card memories) and guarantees the consistency and the privacy of its data. By controlling internally the accesses to the data, and by carrying out the database engine inside, the operating system gives to a user only what he can get. It is a fundamental feature of all smart cards. The CQL card improves it by providing the notion of views on parts of data structure.

2.2 Overview of the CQL Card

The CQL card is the result of a research effort being done by RD2P [GG92] (Recherche et Développement Dossier Portable), the CNRS (Centre National de la Recherche Scientifique), the Medical and Scientific Universities of Lille and Gemplus corporation [Gem92].

The CQL card is the first smart card that uses DBMS concepts. This sophisticated operating system has been implemented thanks to the emergence of powerful card chips and by the use of a high level development system dedicated to smart cards. This development tool developed at RD2P named *C_Card* is described in [GP91]. It enables developing card ROM codes with the C language rather than specific machine languages. It generates a pseudo-code executed by an interpreter which is also loaded into the card ROM. The interpreter takes up less than 2 KB in the ROM. The CQL pseudo-code has been optimized in size and speed in order to be loaded into the remaining 8 KB of the ROM.

The database engine manages “users” which access to the database “objects” (tables, views and dictionaries) according to their “privileges”. It is a relational database engine added to a secured users' management. The system manages all data and object structures. It uses its own tables for this purpose, in the same way as any other user. The system includes 3 tables :

- *table ** containing the description of tables, views and dictionaries
- *table *A* containing the description of users
- *table *G* containing the description of access rights (or privileges).

The applicative data and database structures are stored in the 8 KB of the card EEPROM. Data held in the card are accessed using CQL a high level Card Query Language [Par94] which is a subset of the standard SQL (Structured Query Language). The database engine carries out each CQL request as follows :

- receipt of a request
- syntactic analysis of the request
- user's access rights control
- execution of the request
- sending of a response

2.3 The CQL Language

The CQL language used to communicate with the card is described in the appendix. The bracketed numbers following in the text refer to the CQL commands described in this appendix. This subset of SQL has been chosen according to our needs of sharing data and maintaining privacy. The data are stored in tables. Views are logical and dynamic subsets of tables without data duplication. The owner of an object may grant privileges (select, insert, update or delete on table or view) to other users. Theses mechanisms enable sharing data and the card users' management guarantees the data privacy. It seems to us that it is a good solution for the specific problems of a portable data file. To resume, this subset enables us offering a consistent language to :

- manage the card users ([1] to [7])
- create database structures ([11] to [15])
- manipulate the database data ([18] to [25] and [29] to [34])
- grant and revoke privileges ([16] and [17])
- control the users' access rights ([3] and [8] to [10])
- and guarantee the database consistency ([26] to [28])

2.4 Users

The CQL card manages two types of users : the *Application Managers* (AM) and the *Standard Users* (SU), as well as two predefined users: the *Card Issuer* (CI) and the anonymous user called *PUBLIC*.

The Card Issuer and the user PUBLIC are predefined at the manufacturing level. They are unique and cannot be erased. The Card Issuer may create Application Managers and Standard Users. The Application Managers may create Standard Users. The user PUBLIC is the default user who does not have to present a password. The Standard Users and the user PUBLIC cannot create other users.

The users management is dynamic. The AM's and SU's can be created all along the card life cycle. The user's owner (the one who has created him) may delete him all along the card life cycle. There is no theoretical limit to the number of users. The card identifies a user by his name and his password, and monitors successive erroneous password presentations by a ratification counter. Once the ratification counter has reached its maximum value the user is locked and cannot anymore access to the database card.

2.5 Objects

The CQL card manages the following objects : tables, views and dictionaries. Objects may be created only by the Card Issuer and the Application Managers. The user who creates an object is the sole owner of this object with all the rights on it

including the right to delete it. Nevertheless, at any time, he may grant to other users access privileges to any object he owns (*cf.* paragraph 2.6).

The basic object managed by the CQL card is the *table*. A table is defined by its name, a number of columns and a name for each column. Data are stored and accessed by rows. Only one type of data is authorized : varying length character strings. The CQL Data Manipulation Language (DML) is a compatible subset of the SQL DML. It allows to read (`FETCH`), add (`INSERT`), remove (`DELETE`) or modify (`UPDATE`) a row. By using a predicate clause (`WHERE` clause) the CQL card is able to process logical request on a table. The `WHERE` clause defines a condition for logically selecting a set of rows.

With smart cards using file structures the security granularity is the file, whereas with CQL the security can be expressed at data element level through relational *views*. A view is a select clause definition referring only to one table and recorded in the card. It defines a dynamic and logical selection of rows and columns onto a table. By using the mechanism of view the data of a table can be shared among several users without data duplication. Only the tables contain data, thus, in a view, it is only possible to manipulate data for reading and sometimes updating. Because the CQL card does not manage keys on tables, it is important to provide a column with unique values for avoiding trouble during an updating operation. By the mechanism of views an Application Manager can give to another user a specific "vision" to his data.

The *Dictionaries* provide general information on the database objects structure (tables, views and dictionaries) and on users privileges. The dictionaries are only accessible for reading. They store information that enables using a CQL card without pre-existent knowledge of the card database structure. By this feature which enables exploring the card database structure it is possible to design open systems.

2.6 Access Privileges

The creator of an object is the only owner of this object. They are the only one able to delete it and update it by reading, adding, deleting and updating data into it. The creator may decide at any time to grant one or more of these privileges to any other user. These privileges cannot be passed on to other users and can be revoked by the owner only. These characteristics allow the owner of a table to grant privileges concerning a table or a part of it, a view, and thereby to control the actions carried out on the data it contains. The owner of a table can define specific access conditions for each view he has created. Like that he can reliably control the accesses to the data stored in that table.

2.7 Security Mechanisms

Application security is guaranteed by the card through three mechanisms which perform access control by double authentication, implement a tracing function and insure the database integrity.

In addition to the administrative connection functions (presentation of the name and password), the card performs a function of *double authentication*. It provides the host system and the card with a guarantee concerning the identity of each other. This process is based on pseudo-random number generation and the secret key algorithm DES [GQU92].

The *Tracing* function is assumed by the card system upon recognition of the special identifier USER as the name of the last column of a table. In this case, for each updating operation onto a row of the table, the card system places the current user's name in the column USER. The table may thus be accessed by several users providing each of them verify the name of the latest one who has updated a particular row.

In many DBMS's there are mechanisms that assume the *database integrity i.e.* that assume transactions should leave the database in a consistent state. In the CQL card we have implemented the mechanism of one phase commit. The commit provides that all the modifications required since the beginning of the transaction are fully executed. The rollback is the command that restores the initial context before the beginning of the transaction. If the card is disconnected during a transaction, the next connection will be started by an automatic rollback.

3 Database Card Integration into the Information Systems

The database card can be seen as a link among many information systems which cannot or do not wish to communicate. It carries data of different systems offering interchanges of information and computing it in the same way. Therefore, the challenge for this database card is to be interoperable with many systems and to communicate with them under a uniform interface. Adopting a subset of SQL for its set of commands, the IC Card becomes accessible via a well-known language. However the problem of data-interchange among various databases and applications remains[Bro93]. In regards to this problem, a group of leaders in the database and systems industry, the SQL Access Group (SAG), has defined a Call Level Interface (CLI) which provide a common interface for accessing to heterogeneous databases. Based on this CLI, Microsoft proposes specifications of an API (Application Programming Interface) called ODBC (Open Database Connectivity) [Odbc92]. We have implemented an ODBC driver for the CQL card which enables using it with many vendor software and helping to solve the problem of the connectivity. This development has been evolved with the following guidelines :

- *Embrace Standard* : The CQL card complies with smart card standards ISO 7816 parts 1, 2 and 3 for physical characteristics, contacts location and transmission

protocols. However the CQL card embraces DBMS standard for its set of commands, it uses a subset of SQL [Iso5], called CQL for data interchange

- *Top-Down Approach* : Considering the smart card ISO 7816 part 4 Draft standard on Inter-Industry Commands for Interchange [Iso4], using SQL as a smart card language may seem unusual. We claim that such language could ease the integration of smart cards into information systems. The use of computer standards rather than specific smart card standards favours the integration of smart cards and provides interoperability between smart cards and systems
- *Implement Middleware* : Formally, interoperability means the capacity for two components to exchange requests and responses based on a mutual understanding of messages [Bro93]. One way for providing a mutual understanding is to implement an intermediate software layer that encodes requests and responses into a uniform code. This intermediate software layer, often called *middleware*, has to be sufficiently generic in order to allow a wide range of applications to interact

The CQL ODBC driver enables using the CQL card through a wide range of market products. Like that, the use of the CQL card through these products simply requires a general knowledge of these products and of the smart cards which become very easy to use and represent real "plug and play" systems [PV94a]. An application using ODBC call level interface to access to data stored in databases is called an ODBC-enabled application. We can run an ODBC-enabled application with a CQL card via the CQL ODBC driver or with another data source via its corresponding driver. From CQL cards up to large databases the same code applies. Such interoperability enhances the integration of smart cards into information systems. It is particularly relevant for the portable data files area of application in which smart cards (*e.g.* health care cards) have to interchange data with a number of various systems (*e.g.* hospital information system, general practitioner PC, emergency portable computer).

3.1 ODBC Technology

ODBC addresses the heterogeneous database connectivity problem using the common interface approach (*cf.* figure 2).

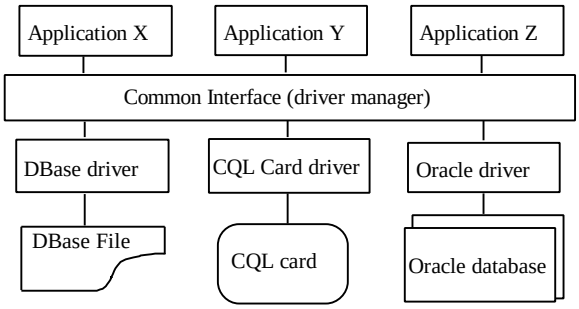

Fig. 2. Applications accessing heterogeneous data sources via a common interface approach

The ODBC interface allows applications to access data from a wide range of DBMS's. Each application uses the same code to communicate with many data sources through DBMS-specific drivers. Therefore these applications are independent of the DBMS's. A Driver Manager sits between applications and drivers providing a mutual understanding of the exchanged messages and a common interface for all applications.

The ODBC API functions are divided into three levels of implementation called respectively Core, Level 1 and Level 2. The first one is strictly an implementation of the CLI developed by the SAG which can be seen as the lowest-common-denominator. The second puts forward new functions for gaining information about the driver and the data source and for setting and retrieving the driver options. The latest allows ODBC to manage sophisticated databases capacities. An ODBC driver has to implement at least the functions of the Core and can process any part of the SQL grammar defined by ODBC. An application accesses a data source by function calls of one or more levels and submits SQL requests depending on the driver conformance. Application developers must decide whether to use the minimum level of functionality, or write the conditional code to test for extended functionalities. In either case, the Core level guarantees a working state.

3.2 Implementation of the CQL ODBC Driver

The CQL ODBC driver is a Windows Dynamic Link Library (DLL). To submit a SQL request to the CQL card and receive results, the driver calls the CQL DLL which sends the statements to the Card Reader Driver. This later routes the statements to the card and sends back the card responses.

The CQL ODBC driver carries out all Core and Level 1 functions and processes the SQL DML part - *i.e.* SELECT, INSERT, UPDATE and DELETE - defined by ODBC. The diver translates each SQL statement to the corresponding CQL statement. The SQL part that it is not supported by the CQL language (*e.g.* `<ordered by>` in a SELECT clause) is ignored although the statement is processed. The Level 1 of implementation has been reached using the dictionaries of the CQL Card which allow the driver to obtain information about the data source

and the data source's system tables. The SQL grammar used by the driver has been limited to the DML because we restrict this driver for the use of structured cards and not for the creation of CQL database structures.

4 Conclusion

In the smart card area, the emergence of applications includes more and more intervening parties pointing out new needs. The file management techniques implemented in the first smart cards do not meet these demands. The database management systems were a technological gap for the information systems. The same gap is necessary for smart cards. We propose a smart card with a downsized DBMS. Each card user, depending on its role, has specific views and privileges onto the data stored in tables. The CQL card is a computer element that deals with information systems and it also must be integrated into these systems. For this reason we have implemented some standards of the DBMS's such as SQL and ODBC.

The CQL card as a database server can be connected to and used from multiple information systems, then the card is becoming an active link between these systems. The card is a device that offers solutions for the problem of interoperability between DBMS's that cannot or do not want cooperate. With the smart card, and thanks to its small size, each person may carry in his back pocket data about himself such as health diagnostics, access conditions to protected buildings, list of debit and credit transactions, etc. There is no question that the application area of smart cards will increase significantly. Thus, the design of information systems must deal with this new support of data. The challenge is to converge the two points of view; the first one concerning the global information system, and the second one the personal and mobile database server.

Acknowledgements

We gratefully acknowledge Georges Grimonprez and Edouard Gordons of RD2P and Gemplus for supporting our work and for their previous work on design and implementation of the CQL card. We would like also to thank Emmanuel Horckmans for his participation in the development of the CQL ODBC Driver without which this result would not have been possible. We are grateful to Andre Gamache (Laval University) for helpful suggestions and comments on earlier and final drafts of this paper.

References

[BP91] R. Beuscart and P. Paradinas. *Smart Cards for Health Care, in* Telematics in Medicine, Elsevier Science Publishers B.V., North-Holland, 1991.

[Bro93] M.L. Brodie. *The promise of distributed computing and the challenges of legacy information systems, in* IFIP Transactions A-25, Interoperable Database Systems (DS-5), Elsevier Science Publishers B.V., North-Holland, 1993.

[Cri90] J. Mc Crindle. *Smart cards*, IFS Publications, Springer-Verlag, 1990.

[Gem92] Gemplus. *CQL Card and Language Reference Manual*, Gemplus, 1992.

[GG92] E. Gordons and G. Grimonprez. *A card as element of a distributed database*, IFIP WG 8.4 Workshop, P. Paradinas and G. White : The portable office. Microprocessor cards as elements of distributed offices, Ottawa, Canada, 1992.

[GP91] G. Grimonprez, and P. Paradinas. *A new approach in code development : C_Card and Cossack, in* proceedings of CardTech'91, Washington D.C., U.S.A., 1991.

[GQU92] L. Guillou, J-J. Quisquater, and M. Ugon. *The Smart Card : A standardised Security Device Dedicated to Public Cryptology*, Ed G. Simmons : Contemporary Cryptology, IEEE-Press, 1992.

[Inca92] European Nervous System (ENS). *The INCA project (Information network and card for the adaptated management of European road transport)*, number E20003, EEC Documentation, 1992.

[Iso1] ISO/IEC 7816-1. *Identification cards - Integrated circuit(s) cards with contacts : Dimensions and locations of the contacts*, ISO, 1987.

[Iso2] ISO/IEC 7816-2. *Identification cards - Integrated circuit(s) cards with contacts : Physical characteristics*, ISO, 1988.

[Iso3] ISO/IEC 7816-3. *Identification cards - Integrated circuit(s) cards with contacts : Electronic signals and transmission protocols*, ISO, 1989.

[Iso4] ISO/IEC 7816-4. *Identification cards - Integrated circuit(s) cards with contacts : Interindustry commands for interchange (CD)*, ISO, 1992.

[Iso5] ISO/IEC 9075. *Information Technology - Database - SQL*, ISO, 1992.

[Odbc92] Microsoft Corporation. *Microsoft Open Database Connectivity Backgrounder*, Microsoft, october 1992.

[Par94] P. Paradinas. *The CQL Database Smart Card*, GMD, Smart Card Workshop, Darmstadt, Germany, February 1994.

[Pat90] M. Paterson. *"Memories are made of this..." ...a look at memory considerations for Smart Card applications*, Semiconductor engineering bulletin, Motorola Ltd, 1990.

[Pey94] P. Peyret. *RISC-Based, Next-Generation Smart Card Microcontroller Chips, in* proceedings of CardTech'94, Washington D.C., U.S.A., April 1994.

[PV94a] P. Paradinas and J.J. Vandewalle. *How to integrate Smart Cards in Standard Software without writing specific code?, in* proceedings of CardTech'94, Washington D.C., U.S.A., April 1994.

Appendix : The CQL Language

Administrative Command Class

[1] `CREATE APPLICATION <application name> <ratification thresold> <password>;` *`to create an Application Manager`*

[2] `CREATE USER <user name> <ratification thresold> <password>;` *`to create a Standard User`*

[3] PRESENT <user name> <password>; *to open a session with the card*

[4] CHANGE PASSWORD <previous password> <new password>; *to allow users to modify their password*

[5] UNLOCK <user name>; *to allow the owner of a user to unlock them when its ratification counter has reached the maximum*

[6] DELETE USER <user name>; *to delete an AM or a SU*

[7] STATUS; *to read the card serial number and the number of bytes used in the user memory*

[8] CREATE KEY <logical key name> <user name> <key value>; *to create or erase a key*

[9] AUTHENTICATE <user name> <logical key name> <host random value>; *to initialize a double authentication process*

[10] CHECK <user name> <encrypted password>; *to make an encrypted presentation of the password*

Structure Definition Command Class

[11] CREATE TABLE <table name> (<column name> [,<column name>,...]); *to create a Table*

[12] CREATE VIEW <view name> AS <select clause>; *to create a View*

[13] CREATE DICTIONARY <generic name of dictionaries>; *to create the tables, users and privileges Dictionnaries*

[14] DROP TABLE <table name>; *to drop a Table*

[15] DROP VIEW <view or dictionary name>; *to drop a View or a Dictionary*

[16] GRANT <list of privileges> ON <table, view or dictionary name> TO <user name>; *to grant a privilege on a object to a user*

[17] REVOKE <privilege> ON <table, view or dictionary name> TO <user name>; *to revoke a privilege granted on a object to a user*

Date Handling Command Class

[18] DECLARE CURSOR FOR <select clause>; *to create a cursor pointing onto a logical selection of rows*

[19] OPEN; *to position the cursor onto the first row which satisfies the logical selection*

[20] NEXT; *to position the cursor onto the next row which satisfies the logical selection*

[21] FETCH; *to prepare the data units pointed by the cursor to be sent back by the system after a READ RECORD command*

[22] FETCH_NEXT; *FETCH and NEXT command*

[23] INSERT INTO <table name> VALUES (<string> [,<string>,...]); *to insert a new row*

[24]ERASE ; *to delete the row pointed by the cursor*

[25]UPDATEC SET <column name> = <string> [,<column name> = <string>,...]; *to update the row pointed by the cursor*

[26]BEGIN TRANSACTION; *to initialize a transaction*

[27]COMMIT; *to validate all transaction modifications*

[28]ROLLBACK; *to restore the initial state of memory prior to the beginnig of the transaction*

[29]READ RECORD <length requested>; *to read the data prepared by the FETCH and FETCH_NEXT commands*

Select Clause

[30]<select clause> **::=** SELECT <column selection> FROM <table or view name> WHERE <where clause>;
[31]<column selection> **::=** * **|** <column name> [,<column name>,...]
[32]<where clause> **::=** <predicate> [AND <predicate> AND ...]
[33]<predicate> **::=** <column name> <comparison operator> <string>
[34]<comparion operator> **::=** < **|** <= **|** > **|** >= **|** = **|** <>