How to integrate Smart Cards in Standard Software

without writing specific code ?

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

(1) RD2P
Recherche et Développement
Dossier Portable
CHRU Calmette - R. Pr. Leclerc
59037 Lille Cédex - France
Tel. : (33) 20 44 60 44
Fax : (33) 20 44 60 45
Email : pierre@rd2p.lifl.fr
jeanjac@rd2p.lifl.fr

(2) UNIVERSITE LAVAL
Faculté des Sciences
Dpt 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 - Plaine de Jouques
13 881 Gémenos - France
Tel. : (33) 42 32 50 00
Fax : (33) 42 32 50 90
Email : pierre@gemplus.fr

Abstract.

When you add a CD-ROM reader to your computer, this new device becomes integrated in part of your environment. The CD-ROM performs like a floppy disk. When using a spreadsheet and a word-processing, if you *cut* some cells in the first application you can *paste* these cells in the second application.
These two services are provided by application software and operating systems (Windows for PC or System 7 for Apple Computer), but the end user does know nothing about the complex implementation of these services. With smart card it is not the same thing; you have to add an application software and a reader to your system. So, in many case this application is not really opened, it cannot be operated with other software. The main challenges for smart cards as for computer systems is to be opened.
Today the SAG (SQL Access Group composed of major manufacturers, software providers and end users) has defined an open access method to data stored in many system like databases. This approach gives a unique CLI language (Call Level Interface) for applications that needs to make data requests. The first implementation of this recommendation is ODBC (Open DataBase Connectivity) from Microsoft. It is endorsed by a large panel of manufacturers and software providers.
We propose to extend this approach to smart cards and software tools like drivers. In this case, all software products based on these standards can easily integrate smarts cards and their software tools without writing a specific program !
CQL a database server in a smart card that use CQL (Card Query Language) a subset of SQL-Standard (Structured Query Language) and this CLI packaging are completely usable from application software. Database concepts are included in CQL-Card (tables, views, dictionaries, access privileges in select/insert/update/delete) and there are the same that in ODBC.

Keywords. Interoperability, CQL card, SQL, Open Database Connectivity.

1. THE CHALLENGES OF INTEROPERABILITY AND "PLUG AND PLAY" SYSTEM

The main trends of information processing are going to provide application interoperability and plug and play system. Smart cards are concerned. We show in this section how smart cards can be plugged it in a Personal Computer (PC) and why their software integration is nowadays insufficient. In the next section we propose an example that shows a way for integrating smart cards in standard software.

1.1. Plug and Play hardware

The widespread PCMCIA standard (PC Memory Card International Association) and its extensions that provide more memory storage and connection facilities are examples of this evolution. In the same way, but more in term of software, activities around object oriented technologies try to achieve the same target. The application interoperability is based on three objectives :

- Services providers (servers) and user of services (clients) should communicate without having knowledge of the implementation of each other.
- Old information system components should continue to operate in these architectures and should be accessed like new ones : it's the problem of the legacy information system migration [BRO93].
- The application development should be independent of hardware constraints.

These trends result from needs for applications to have open and standard systems. This new approach in information systems is more and more required by many users. So, application designers can't ignore these requirements. In this context, smart card technologies must take these two challenges into account. Thus, to succeed, the smart card should be available in most of the existent systems in terms of hardware and software.

The GPR (Gemplus Pocket Reader) [PEY94], a thin smart card interface built in a PC Card format (PCMCIA), is the key to the connection of smart cards to many notebook, Personal Digital Assistant (PDA), or pen-based computers which are now fitted with one or more PCMCIA slots. With this device, all equipment fitted with PCMCIA slots becomes a smart card terminal (*cf.* figure 1). Moreover, containing its own application program, the pocket reader meets the concept of Plug in and Play.

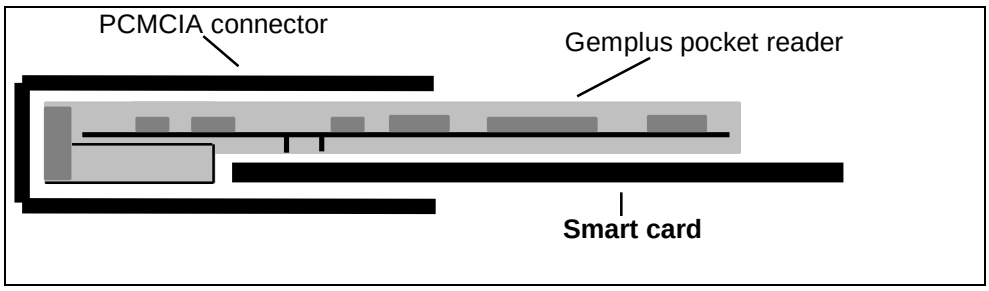

Figure 1. *Smart card is held on the GPR back side*

Thus the plug and play technology for accessing smart cards is reached : GPR lets PC hardware and smart cards work together automatically.

1.2. The challenge of smart cards integration in the information systems

The second part of our challenge is the software integration. Before exposing our proposition, we propose a brief overview of smart card environment and information processing.

In PC environment, Windows is a de facto standard. The Windows strategy to connect front-end applications to various back-end services is based on WOSA (Windows Open Services Architecture), a single, system-level interface to define a uniform computing environment. WOSA enables Windows applications to connect to all the services across multiple computing environment by making a set of common APIs (Application Programming Interfaces) available to all applications. For our problem, three components of WOSA seem to be relevant :

- ODBC, Open DataBase Connectivity, API addresses database connectivity technology for data access or retrieval (see below).
- TAPI, Telecommunication API for telecommunication applications where smart cards are concerned (phone cards, Subscriber Identification Module cards for Global System for Mobile communications or GSM, etc.).
- EFS, Extension for Financial Services addresses payment and banking environment where smart cards are well known by smart credit card or electronic purse applications.

On the other hand, it is interesting to note the emergence of open system specifications like DCE (Distributed Computing Environment) from OSF (Open Software Foundation) or CORBA (Common Object Request Broker Architecture) from OMG (Object Management Group). These new trends are present in Windows by OLE (Object Linking and Embedding).

For the smart card point of view, the question is complex : the features of the smart cards are limited but its microcomputer can provide high level functionalities. Seeing the smart card as a secured data source is interesting, the database management system concepts may be apply at an individual level . In the smart card area, there exists different approaches list below :

- DOSCARD is a smart card and a specific driver that enable applications to use this smart card like a floppy disk.
- Virtual card or CAPI (Card API) [LL93] is a way for defining a common software interface to provide product with interoperability between smart cards from different manufacturers.

Theses approaches provide interoperability : a floppy disk is recognized by the operating system, so by all applications, and a virtual smart card is an interoperable smart card based on a lowest common denominator among features of different smart cards. But the first one reduces the set of functionalities of the smart card to a simple file management, and the second one provides a smart card-dependent API based on a lcd, so that reduces smart card functionalities and that not embrace current computer APIs. From spreadsheet, from Database Management System (DBMS) or from any current software application, if you try to have access to smart card data, we can suppose they are in files, so for processing them we need to :

- know the pathname of the file

- know the data structure of the file (*e.g.* linear fixed, cyclic or transparent)
- have a software that assumes these two previously functions and the format conversions

We bring to give more latitude for solving the integration of the smart card in current software applications. Having learnt the lesson of the above proposals we try to give an answer to how attach a card at a high level interface. Our objectives are to simplify the access to the data stored in a smart card, and to make the card as a current component of information systems. Our approach is based on the SQL Access Group (SAG) specification, the implementation of a computer standard API, so for the smart card area it's a top-down approach.

2. AN EXAMPLE : THE CQL ODBC DRIVER

In order to illustrate ideas discussed above we present in this section a card that can be used by many off the shelf software without writing specific code for this card. The configuration required to link this card to a Personal Computer (PC) is a standard card reader and the CQL ODBC driver we have developed. This development is based on previously stated ideas for reaching interoperability i.e. :

- **Embrace Standard** : The CQL card is a card conforms to smart card standards ISO 7816 parts 1, 2 and 3 for physical characteristics, contacts location and transmission protocols. However the CQL card embraces computer standard for its set of commands, it uses a subset of SQL [ISO9075], we called CQL, for data interchange.
- **Top-Down Approach :** Considering the smart card ISO 7816 part 4 Draft standard on Inter-Industry Commands for Interchange, using SQL (standardized request language for database) as a smart card language may seem unusual in this field. We claim that such language may ease the integration of smart cards into information systems (see other reasons below). Use of computer standards rather than specific smart card standards favors integration of smart cards and provides interoperability between smart card and systems.
- **Implement Middleware** : Formally, interoperability means the capacity for two components to exchange requests and answers based on a mutual understanding of messages [BRO93]. One of these means of providing this mutual understanding of exchanged messages between the two components (like a smart card and an information system in our case) is to implement an intermediate software layer that encodes request and answer into a uniform code understandable by each other. This intermediate software layer often called middleware has to be sufficiently generic in order to allow a wide range of applications to interact. For the CQL card we have implemented an ODBC driver in conformity with the Microsoft ODBC specifications [ODBC93]. ODBC is the first implementation of the SQL Access Group (SAG) Call Level Interface (CLI) specifications for accessing to databases in heterogeneous environment.

So, thanks to this consistency approach, we have a portable Windows software (ODBC is a part of the standard Windows and Windows NT operating systems) that allows everyone to use the CQL card through a wide range of market products like Microsoft Access, Visual Basic or, Computer Associates Q by X. The use of the CQL card through these products just requires knowledge of these products and very little knowledge of smart cards which become very easy to use and represents a real "plug and play" system and is fully promoted by Microsoft.

An application using ODBC call level interface to access to information stored in a database is called an ODBC-enabled application. So we can write an ODBC-enabled application and perform it with a CQL card via the CQL ODBC driver or with a dBase file via the dBase ODBC driver or with an Oracle Database via the Oracle ODBC driver! From the CQL card up to a large Oracle database the same code applies. Such interoperability [SCH93] enhances integration of smart cards into information systems. It's particularly relevant to portable data files application area where smart cards (*e.g.* health care card) have to exchange data with a number of various systems (*e.g.* hospital information system, general practitioner PC, emergency portable computer).

In summary, the CQL ODBC driver offers the following benefits :

- It allows users to operate CQL cards from their current application; that's permits now for developers more quickly handle and evaluate the card by running very simple functions like "display", "query" or "report".
- It allows software developers to quickly test, validate and evaluate a CQL card application without real cards by using, for example, dBase files in order to simulate cards. Once the application has been validated, all the code written for dBase files remains directly usable for CQL cards.
- It simplifies card application development. Just knowing SQL (so CQL) a user can develop a CQL card application. Requests and responses structures are defined by the ODBC call level interface and are independents of the card.
- It allows corporations to protect their investments in CQL card. Today ODBC is a portable Application Program Interface (API) enabling the same data access technology across both Windows and Apple Macintosh platforms [ODBC92]. ODBC insulates applications from modifications to networking transports, servers, and from future smart card technologies.

The following discussion will examine underlying concepts and technologies of the CQL ODBC driver :

- CQL smart card characteristics
- ODBC description
- CQL ODBC implementation, installation and utilization

2.1. The CQL card

The Gemplus CQL card [GEM1] [GEM2] [GEM3] is the result of research effort done by RD2P [GG92] [GRI92] (Recherche et Développement Dossier Portable), CNRS (National Research Center), Medical and Scientific Universities of Lille and Gemplus corporation.

The CQL card is the first smart card in the world to use Database Management Systems (DBMS) concepts. The CQL card is a portable database for health applications or government documents where data are required by many different partners [PAR93]. All the information required by these various partners can be shared without duplication and with full security measures.

Database engine is carried out by the card. It manages “users” entities which handle different “objects” according to their “privileges”. Information is stored in tables, and views

are logical and dynamic subsets of tables without duplication. An owner of an object can grant privileges (select, insert, update and delete on table or view) to other partners. Furthermore dictionaries are specific views containing card description, they can be read to allow computer environments to vary in accordance with card structures.

Information held in the CQL card are accessed using CQL (Card Query Language) a high level database query language which is a subset of the market standard SQL (Structured Query Language). CQL card can be easily integrated into Information Systems using standard database access tools and complete software packages environment for application developments. From smart card up to mainframe concepts are identical : databases and SQL/CQL.

2.1.1. Partner agent within CQL card

CQL card manages four types of partners with the following profiles :

- Card Issuer (CI)
- Application Manager (AM)
- Standard User (SU)
- anonymous user called PUBLIC

At the manufacturing level, the card contains the Card Issuer and PUBLIC. They are unique and can't be erased. The Card Issuer can create Application Managers and Standard Users. Application Managers can create Standard Users. PUBLIC is the default user who doesn't have to present a PIN code. Standard Users and PUBLIC can't create others partners.

The users management is dynamic and flexible. Partners AM and SU can be created all along the card life cycle. The owner of a partner (the one who created him) can delete him all along the card life cycle. Excepting memory size, there's no limit to the number of partners. The card identifies a user by his name and his password, and monitors successive erroneous password presentations by a ratification counter.

The user management in the CQL card is implemented with the following statements :

- `PRESENT` *to present card with a user name and a password*
- `CHANGE PASSWORD` *to allow users to modify their password*
- `UNLOCK` *to allow the owner of a user to unlock him when his ratification counter has reached the maximum*
- `CREATE APPLICATION` *to create a user with the Application Manager profile*
- `CREATE USER` *to create a user with the Standard User profile*
- `DELETE USER` *to delete an AM or a SU*

2.1.2. CQL Objects

The CQL card manages following objects : tables, views and dictionaries.

Objects can be created only by the Card Issuer and Application Managers, so Standard Users and PUBLIC can't create objects. The partner who creates an object is the single owner of this object, he exclusively owns all the rights including the one to delete it. Nevertheless, he may at any time grant to other partners access privileges to any object he owns(*cf.* section 2.1.3).

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 in the horizontal part of the table, the rows. Rows are composed with one or more columns. Only one type of data is authorized : varying length character strings.

MEDCARE	**DATE**	**Diagnostics**	**Therapy**
	92.12.05	bronchitis	antibiotic
	92.12.18	hypertension	clonidine
	93.01.23	bronchitis	antibiotic
	93.02.01	unstable angina	trinitrine

```
SELECT diagnostics FROM medcare
WHERE date > '92.12.31' AND date <
'94.01.01';
```

Figure 2. *A sample CQL table for health care application and an example of a CQL command*

The CQL Data Manipulation Language (DML) is a compatible subset of the SQL DML. It permits to read (`FETCH`), add (`INSERT`), remove (`DELETE`) or modify (`UPDATE`) a row or several rows in a table. By using a predicate clause, introduced by the keyword `WHERE`, the CQL card is able to process logical request on a table. The `WHERE` clause defines a condition satisfied by rows selected. The CQL search condition is a combination of predicates separated with the logical connective `AND`. An expression consisting in a search condition is `<column name> <comparison operator> <constant>` where the CQL comparison operator is one of the six usual comparison operators (=, <>, <, <=, >, >=). The select can be expressed by this form : `SELECT <commacolumnlist> FROM <table name> WHERE <search condition>`. It gives a logical condition to selected information in a table. With others smart cards, when you use file structure the security granularity is the file or a record, whereas with CQL the security can be expressed at data element level.

The security management of the card uses the object of a **view** on a table. A view is a select clause definition referring to a table and recorded in a special table stored among others in the card. It defines a dynamic and logical selection of rows and columns of a table. By using the view definition, there's no need to duplicate part of data to another tables, avoiding information redundancy. Only tables content information, so it's only possible to manipulate data in a view for reading and updating. With views an Application Manager can give to another partner specific “visions” of his data.

Dictionaries provide general information on the database objects structure (tables, views and dictionaries) and on users privileges granted by the owner of objects. It's possible to access dictionary data for reading only. With this information, available by CQL request, it is possible to use a card without information about it. With this feature and software available to explore structure it is possible to design open systems. A card and its structure are initially unknown but can be used! It is a media transparency mechanism.

These functions have been implemented the following statements from the SQL standard [ISO9075] :

- `CREATE TABLE` *(respectively* `VIEW`*,* `DICTIONARY`*) to create a table (respectively a view, a dictionary)*
- `DROP TABLE` *(respectively* `VIEW`*) to delete a view (respectively a view)*
- `INSERT` *to insert a new row*
- `DELETE` *to delete a row*
- `UPDATE` *to update a row*
- `DECLARE CURSOR` *to define a selection clause*
- `FETCH` *to read a row,* `NEXT` *to read the following*

2.1.3. Access privileges

The creator of an object (table, view or dictionary) is the only owner of this object. He is the only one able to delete this object and to read, add, delete and update data into it. The creator can decide at any time to grant one or more of these privileges to other partners : CI, AM, SU or PUBLIC. These privileges can't be passed on to other users and can be revoked by owner. These characteristics allow the owner of a table to grant privileges concerning this table or a part of it (a view) and thereby control actions carried out on data it contains.

These functions have been implemented by the SQL standard grant and revoke commands :

- `GRANT` *to grant a privilege on a object to a card partner*
- `REVOKE` *to revoke a privilege granted on a object to a card partner*

Privileges granted to users, offer the possibility to handle data through a view as if they were really recorded. An owner of a table can define specific access conditions to each specific view he's created. Like that he can smartly control accesses to its information stored in that table. This two functions can be activated during all the card life cycle by authorized user.

2.1.4. Security mechanisms

Applications security is guaranteed by the CQL card by mean of three functions that performs :

- access control by double authentication
- tracing

- database integrity

In addition to the administrative connection functions (name and password presentation), the card makes use of sophisticated functions to perform **double authentication**. This double authentication provides the host system and the card with a guarantee as to the right identity. This process is based on pseudo-random number generation and on the secret key algorithm DES [GQU92].

Tracing is assumed by the card system when it recognizes the special identifier `USER` as the name of the last column of a table when it's created. In this case, for each access to the table, the card system places the name of the current user in this column. The table can thus be accessed by several users but each of them can verify the name of the latest who's accessed to a particular row.

In many DBMS 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 disconnecting during a transaction, the next connection will be started by an automatic rollback.

These functions have been implemented with the following statements :

- `CREATE KEY` *to create or erase a key*
- `AUTHENTICATE` *to initialize a double authentication process*
- `CHECK` *to make an encrypted presentation of the password*
- `BEGIN TRANSACTION` *to initialize a transaction*
- `COMMIT` *to validate all transaction modifications*
- `ROLLBACK` *to restore the initial state of memory prior to the beginning of a transaction*

2.1.5. Technical characteristics

The number of users and objects is only limited by the memory space. Today application memory space of database card is 8 KB. The CQL language, like SQL, is independent of memory implementation. It implies the possibility to design cards with less or more memory without changing the host software application. When data are deleted or structure dropped memory is not lost and space is reclaimable by the CQL core. The card communication protocol is fully complying with ISO standard 7816-3 with T=0.

2.2. The CQL ODBC Driver

In this section we present the basic philosophy of ODBC, and then we explain how we have implemented the CQL ODBC driver and what are the choices we have made. Finally and briefly we show how to install and use this driver. Today this driver is available from both authors.

2.2.1. What is ODBC?

ODBC is a component of Microsoft Windows Open Services Architecture (WOSA). ODBC addresses the heterogeneous database connectivity problem using the common interface approach (*cf.* figure 3). ODBC defined Application Program Interface (API) to access all data sources. This API is based on a Call Level Interface (CLI) specification, which was developed by a consortium of over 40 companies members of the SQL Access Group (SAG) [ODBC 92].

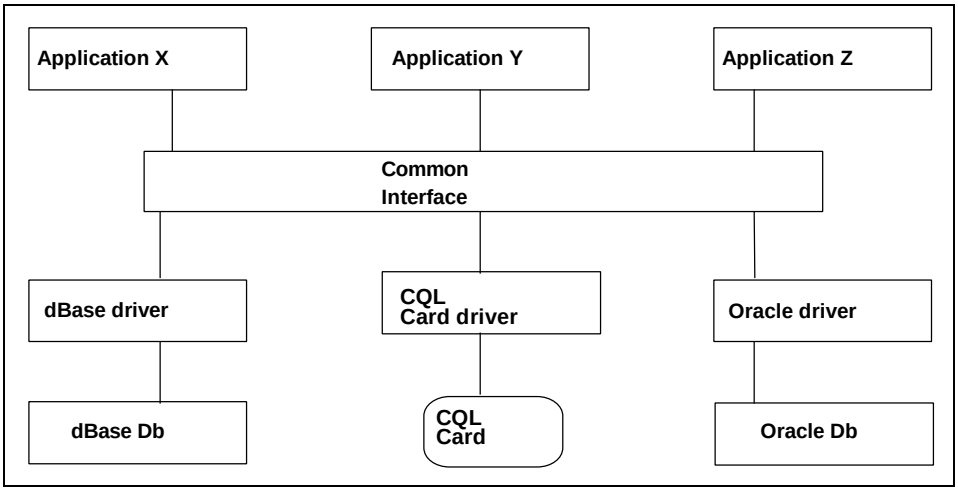

Figure 3. *Applications accessing heterogeneous data sources via a common interface approach.*

The ODBC interface, a single API, allows applications to access data from a variety of database management systems (DBMS). Each application uses the same code, as defined by the API specification, to talk to many types of data sources through DBMS-specific drivers. So these applications are independent of the DBMS's from which they access data. A Driver Manager sits between applications and drivers. In Windows, the Driver Manager and the drivers are implemented as DLLs. The ODBC Driver Manager is a part of the standard Windows and Windows NT operating systems.

Figure 4 shows how ODBC works :

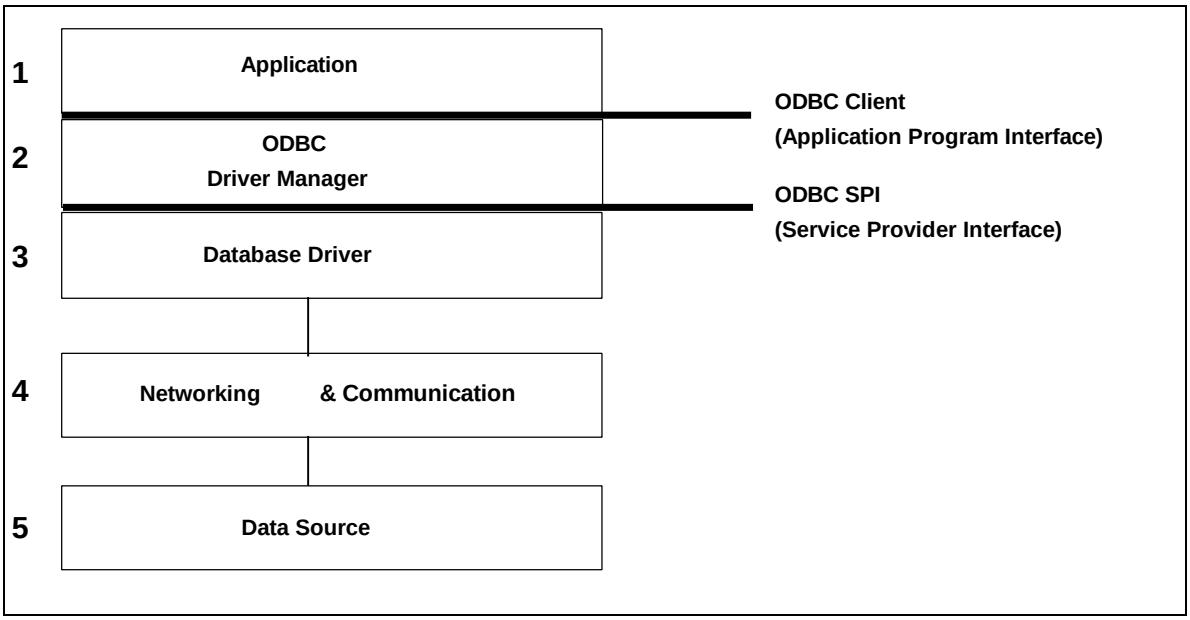

Figure 4. *ODBC Architecture*

1 The application call ODBC API functions to connect to a data source, submit SQL statements and retrieve results, and disconnect.

2 The driver manager loads the ODBC Driver dynamically for the application, passes on requests to driver and sendbacks results to application.

3 The driver processes ODBC function calls, translates the SQL grammar defined by ODBC to the grammar required by the target data source, submits SQL requests to the data source and returns results to the application.

4 The networking software layer carries request and results over the network.

5 The data source processes requests from driver and returns results to driver.

ODBC API functions are divided into 3 levels of implementation called respectively Core, Level 1 and Level 2. The first one is a strictly implementation of the CLI developed by the SAG, it can be seen as the lowest-common-denominator. The second proposes new functions for obtaining information about driver and data source and for setting and retrieving driver options. The latest allows ODBC to manage sophisticated databases capacities. An ODBC driver has to carry out at least functions of the Core and can process any part of the SQL grammar defined by ODBC. In summary the ODBC API functions performs following tasks:

ODBC API functions	Levels		
	Core	1	2
Connecting to a Data Source	X	X	X
Obtaining Information about a Driver and Data Source		X	X
Setting and Retrieving Driver Options		X	
Preparing SQL Requests	X		X
Submitting Requests	X	X	X
Retrieving Results and Information about Results	X	X	X
Obtaining Information about the Data Source's System Tables		X	X
Terminating a Statement	X		
Terminating a Connection	X		

Figure 5. *Implementation levels of ODBC API functions*

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 have to decide either use the minimum level of functionality, or write the conditional code to test for extended functionality. The Core level guarantees a working state in either case.

2.2.2. Implementation of the CQL ODBC driver

To implement the CQL ODBC driver we have realize a Windows dynamic linked library (DLL) using the Microsoft ODBC Software Development Kit (SDK) - "set of software components and tools designed to help in developing ODBC drivers and ODBC-enabled applications" [ODBC93]. The CQL ODBC driver consists of two DLLs:

- CQLDRVSP.DLL : CQL driver setup program for specific installation and maintenance requirements of the CQL ODBC driver. This program is loaded when installing the CQL ODBC driver via the ODBC Setup and Administration utilities.
- CQLDRV.DLL : it is the driver for CQL cards. It processes ODBC function calls, translates the SQL grammar defined by ODBC to the CQL grammar, submits SQL requests to the CQL card and returns results to the application.

For submitting a SQL request to the CQL card and receive results, the driver calls the CQL DLL which sends to the Card Reader Driver the statements. This later routed statements to and from the card inside the Card Reader. Figure 6 summarizes the application-to-CQL card architecture using the CQL ODBC driver.

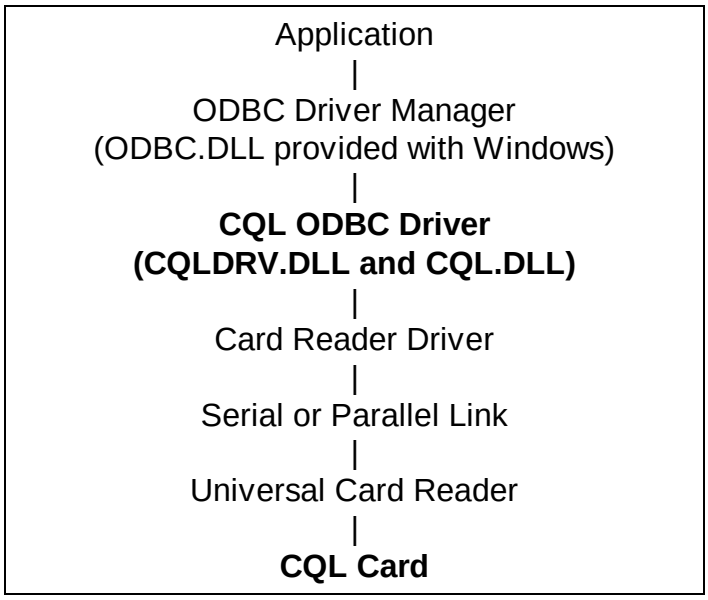

Figure 6. *From application to CQL card via the CQL ODBC driver*

The CQL ODBC driver has been developed using the C language. Today, it carries out all the features of the Core and Level 1 functionalities and processes the Data Manipulation Language (DML) part - i.e. SELECT, INSERT, UPDATE and DELETE - of the SQL grammar defined by ODBC. 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 confine this driver for the use of structured cards and not for the creation of CQL database structures (or personalization).

2.2.3. Installing and using the CQL ODBC driver

Once installed the CQL ODBC Driver, the system [ODBC93] needs a CQL data source name from the list of available data sources managed by the ODBC Administrator.

1 Start the ODBC Administrator.

2 In the Data Sources dialogue box, choose the Add button. The Add Data Source dialogue box is displayed.

3 In the Installed ODBC Drivers list, select CQL and choose the OK button. The CQL ODBC Driver dialogue box is displayed.

4 In the CQL ODBC Driver dialogue box, set the option values as necessary and choose the OK button. The CQL ODBC Driver dialogue box has the following options :

Data Source Name : a name by which identify the data source. For example, "Student Card".
Connexion Port : the port used for connecting the card reader. For example, "COM2".
Session Name : a name by which identify each dialogue session with the CQL Card. For example "FOO".

At this point the CQL ODBC Driver is fully installed and can be used by selected the CQL data source name at the time of the connection with a data source. Establishing a

connection with the CQL card, username and password have to be entered for the card to control the user identity and verify his privileges. After this the user can work with the CQL card like with a dBase File to perform querying (using Q by X or Access Basic) or reporting (using Visual Basic or Crystal Report) without writing code because these products are yet packaged to be ODBC-enabled applications.

Herein a sample (and simple) student card application. This code lists the set of CQL statements to execute for creating the database inside the CQL card.

```
; CARD CONNECTION
connect to COM2 as CQL;

; AMS, SIMPLE USERS, TABLES, VIEWS
; AND DICS CREATED BY THE CARD ISSUER
present E 'TESTCODE';
create user STUD 3 'STUD';
create application HEALTH 3 'HEALTH';
create application SCHOOL 3 'SCHOOL';
create table ID(NAM,SUR,BTHDAY,ADDR,CITY);
create view IDPUB as select NAM,SUR from ID;
create dictionary EDIC;

; TABLES AND DICTIONARIES CREATED BY THE
; APPLICATION MANAGER 'HEALTH'
present HEALTH 'HEALTH';
create table INFO(WGHT,HGHT,BLOODG,VACC);
create table CARES(DATES,PB,CARE,SLEAVE);
create dictionary HDIC;

; USERS, TABLES AND DICTIONARIES CREATED
; BY THE APPLICATION MANAGER 'SCHOOL'
present SCHOOL 'SCHOOL';
create user PROF 3 'PROF';
create table REGIST(YEAR,CYCLE,SESS,TITLE);
create table RESULT(DATE,LESSON,COEFF,MARK);
create dictionary SDIC;

; PRIVILEGES GRANTED BY THE CARD ISSUER
; AND EACH APPLICATION MANAGER
present E 'TESTCODE';
grant select on EDIC_T to PUBLIC;
grant select on IDPUB to PUBLIC;

present HEALTH 'HEALTH';
grant select on INFO to PUBLIC;
grant select on HDIC_T to PUBLIC;

present SCHOOL 'SCHOOL';
grant select on REGIST to PUBLIC;
grant all on RESULT to PROF;
grant select on SDIC_T to PUBLIC;
grant select on RESULT to STUD;

;CARD DISCONNECTION
disconnect CQL;
```

Figure 7. *Set of CQL statements for creating a student database card*

Herein a source for an ODBC-enabled application using the above defined student card. Remark the code is never specific to the CQL card because the choice of the data source name is made during execution : the choice may be a CQL data source or a dBase data source! In any case, the same code applies.

```
/* Access to the database */
retcode = SQLAllocEnv(&henv);                          /* Allocate enviroment handle */
if (retcode == SQL_SUCCESS) {
retcode = SQLAllocConnect(henv, &hdbc);         /* Allocate connection handle */
if (retcode == SQL_SUCCESS) {
     /* Connect to data source */
     retcode = SQLDriverConnect(hdbc, NULL, "", 0, NULL, 0, NULL,
SQL_DRIVER_COMPLETE);
     /* OPEN A CONNECTION DIALOG BOX FOR CHOICE THE DATA SOURCE
      IF IT IS A CQL CARD USERNAME AND PASSWORD WILL BE REQUIRED */
     if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
          /* Process data after successful connection */
          retcode = SQLAllocStmt(hdbc, &hstmt); /* Statement handle */
          if (retcode == SQL_SUCCESS) {
               retcode = SQLPrepare(hstmt,"INSERT INTO RESULT (DATE, LESSON, COEFF,
MARK)
                         VALUES ('94/01/12', 'HISTORY', '2', '15')", SQL_NTS);
               if (retcode == SQL_SUCCESS) {
                    retcode = SQLExecute(hstmt); /* Execute statement with */
               }
               retcode = SQLFreeStmt(hstmt, SQL_DROP);
/* Continued */
```

Figure 8. *Sample of a C code for an ODBC-enabled application*

3. CONCLUSION : ADVANTAGES AND ENHANCEMENTS

In the smart card services providers area, the emergence of application including more and more intervening parties implies need of open systems for smart card. The file management techniques implemented in the earlier smart cards don't meet these demands. The database management systems were a technologic gap for the information systems. The same gap is necessary for smart cards. We propose a smart card embarking a DBMS, it manages users, tables, views, dictionaries, and privileges granted to users on these data structures. Then, each card partner, depending on its role, have specific visions and accesses onto the data maintained by the card.

The CQL card is a computer element that deals with information systems, it has to be integrated into these systems. For this reason we have embraced DBMS standards like SQL (for the set of the card commands) and ODBC for the connectivity. We have implemented software tools enabling the use of the card like any other ODBC data source. The CQL card like other DBMS's can be used as a data provider. Thanks to the database management concepts and the CQL ODBC driver the CQL card becomes easy to use, it requires only the knowledge of SQL and it can be used from usual software. These features enhance implementation, testing, evaluation and simulation of a CQL card application. The conformance to the computer standards protects the corporations' investments.

The CQL card as a database server can be connected to and used from multiple information systems, then the card becomes an active link between these systems. The card is a tool that offers solutions for the problem of interoperability between DBMS's that can't or don't want cooperate. With smart card, and thanks to its small size, each person may carry in his back pocket data about himself like health diagnostics, access conditions to protected buildings, list of debit and credit transactions, etc. There's no question that the success of

smart cards will be going on. The design of information systems have to deal with this new support of data. The challenge is to conciliate the two points of view : the one of the global information system, and particularly, the one of the carrier of the personal and mobile database server. The CQL card and the CQL ODBC driver are a first step in this direction.

ACKNOWLEDGEMENTS

We gratefully acknowledge Georges Grimonprez and Edouard Gordons for supporting our work and for their work onto the conception and implementation of the CQL card. We would like 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 for helpful suggestions and comments on earlier drafts of this paper.

REFERENCES

[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.

[FGGP93] I. Feneyrol, E. Gordons, G. Grimonprez and P. Paradinas. *High level Operating System Instruction*, Card Tech'93, Washington DC, 1993.

[GEM1] Gemplus. *CQL Card Reference Manual V1.0*, Gemplus documentation, 1992.

[GEM2] Gemplus. *CQL Language Reference Manual V1.0*, Gemplus documentation, 1992.

[GEM3] Gemplus. *CQL Developement Kit Manual V1.0*, Gemplus documentation, 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.

[GRI92] G. Grimonprez. *Etude et réalisation d'une carte à microprocesseur intégrée aux SGBD, Mémoire d'Habilitation à diriger la Recherche*, Université de Lille I, France, 1992.

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

[HAY92] M.P. Haye. *The problematic of designing an information system using smart cards*, In Inforsid 92, Clermont-Ferrand, France, 1992.

[ISO9075] ISO/IEC 9075. *Information Technology - Database - SQL*, ISO, 1992.

[LL93] P. S. Lee and J. Lee. *Innovate approaches to smart card integration*, Card Tech'93, Washington DC, 1993.

[ODBC92] Microsoft Corporation. *Microsoft Open Database Connectivity Backgrounder*, Microsoft documentation, october 1992.

[ODBC93] Microsoft Corporation. *Microsoft Open Database Connectivity Software Development Kit 2.0*, Microsoft documentation, 1993.

[PEY94] P. Peyret. *GPR : a smart card reader for all portable equipment*, GMD - Workshop on Smart Cards, Darmstat, Germany, February 1994.

[SCH93] R. Schwartz. *Writing Interoperable Applications with ODBC*, ODBC Program Management, Microsoft documentation, October 1993.
