Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Home Affairs FDHA
**Swiss Federal Archives SFA**

**Berne, 4. December 2013**

# SIARD

## Format Description

Format Version 1.0

Document Version: 1.3

### Summary

This document describes the data format underlying the Swiss Federal Archive's *SIARD Suite* application.

SIARD (Software Independent Archival of Relational Databases) was developed as part of the SFA's ARELDA project for digital archiving. It is a normative description of the file format used for the long term preservation of relational databases.

SIARD is a nonproprietary, published open standard. It is supported by *SIARD Suite,* which can be used to convert real relational databases (e.g. MS Access, Oracle and SQL Server) into the SIARD format. The SIARD format is based on open Standards – e.g. ISO norms Unicode, XML, SQL1999 and the industry standard ZIP. By using internationally accepted standards it aims at guaranteeing long-term preservation of and access to the most commonly used relational database model.

The SIARD format is currently employed by the Swiss Federal Archives as well as by various Swiss government offices and agencies. In May 2008 it was accepted as the official format of the European PLANETS project for archiving relational databases.

## Table of Contents

# 1 Orientation

This is a technical document. Its intended audience are IT specialists concerned with the long-term preservation of relational databases. The document is based on the Swiss government decision of 23 January 2008 concerning the handling of electronic data and documents. The SIARD format specifications are further based on the archiving strategy of the ARELDA project of 11 April 2006.

Within the framework of the abovementioned ARELDA project (ARchiving of ELectronic DAta), the Swiss Federal Archives[1] developed the application *SIARD Suite* for long-term archiving of relational database content.

The application was initially developed in prototype form. In the interim, a newer version is available: *SIARD Suite*

This document does *not* describe the application, but describes the format in which databases are archived.

The published SIARD database format can be used independently of the *SIARD Suite* applications for long-term archiving of electronic databases. If structure and content of a database are migrated to the SIARD format, it will be possible to access the database data at any later time, even if the original database software is either unavailable or not executable. This has been achieved through the best use of suitable internationally supported standards in the SIARD format. This long term interpretability of database content is based essentially on the two ISO standards, XML and SQL:1999.

SIARD was accepted in May 2008 as the official format for archiving relational databases of the European PLANETS[2] project.

The SIARD format was published in 2013 as the official e-government standard eCH-0165[3]. The standard is more precise than this earlier document but does not differ from it materially, except for the URI used to identify the *metadata.xsd*. A changeover to use the

---

1 Swiss Federal Archives SFA [http://www.bar.admin.ch/]

2 PLANETS= Preservation and Long term Access via NETworked Services. See the Planets website at http://www.planets-project.eu/

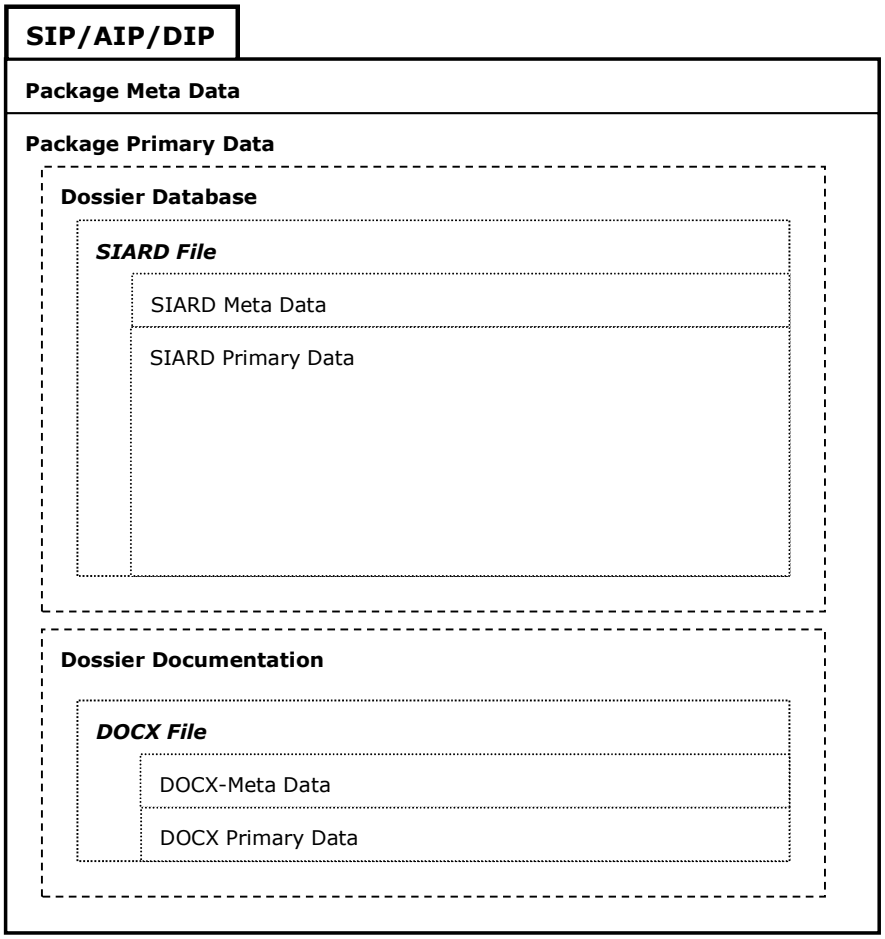
3 The standard eCH-0165 can be downloaded from: http://www.bar.admin.ch/dienstleistungen/00823/00825/

# 2 Scope

Please note that the SIARD format represents the long term storage format for only one specific type of digital document (namely, relational databases) and is thus designed completely independently of package structures such as SIP (Submission Information Package), AIP (Archival Information Package) and DIP (Dissemination Information Package) in the OAIS[4] model.

It is assumed that a database in SIARD format is archived as part of an archive package which contains additional documents (for understanding the database relevant business records...).

Similar to a modern Word or e-mail file which has an internal structure with metadata, primary data and various auxiliary data, an archived relational database contains, alongside the actual primary data, its own metadata that details the records - regardless of the metadata catalogue collected in an archive's OAIS packages.




[4] OAIS = Open Archival Information System, ISO-Standard ISO 14721:2003

# 3 Basic Principles

## 3.1 Use of standards

In order to guarantee interpretability of database content in the long term, the SIARD format is in essence based on both ISO standards XML[5] and SQL:1999[6].

All database contents are stored in a collection of XML files (schema definitions and SQL code, both conforming to SQL:1999). The only exceptions are larger BLOB and CLOB data (Binary Large OBjects and Character Large OBjects), which are stored in separate binary files but referenced in the XML-files.

## 3.2 Databases as documents

A relational database is handled by SIARD as a single document for archiving. Similar to a Word or an e-mail file, it can be made up internally of many parts. Normally, all tables must be archived together in a relational database so that references between the data in individual tables are maintained.

To lend more emphasis to this analogy, a database archived in the SIARD format is stored as a single file. This file is stored as a single (uncompressed) ZIP7 Archive which contains the above-mentioned XML and binary files in a specific folder structure. Different databases are archived in different SIARD-files.

## 3.3 Character sets and characters

In general, all data are stored in a Unicode8 character set. During extraction from databases which support other character sets, mapping to the corresponding Unicode characters is carried out. For this reason, SIARD generally translates national character string types in the database software (NCHAR, NVARCHAR and NCLOB) into non-national types (CHAR, VARCHAR and/or CLOB).

This convention is well supported by XML, independently of whether an XML file is stored in the UTF9-8 or UTF-16 format.

Certain characters, with a special meaning in the XML format in the SIARD format XML files are substituted by entity references, namely in all fields of type xs:string.

---

[5] XML = Extensible Mark-up Language, ISO/IEC 19503:2005

[6] Jim Melton, Alan R. Simon: *SQL:1999 – Understanding Relational Language Components*, Morgan Kaufmann Publishers, 2002, ISBN 1-55860-456-1 and

Jim Melton: *Advanced SQL:1999 – Understanding Object-Relational and Other Advanced Features*, Morgan Kaufmann Publishers, 2003, ISBN 1-55860-677-7

[7] ZIP files were originally defined by Phil Katz. They are very widespread nowadays. Microsoft offers compressed ZIP folders as part of the Windows Operating System. JAVA-Archives (JAR-Files) are ZIP Files.

PKWare's currently published specification (version 6.3.2) can be found at http://www.pkware.com/documents/casestudies/APPNOTE.TXT.

[8] Unicode = an international standard for character encoding systems, corresponding to ISO 10646. The Unicode Consortium: The Unicode Standard, Reading MA, Addison Wesley, 2000.

[9] UTF = Unicode Transformation Format.

In addition the Unicode control characters 0-31 (except 9, 10, and 13) and 127-159 are replaced using a backslash as an escape character, in order to preserve XML conformity.

Character Replacement

Characters that cannot be represented in UNICODE (Codes 0-8, 14-31, 127-159) as well as the escape character '\' and multiple space characters are escaped as \u00<xx> in XML. Quote, less-than and ampersand are represented as entity references in XML.

| Original characters | Characters in SIARD- Format |
|---|---|
| 0 bis 8 | \u0000 bis \u0008 |
| 14-31 | \u000E bis \u001F |
| 32 | \u0020, for multiple spaces |
| & | &amp; |
| < | < |
| \ | \u005c |
| 127 bis 159 | \u007F bis \u009F |

## 3.4 Identifiers

SQL:1999 provides regular designators - without spaces, special characters and where case is irrelevant - which are stored in upper case. It also provides for designators in quotation marks with unambiguous notation, which may contain special characters. In expressions they are surrounded by "double quotes".

The Unicode standard defines which characters are special and which are the capital letter version of an alphabetic character.

A regular designator is stored in the metadata in upper case whereas a delimited designator is stored with the surrounding quotation marks. The SQL:1999 standard states: As soon as a designator contains a character which a normal designator may not contain, it becomes a delimited designator.

# 4 The SIARD-format

## 4.1 Orientation: Relational Databases

A **Database** usually consists of one or more database **Schemata**[10] as well as individual users' and roles' access right definitions on specified portions of the database. In SQL:1999, **Users** and **Roles** can carry **Privileges** (authorisations).

**Schemata** are receptacles for **Tables**, **Views** and **Routines**.

**Tables** consist of a table definition with fields which assign a name and a type to each **Column** of the table, of data records which contain the actual **Primary Data**, an optional **Primary Key**, **Foreign Keys**, which guarantee referential integrity, **Candidate Keys**, which serve to identify a record in the table, and **Check Constraints**, which guarantee consistency. Optionally, **Triggers** can be defined for a table.

**Views** are standard queries stored in the database. The query result is a table which also contains fields and data records, but no constraints.

**SQL Routines** (a.k.a. Stored Procedures) are primarily important for understanding view queries where they can occur in partial expressions.

A relational database thus consists of a quantity of structured database objects (e.g. schema, view, etc.) as well as table contents.

## 4.2 Structure of the SIARD Archive File

A relational database archived in the SIARD format also consists of two components: metadata, which describe the structure of the archived database and primary data, which represent table contents. Furthermore, the metadata provide information about where to find which primary data in the archive.

Database metadata and primary data are stored together in an uncompressed ZIP archive with the filename extension ".siard". The primary are stored in the folder *content* and the metadata in the folder *header.*

The following representation is intended to clarify this structure:

content

- schema1
  - table1
    - table.xsd
    - table.xml
    - lob1[11]
      - record1.txt / record1.bin
    - lob2

---

[10] A database schema is a kind of namespace prefix. A database catalogue contains the metadata of all contained schemata. The "catalog" level in SQL:1999 is equivalent to the "Document Database" which one can convert into archive format using SIARD.

[11] LOB = Large Object, like BLOB = Binary Large OBjects and CLOB = Character Large OBjects, see section 4.1

record1.txt / record1.bin

...

table2

table.xsd

table.xml

...

schema2

...

header

metadata.xsd[12]

metadata.xml

The directories content and header are stored as separate (empty) entries content/ and header/ into the ZIP file. It is necessary that the entry of the *header* directory is inserted after all primary data and before all other metadata entries to facilitate integrity verification of the primary data. The message digest mentioned below is to be computed from offset 0 to the offset of the *header/* entry.

The following descriptions detail the formats of the two folders and the files contained therein.

## 4.3 Metadata in a SIARD Archive

The SIARD archive metadata store the structure of the archived database and indicate which primary data can be found where in the archive.

All metadata are collected in a single *metadata.xml* file in the *header* folder. The file is hierarchically constructed, similarly to a relational database.

The following paragraphs describe the contents of the individual levels. A "yes" in the "opt." column indicates that the item is optional, i.e. may be missing.

### 4.3.1 Database Level Metadata

The *metadata.xml* file contains following global information at the level of database:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| version | no | Format version (SIARD format version, not a SIARD program's!); currently always “1.0” |
| dbname | no | Short database identifier |
| description | yes | The meaning and content of the database as a whole. |
| archiver | yes | Name of the person who carried out the archiving of the primary data from the database |
| archiverContact | yes | Contact data (telephone, e-mail) of the person who carried out the archiving of the primary data from the database |

[12] XSD = XML-Schema-Definition.

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| dataOwner | no | Owner of the data in the database; the institution or person that, at the moment of archiving, has the right to grant usage rights for the data and that is responsible for the observance of legal obligations, such as data protection guidelines. |
| dataOriginTimespan | no | Origination period of the data in the database; an approximate time as text |
| producerApplication | yes | Name and version of program that originally produced the SIARD file from the database |
| archivalDate | no | Archiving date; Date on which the primary data were archived. |
| messageDigest | no | Hexadecimal message digest code over the Folder *content* with a prefix which indicates the type of the Digest-Algorithm (MD5[13] or SHA1[14]).<br>The message digest facilitates a quick check of the integrity of the primary data. |
| clientMachine | yes | DNS[15] Name of the (client) computer on which the archiving was carried out |
| databaseProduct | yes | Database product and version, from which the archiving of the primary data was done |
| connection | yes | The connection string used for archiving the primary data |
| databaseUser | yes | Database UserId of the SIARD-tool user for archiving the primary data from the database |
| schemas | no | Lists of schemata in the database |
| users | no | List of the database users |
| roles | yes | List of the database roles |
| privileges | yes | List of the user and role privileges |

### 4.3.2 Schema Level Metadata

Schema metadata are archived in the *metadata.xml* file along with the global database information.

The following schema metadata are stored in the *metadata.xml* file*:*

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Database schema name |
| folder | no | Name of the schema folder under *content* in the SIARD archive |
| description | yes | Description of the meaning and content of the schema |
| tables | no | List of the tables in the database |

[13] MD5 = Message-Digest Algorithm 5.

[14] SHA = Secure Hash Algorithm.

[15] DNS = Domain Name System, a distributed database which administers the Internet namespace.

| | | |
|---|---|---|
| views | yes | List of the queries stored in the database |
| routines | yes | List of routines (formerly stored procedures) in the schema |

The primary data of each schema is archived in the SIARD archive in a *content* subfolder. The schema name of the database is not suitable as a folder name because the SQL syntax allows names which are unacceptable in many file systems.

Only the simplest names for folders and files with ASCII characters are allowed within the SIARD format, namely digits and alphabetic characters, without special characters which are guaranteed compatible with all file systems types. Thus, SIARD generates the schema folder names *schema1*, *schema2*...

### 4.3.3 Table Level Metadata

Table level metadata are stored in the *metadata.xml* file along with global database data and schema metadata.

The following table metadata are stored in the *metadata.xml* file*:*

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Table name in the schema |
| folder | no | Name of the table folder in the schema folder |
| description | yes | Description of the meaning and content of the table |
| columns | no | List of the columns in the table |
| primaryKey | yes | Primary key of the table |
| foreignKeys | yes | List of foreign keys of the table |
| candidateKeys | yes | List of candidate keys of the table |
| checkConstraints | yes | List of the check constraints on the table |
| triggers | yes | List of table triggers |
| rows | no | Number of datasets in the table |

The primary data of a table is archived in the SIARD-archive in the *content* folder in a sub-folder of the schema to which the table belongs. SIARD automatically generates the names *table1*, *table2*, *table3*...

### 4.3.4 Column level Metadata

Column level metadata are stored in the *metadata.xml* file along with global database data, schema and table metadata.

The following column metadata are stored in the *metadata.xml* file*:*

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Column name in the table |
| folder | yes | Name of the LOB folder in the table folder |
| type | no | SQL:1999 column type |

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| typeOriginal | yes | original column type |
| defaultValue | yes | Column default |
| nullable | yes | Optional entry |
| description | yes | The meaning and content of the column |

Because various SQL database programs (designated as conformant) support very different data types, the *original* type is listed here along with the SQL:1999 type. The translation of the proprietary type to the SQL:1999 type is to be defined and documented for each database program supporting the SIARD format.

The optional LOB folder name is needed only for columns of type Large Object (e.g. BLOB or CLOB). SIARD automatically generates the names *lob1*, *lob2*...

The files which represent the Large Object fields are created in these folders and called *record1.txt*, *record2.txt*, or *record1.bin*, *record2.bin...* These are referenced in the data XML file.

All column constraints except nullability are stored as table constraints.

### 4.3.5 Primary key Metadata

The following primary key metadata are stored in the *metadata.xml* file*:*

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the primary key |
| column | no | List of names of columns of the primary key |
| description | yes | The meaning and content of the primary key |

### 4.3.6 Foreign key Metadata

The following foreign key metadata are stored in the *metadata.xml* file*:*

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the foreign key |
| referenced-Schema | no | Schema of external table referenced by foreign key |
| referencedTable | no | External table referenced by foreign key |
| reference | no | Reference (list of columns and referenced columns) |
| matchType | yes | Match type (FULL, PARTIAL or SIMPLE) |
| deleteAction | yes | Delete action, e.g. CASCADE |
| updateAction | yes | Update action e.g. SET DEFAULT |
| description | yes | The meaning and content of the foreign key |

The referenced external table name can be of type *table* or *schema.table*. In this case, delimited data names are set in quotation marks.

The match type is FULL, PARTIAL or SIMPLE. The delete and change actions contain those approved in the SQL:1999 standard.

### 4.3.7 Reference Metadata

The list of references consists of the following elements:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| column | no | Name of the column |
| referenced | no | Name of the referenced column |

### 4.3.8 Candidate key Metadata

The following candidate key metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the candidate key in the database schema |
| column | no | List of names of columns of the candidate key |
| description | yes | The meaning and content of the candidate key |

### 4.3.9 Check constraint Metadata

The Check constraint consists of a condition to be examined. This is as an expression of type BOOLEAN (with value *true*, *false* or *unknown*) as defined in SQL:1999 syntax.

The following check constraint metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the check constraint |
| condition | no | Condition of the check constraint |
| description | yes | The meaning and content of the check constraint |

### 4.3.10 Trigger level Metadata

The following trigger metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the trigger in the table |
| actionTime | no | BEFORE or AFTER |
| triggerEvent | no | INSERT, DELETE, UPDATE [OF < trigger column list>] |
| aliasList | yes | <old or new value alias list> |
| triggeredAction | no | <triggered action> |
| description | yes | The trigger's meaning and content |

### 4.3.11 View Level Metadata

The following view metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the view in the schema |
| columns | no | List of the view's columns |
| query | yes | SQL:1999 query which defines the view |
| queryOriginal | yes | Original SQL query which defines the view |
| description | yes | The meaning and content of the view |

As the different so-called SQL conformant database programs allow very different query syntax, the *original* query is stored along with the SQL:1999 query. For each database program which supports the SIARD format, a translation of the proprietary query syntax to SQL:1999 queries is to be defined and documented.

The column metadata of a view have the same structure as the column metadata of a table.

### 4.3.12 Routine Level Metadata

The following routine metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the routine in the schema |
| description | yes | The meaning and content of the routine |
| source | yes | Original source code of the routine (VBA, PL/SQL, JAVA) |
| body | yes | Source code of the SQL:1999 compatible routine |
| characteristic | yes | Routine characteristics |
| returnType | yes | Routine's return type (if it is a function) |
| parameters | yes | Parameter list |

As many database programs provide proprietary routines, where no transformation to a proper SQL:1999 query is possible, the original source code of the routine can be archived here, e.g. PL/SQL in the case of Oracle databases or VBA in the case of MS Access modules.

### 4.3.13 Parameter Metadata

The following metadata are stored in the *metadata.xml* file for each parameter:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the parameter |
| mode | no | Parameter mode (IN, OUT or INOUT) |
| type | no | The parameter's SQL:1999 type |

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| typeOriginal | yes | Parameter's original type |
| description | yes | The meaning and function of the Routine |

As with the column descriptions, the *original*, proprietary, parameter type can be given here.

### 4.3.14 User Level Metadata

The following user metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | User name |
| description | yes | The user's function and significance |

### 4.3.15 Role Level Metadata

The following role metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| name | no | Name of the role |
| admin | no | Administrator of the role (user or role) |
| description | yes | The role's meaning and function |

### 4.3.16 Privilege Level Metadata

The following privilege metadata are stored in the *metadata.xml* file:

| *Identifier* | *opt.* | *Description* |
|---|---|---|
| type | no | privilege granted (e.g. SELECT) |
| object | yes | Object on which the privilege is to be applied |
| grantor | no | Authoriser who grants the privileg |
| grantee | no | Receiver (user or role) of the privilege |
| option | yes | Grant option (ADMIN or GRANT) |
| description | yes | The grant's significance and function |

## 4.4 Primary Data in the SIARD Archive

As previously stated, the primary data of an archived relational database can be found in the *content* folder in the SIARD archive's document root. If this file is empty (and there is no digest code in the metadata), it is an empty SIARD archive that contains only metadata definitions which describe a database structure.

The primary data of each table is archived in the SIARD archive in the *content* folder, in a schema subfolder to which the table belongs. SIARD generates the names *schema1*, *schema2*, *schema3* ...., automatically for the schema folders and *table1*, *table2*, *table3* ....for the table folders.

Table data (primary data) are stored in a XML file named *table.xml.*

For each table, an XML schema definition (*table.xsd*) is generated, which defines the primary data's XML storage format. This schema definition reflects the table's SQL schema metadata and specifies that the table is stored as a sequence of lines which contain a sequence of column entries with different XML types. The name of the table tag is *table*; that of the record tag is *row*; that of the column tag is *c1, c2...* (or *siard: c1, siard: c2...,* if the name space were explicitly included).

Values are transformed to their corresponding XML types in compliance with SQL/XML[16].

When a table contains data of large object type (BLOB, CLOB...) that are larger than 4000 characters and/or 2000 bytes, separate files are produced for each. SIARD automatically generates the folder names *lob1*, *lob2*... for each affected column. The files that represent the large object fields are created in these folders and called *record1.txt*, *record2.txt*, or *record1.bin*, *record2.bin....*

To avoid creating empty folders, folders are only created when needed, i.e. when they contain data.

16 http://www.sqlx.org/

# 5 Appendix: XML Schema Definitions

## 5.1 metadata.xsd

The XML schema definition (*metadata.xsd*) defines the structure of the *metadata.xml* file in the *header* folder.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- $Workfile: metadata.xsd $ ********************************** -->
<!-- Metadata schema for SIARD 1.0                                  -->
<!-- Version    : $Id: metadata.xsd 680 2008-02-14 19:58:41Z hartwig $ -->
<!-- Application: SIARD Suite                                       -->
<!--    Software-Independent Archival of Relational Databases       -->
<!-- Platform   : XML 1.0, XML Schema 2001                          -->
<!-- Description: This XML schema definition defines the structure  -->
<!--    of the metadata in the SIARD format                         -->
<!-- ************************************************************ -->
<!-- Copyright  :  2007, Swiss Federal Archives, Berne, Switzerland -->
<!-- ************************************************************ -->
<xs:schema id="metadata"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd"
  targetNamespace="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <!-- root element of an XML file conforming to this XML schema -->
  <xs:element name="siardArchive">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>
          Root element of meta data of the SIARD archive
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <!-- name of the archived database -->
        <xs:element name="dbname" type="mandatoryString"/>
        <!-- short free form description of the database content -->
        <xs:element name="description" type="xs:string" minOccurs="0"/>
        <!-- name of person responsible for archiving the database -->
        <xs:element name="archiver" type="xs:string" minOccurs="0"/>
        <!-- contact data (telephone number or email adress) of archiver -->
        <xs:element name="archiverContact" type="xs:string" minOccurs="0"/>
        <!-- name of data owner (section and institution responsible for data)
             of database when it was archived -->
        <xs:element name="dataOwner" type="mandatoryString"/>
        <!-- time span during which data where entered into the database -->
        <xs:element name="dataOriginTimespan" type="mandatoryString"/>
        <!-- date of creation of archive (automatically generated by SIARD) -->
        <xs:element name="archivalDate" type="xs:date"/>
        <!-- message digest code over all primary data in folder "content" -->
        <xs:element name="messageDigest" type="digestType"/>
```

```
      <!-- DNS name of client machine from which SIARD was running for archiving -->
      <xs:element name="clientMachine" type="xs:string" minOccurs="0"/>
      <!-- name of database product and version from which database originates -->
      <xs:element name ="databaseProduct" type="xs:string" minOccurs="0"/>
      <!-- connection string used for archiving -->
      <xs:element name="connection" type="xs:string" minOccurs="0"/>
      <!-- database user used for archiving -->
      <xs:element name="databaseUser" type="xs:string" minOccurs="0"/>
      <!--  list of schemas in database  -->
      <xs:element name="schemas" type="schemasType"/>
      <!-- list of users in the archived database -->
      <xs:element name="users" type="usersType"/>
      <!-- list of roles in the archived database -->
      <xs:element name="roles" type="rolesType" minOccurs="0"/>
      <!-- list of privileges in the archived database -->
      <xs:element name="privileges" type="privilegesType" minOccurs="0"/>
    </xs:sequence>
    <!-- constraint: version number must be 1.0 -->
    <xs:attribute name="version" type="versionType" use="required" />
  </xs:complexType>
</xs:element>

<!-- complex type schemas -->
<xs:complexType name="schemasType">
  <xs:annotation>
    <xs:documentation>
      List of schemas
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="schema" type="schemaType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type schema -->
<xs:complexType name="schemaType">
  <xs:annotation>
    <xs:documentation>
      Schema element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the schema -->
    <xs:element name="name" type="xs:string" />
    <!-- archive name of the schema folder -->
    <xs:element name="folder" type="fsName"/>
    <!-- description of the schema's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <!-- list of tables in the schema -->
    <xs:element name="tables" type="tablesType"/>
    <!-- list of views in the schema -->
    <xs:element name="views" type="viewsType" minOccurs="0"/>
    <!-- list of routines in the archived database -->
```

```
    <xs:element name="routines" type="routinesType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type tables -->
<xs:complexType name="tablesType">
  <xs:annotation>
    <xs:documentation>
      List of tables
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="table" type="tableType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type table -->
<xs:complexType name="tableType">
  <xs:annotation>
    <xs:documentation>
      Table element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the table -->
    <xs:element name="name" type="xs:string"/>
    <!-- archive name of the table folder -->
    <xs:element name="folder" type="fsName"/>
    <!-- description of the table's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <!-- list of columns of the table -->
    <xs:element name="columns" type="columnsType"/>
    <!--  primary key -->
    <xs:element name="primaryKey" type="primaryKeyType" minOccurs="0"/>
    <!--  foreign keys  -->
    <xs:element name="foreignKeys" type="foreignKeysType" minOccurs="0"/>
    <!--  candidate keys (unique constraints)  -->
    <xs:element name="candidateKeys" type="candidateKeysType" minOccurs="0"/>
    <!-- list of (check) constraints -->
    <xs:element name="checkConstraints" type="checkConstraintsType" minOccurs="0"/>
    <!--  list of triggers  -->
    <xs:element name="triggers" type="triggersType" minOccurs="0"/>
    <!--  number of rows in the table -->
    <xs:element name="rows" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type views -->
<xs:complexType name="viewsType">
  <xs:annotation>
    <xs:documentation>
      List of views
    </xs:documentation>
```

```
  </xs:annotation>
  <xs:sequence>
    <xs:element name="view" type="viewType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type view -->
<xs:complexType name="viewType">
  <xs:annotation>
    <xs:documentation>
      View element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the view -->
    <xs:element name="name" type="xs:string" />
    <!-- SQL query string defining the view -->
    <xs:element name="query" type="xs:string" minOccurs="0"/>
    <!-- original query string defining the view -->
    <xs:element name="queryOriginal" type="xs:string" minOccurs="0"/>
    <!-- description of the view's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <!-- list of columns of the view -->
    <xs:element name="columns" type="columnsType"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type columns -->
<xs:complexType name="columnsType">
  <xs:annotation>
    <xs:documentation>
      List of columns
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="column" type="columnType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type column -->
<xs:complexType name="columnType">
  <xs:annotation>
    <xs:documentation>
      Column element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the column -->
    <xs:element name="name" type="xs:string" />
    <!-- archive name of the lob folder -->
    <xs:element name="folder" type="fsName" minOccurs="0"/>
    <!-- SQL:1999 data type of the column -->
    <xs:element name="type" type="xs:string" />
```

```
      <!-- original data type of the column -->
      <xs:element name="typeOriginal" type="xs:string" minOccurs="0"/>
      <!-- default value -->
      <xs:element name="defaultValue" type="xs:string" minOccurs="0"/>
      <!-- nullability -->
      <xs:element name="nullable" type="xs:boolean"/>
      <!-- unique, references, check column constraints
           are stored as table constraints -->
      <!-- description of the column's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type primaryKey -->
  <xs:complexType name="primaryKeyType">
    <xs:annotation>
      <xs:documentation>
        primaryKey element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the primary key -->
      <xs:element name="name" type="xs:string" minOccurs="0" />
       <!-- description of the primary key's meaning and content -->
       <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!-- columns belonging to the primary key -->
      <xs:element name="column" type="xs:string"  minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type foreignKeys -->
  <xs:complexType name="foreignKeysType">
    <xs:annotation>
      <xs:documentation>
        List of foreign key constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element  name="foreignKey"  type="foreignKeyType"  minOccurs="1"  maxOccurs="unbounded"
/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type foreignKey -->
  <xs:complexType name="foreignKeyType">
    <xs:annotation>
      <xs:documentation>
        foreignKey element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the foreign key -->
      <xs:element name="name" type="xs:string" />
```

```
      <!--  referenced schema -->
      <xs:element name="referencedSchema" type="xs:string"/>
      <!-- referenced table -->
      <xs:element name="referencedTable" type="xs:string"/>
      <!--  references -->
      <xs:element name="reference" type="referenceType" minOccurs="1" maxOccurs="unbounded"/>
      <!-- match type (FULL, PARTIAL, SIMPLE) -->
      <xs:element name="matchType" type="matchTypeType" minOccurs="0"/>
      <!-- ON DELETE action e.g. ON DELETE CASCADE -->
      <xs:element name="deleteAction" type="xs:string" minOccurs="0"/>
      <!-- ON UPDATE action e.g. ON UPDATE SET DEFAULT -->
      <xs:element name="updateAction" type="xs:string" minOccurs="0"/>
      <!-- description of the foreign key's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type reference -->
  <xs:complexType name="referenceType">
    <xs:annotation>
      <xs:documentation>
        reference element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- referencing column -->
      <xs:element name="column" type="xs:string"/>
      <!-- referenced column (table.column) -->
      <xs:element name="referenced" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type candidateKeys -->
  <xs:complexType name="candidateKeysType">
    <xs:annotation>
      <xs:documentation>
        List of candidate key (unique) constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element     name="candidateKey"     type="candidateKeyType"     minOccurs="1"     max-
Occurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type candidateKey -->
  <xs:complexType name="candidateKeyType">
    <xs:annotation>
      <xs:documentation>
        candiate key (unique) element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
```

```
      <!-- database name of the candidate key -->
      <xs:element name="name" type="xs:string"/>
       <!-- description of the candidate key's meaning and content -->
       <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!-- columns belonging to the candidate key -->
      <xs:element name="column" type="xs:string"  minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type check constraints -->
  <xs:complexType name="checkConstraintsType">
    <xs:annotation>
      <xs:documentation>
        List of check constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element   name="checkConstraint"   type="checkConstraintType"   minOccurs="1"   max-
Occurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type check constraint -->
  <xs:complexType name="checkConstraintType">
    <xs:annotation>
      <xs:documentation>
        Check constraint element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the constraint -->
      <xs:element name="name" type="xs:string"/>
      <!-- check condition -->
      <xs:element name="condition" type="xs:string"/>
      <!-- description of the constraint's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type triggers -->
  <xs:complexType name="triggersType">
    <xs:annotation>
      <xs:documentation>
        List of triggers
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="trigger" type="triggerType" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type trigger -->
  <xs:complexType name="triggerType">
```

```
  <xs:annotation>
    <xs:documentation>
      Trigger element in siardArchive
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the trigger -->
    <xs:element name="name" type="xs:string" />
    <!--  action time -->
    <xs:element name="actionTime" type="actionTimeType"/>
    <!--  trigger event INSERT, DELETE, UPDATE [OF <trigger column list>] -->
    <xs:element name="triggerEvent" type="xs:string"/>
    <!--  alias list <old or new values alias> -->
    <xs:element name="aliasList" type="xs:string" minOccurs="0"/>
    <!--  triggered action -->
    <xs:element name="triggeredAction" type="xs:string"/>
    <!-- description of the trigger's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type routines -->
<xs:complexType name="routinesType">
  <xs:annotation>
    <xs:documentation>
      List of routines
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="routine" type="routineType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type routine -->
<xs:complexType name="routineType">
  <xs:annotation>
    <xs:documentation>
      Routine
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of routine in schema -->
    <xs:element name="name" type="xs:string"/>
    <!-- description of the routines's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <!-- original source code (VBA, PL/SQL, ...) defining the routine -->
    <xs:element name="source" type="xs:string" minOccurs="0"/>
    <!--  SQL:1999 body of routine  -->
    <xs:element name="body" type="xs:string" minOccurs="0"/>
    <!--  routine characteristic -->
    <xs:element name="characteristic" type="xs:string" minOccurs="0"/>
    <!--  SQL:1999 data type of the return value (for functions) -->
    <xs:element name="returnType" type="xs:string" minOccurs="0"/>
```

```
    <!--  list of parameters -->
    <xs:element name="parameters" type="parametersType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type parameters -->
<xs:complexType name="parametersType">
  <xs:annotation>
    <xs:documentation>
      List of parameters of a routine
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="parameter" type="parameterType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type parameter -->
<xs:complexType name="parameterType">
  <xs:annotation>
    <xs:documentation>
      Parameter of a routine
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!--  name of parameter  -->
     <xs:element name="name" type="xs:string"/>
     <!--  mode of parameter (IN, OUT, INOUT) -->
     <xs:element name="mode" type="xs:string"/>
     <!--  SQL:1999 type of argument -->
     <xs:element name="type" type="xs:string"/>
     <!-- original data type of the argument -->
     <xs:element name="typeOriginal" type="xs:string" minOccurs="0"/>
     <!-- description of the parameter's meaning and content -->
     <xs:element name="description" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type users -->
<xs:complexType name="usersType">
  <xs:annotation>
    <xs:documentation>
      List of users
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="user" type="userType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type user -->
<xs:complexType name="userType">
  <xs:annotation>
```

```
    <xs:documentation>
      User
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- user name -->
    <xs:element name="name" type="xs:string"/>
    <!-- description of the user's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type roles -->
<xs:complexType name="rolesType">
  <xs:annotation>
    <xs:documentation>
      List of roles
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="role" type="roleType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<!-- complex type role -->
<xs:complexType name="roleType">
  <xs:annotation>
    <xs:documentation>
      Role
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- role name -->
    <xs:element name="name" type="xs:string"/>
    <!-- role ADMIN (user or role) -->
    <xs:element name="admin" type="xs:string"/>
    <!-- description of the role's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- complex type privileges -->
<xs:complexType name="privilegesType">
  <xs:annotation>
    <xs:documentation>
      List of grants
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="privilege" type="privilegeType" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

```
<!-- complex type privilege -->
<xs:complexType name="privilegeType">
  <xs:annotation>
    <xs:documentation>
      Grant (incl. grant of role)
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- privilege type (incl. ROLE privilege or "ALL PRIVILEGES" -->
    <xs:element name="type" type="xs:string"/>
    <!-- privilege object (may be omitted for ROLE privilege) -->
    <xs:element name="object" type="xs:string" minOccurs="0"/>
    <!-- GRANTED BY -->
    <xs:element name="grantor" type="xs:string"/>
    <!-- user list of users or roles or single value "PUBLIC" -->
    <xs:element name="grantee" type="xs:string"/>
    <!-- optional option "GRANT" or "ADMIN" -->
    <xs:element name="option" type="privOptionType" minOccurs="0"/>
    <!-- description of the grant's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- simple type for digest string -->
<xs:simpleType name="digestType">
  <xs:annotation>
    <xs:documentation>
      digestType must be empty or prefixed by MD5 oder SHA1
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:pattern value="((MD5|SHA-1).*)?"/>
  </xs:restriction>
</xs:simpleType>

<!-- simple type for version number -->
<xs:simpleType name="versionType">
  <xs:annotation>
    <xs:documentation>
      versionType must be constrained to "1.0"
      for conformity with this XLM schema
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:enumeration value="1.0"/>
  </xs:restriction>
</xs:simpleType>

<!-- simple type for privilege option -->
<xs:simpleType name="privOptionType">
  <xs:annotation>
```

```
    <xs:documentation>
      privOptionType must be "ADMIN" or "GRANT"
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:enumeration value="ADMIN"/>
    <xs:enumeration value="GRANT"/>
  </xs:restriction>
</xs:simpleType>

<!-- simple type for mandatory string
     which must contain at least 1 character -->
<xs:simpleType name="mandatoryString">
  <xs:annotation>
    <xs:documentation>
      mandatoryType must contain at least 1 character
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="preserve"/>
    <xs:minLength value="1" />
  </xs:restriction>
</xs:simpleType>

<!-- simple type of a filesystem (file or folder) name -->
<xs:simpleType name="fsName">
  <xs:annotation>
    <xs:documentation>
      fsNames may only consist of ASCII characters and digits
      and must start with a non-digit
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="([a-z]|[A-Z])([a-z]|[A-Z]|[0-9]).*" />
    <xs:minLength value="1" />
  </xs:restriction>
</xs:simpleType>

<!-- simple type for action time of a trigger -->
<xs:simpleType name="actionTimeType">
  <xs:annotation>
    <xs:documentation>
      actionTime is BEFORE or AFTER
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="BEFORE" />
    <xs:enumeration value="AFTER" />
  </xs:restriction>
</xs:simpleType>

<!-- simple type for match type of a foreign key -->
```

```
  <xs:simpleType name="matchTypeType">
    <xs:annotation>
      <xs:documentation>
        matchType is FULL, PARTIAL or SIMPLE
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="FULL" />
      <xs:enumeration value="PARTIAL" />
      <xs:enumeration value="SIMPLE" />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

## 5.2 metadata.xml

A metadata.xml file which conforms to the XML schema definition is given as an example here:

```
<?xml version="1.0" encoding="UTF-8"?>
<siardArchive                      xmlns="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"                          version="1.0"
xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd metadata.xsd">
  <dbname>jdbc:oracle:thin:@dbhost.enternet.ch:1521:SIARD1</dbname>
  <dataOwner>SIARD</dataOwner>
  <dataOriginTimespan>Fri May 16 11:21:39 CEST 2008</dataOriginTimespan>
  <archivalDate>2008-05-16</archivalDate>
  <messageDigest>MD5B9FB4FA23EFC27F10957533D747A4300</messageDigest>
  <clientMachine>blue2400.enterag.ch</clientMachine>
  <databaseProduct>
    Oracle Oracle9i Enterprise Edition Release 9.2.0.1.0 -
    Production\u000AWith the Partitioning, OLAP and Oracle Data Mining
    options\u000AJServer Release 9.2.0.1.0 - Production
  </databaseProduct>
  <connection>jdbc:oracle:thin:@dbhost.enternet.ch:1521:SIARD1</connection>
  <databaseUser>SIARD</databaseUser>
  <schemas>
    <schema>
      <name>SIARD</name>
      <folder>schema0</folder>
      <tables>
        <table>
          <name>TABLETEST</name>
          <folder>table0</folder>
          <description/>
          <columns>
            <column>
              <name>NID</name>
              <type>DECIMAL(38,0)</type>
              <typeOriginal>NUMBER</typeOriginal>
              <nullable>false</nullable>
            </column>
```

```
      <column>
        <name>SNAME</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>TSCREATED</name>
        <type>DATE</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>false</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>TABLETEST_PK</name>
      <column>NID</column>
      <column>TSCREATED</column>
    </primaryKey>
    <candidateKeys>
      <candidateKey>
        <name>"Unique2"</name>
        <column>SNAME</column>
        <column>TSCREATED</column>
      </candidateKey>
      <candidateKey>
        <name>UNIQUE1</name>
        <column>TSCREATED</column>
      </candidateKey>
    </candidateKeys>
    <rows>2</rows>
  </table>
  <table>
    <name>"TableTest1"</name>
    <folder>table1</folder>
    <description/>
    <columns>
      <column>
        <name>"nID"</name>
        <type>DECIMAL(38,0)</type>
        <typeOriginal>NUMBER</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>"sName"</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>"tsCreated"</name>
        <type>DATE</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>false</nullable>
```

```
      </column>
    </columns>
    <primaryKey>
      <name>"TableTest1_PK"</name>
      <column>"nID"</column>
    </primaryKey>
    <foreignKeys>
      <foreignKey>
        <name>"TableTest1_FK"</name>
        <referencedSchema>SIARD</referencedSchema>
        <referencedTable>TABLETEST</referencedTable>
        <reference>
          <column>"nID"</column>
          <referenced>NID</referenced>
        </reference>
        <reference>
          <column>"tsCreated"</column>
          <referenced>TSCREATED</referenced>
        </reference>
        <deleteAction>RESTRICT</deleteAction>
        <updateAction>CASCADE</updateAction>
      </foreignKey>
    </foreignKeys>
    <candidateKeys>
      <candidateKey>
        <name>UNIQUE3</name>
        <column>"tsCreated"</column>
      </candidateKey>
    </candidateKeys>
    <rows>0</rows>
  </table>
  <table>
    <name>TABLETEST2</name>
    <folder>table2</folder>
    <description/>
    <columns>
      <column>
        <name>NID</name>
        <type>DECIMAL(38,0)</type>
        <typeOriginal>NUMBER</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>SNAME</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>SCLOB</name>
        <folder>lob3</folder>
        <type>CHARACTER LARGE OBJECT(4000)</type>
        <typeOriginal>CLOB</typeOriginal>
```

```
    <nullable>false</nullable>
  </column>
  <column>
    <name>TSCREATED</name>
    <type>DATE</type>
    <typeOriginal>DATE</typeOriginal>
    <nullable>false</nullable>
  </column>
  <column>
    <name>COLBINARYLARGEOBJECT</name>
    <folder>lob5</folder>
    <type>BINARY LARGE OBJECT(4000)</type>
    <typeOriginal>BLOB</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>COLCHARACTER</name>
    <type>CHARACTER(2)</type>
    <typeOriginal>CHAR</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>COLDATE</name>
    <type>DATE</type>
    <typeOriginal>DATE</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>COLDECIMAL</name>
    <type>DECIMAL(2,1)</type>
    <typeOriginal>NUMBER</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>COLDOUBLEPRECISION</name>
    <type>DECIMAL(22,0)</type>
    <typeOriginal>NUMBER</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>COLFLOAT</name>
    <type>DECIMAL(22,0)</type>
    <typeOriginal>NUMBER</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>COLINTEGER</name>
    <type>DECIMAL(38,0)</type>
    <typeOriginal>NUMBER</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>COLNUMERIC</name>
```

```
        <type>DECIMAL(2,1)</type>
        <typeOriginal>NUMBER</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>COLREAL</name>
        <type>DECIMAL(22,0)</type>
        <typeOriginal>NUMBER</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>COLSMALLINT</name>
        <type>DECIMAL(38,0)</type>
        <typeOriginal>NUMBER</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>TABLETEST2_PK</name>
      <column>NID</column>
    </primaryKey>
    <rows>4</rows>
  </table>
</tables>
<views>
  <view>
    <name>"View1"</name>
    <columns>
      <column>
        <name>"nID"</name>
        <type>DECIMAL(38,0)</type>
        <typeOriginal>NUMBER</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>"sName"</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>"tsCreated"</name>
        <type>DATE</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>false</nullable>
      </column>
    </columns>
  </view>
  <view>
    <name>VIEW2</name>
    <columns>
      <column>
        <name>"nID"</name>
```

```
        <type>DECIMAL(38,0)</type>
        <typeOriginal>NUMBER</typeOriginal>
        <nullable>false</nullable>
      </column>
      <column>
        <name>"sName"</name>
        <type>CHARACTER VARYING(31)</type>
        <typeOriginal>VARCHAR2</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>"tsCreated"</name>
        <type>DATE</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>false</nullable>
      </column>
    </columns>
  </view>
</views>
<routines>
  <routine>
    <name>IS_LE</name>
    <description>
      Standalone procedure or function
    </description>
    <returnType>NUMERIC</returnType>
    <parameters>
      <parameter>
        <name>S1</name>
        <mode>IN</mode>
        <type>CHARACTER VARYING</type>
        <typeOriginal>VARCHAR2</typeOriginal>
      </parameter>
      <parameter>
        <name>S2</name>
        <mode>IN</mode>
        <type>CHARACTER VARYING</type>
        <typeOriginal>VARCHAR2</typeOriginal>
      </parameter>
    </parameters>
  </routine>
  <routine>
    <name>"IsZero"</name>
    <description>
      Standalone procedure or function
    </description>
    <returnType>NUMERIC</returnType>
    <parameters>
      <parameter>
        <name>I</name>
        <mode>IN</mode>
        <type>DECIMAL(38,0)</type>
        <typeOriginal>NUMBER</typeOriginal>
```

```
          </parameter>
        </parameters>
      </routine>
    </routines>
  </schema>
</schemas>
<users>
  <user>
    <name>SIARD3</name>
  </user>
  <user>
    <name>SIARD1</name>
  </user>
  <user>
    <name>SIARD2</name>
  </user>
  <user>
    <name>SIARD</name>
  </user>
</users>
<roles>
  <role>
    <name>"siardrole2"</name>
    <admin/>
  </role>
  <role>
    <name>SIARDROLE3</name>
    <admin/>
  </role>
  <role>
    <name>SIARDROLE1</name>
    <admin/>
  </role>
  <role>
    <name>RESOURCE</name>
    <admin/>
  </role>
  <role>
    <name>CONNECT</name>
    <admin/>
  </role>
</roles>
<privileges>
  <privilege>
    <type>SELECT</type>
    <object>TABLE TABLETEST2</object>
    <grantor>SIARD</grantor>
    <grantee>SIARD2</grantee>
  </privilege>
  <privilege>
    <type>SELECT</type>
    <object>TABLE "TableTest1"</object>
    <grantor>SIARD</grantor>
```

```
      <grantee>SIARD1</grantee>
    </privilege>
    <privilege>
      <type>SELECT</type>
      <object>TABLE "TableTest1"</object>
      <grantor>SIARD</grantor>
      <grantee>SIARD3</grantee>
    </privilege>
    <privilege>
      <type>SELECT</type>
      <object>TABLE TABLETEST</object>
      <grantor>SIARD</grantor>
      <grantee>SIARD2</grantee>
    </privilege>
    <privilege>
      <type>UPDATE</type>
      <object>TABLE TABLETEST2</object>
      <grantor>SIARD</grantor>
      <grantee>SIARD2</grantee>
    </privilege>
    <privilege>
      <type>UPDATE</type>
      <object>TABLE TABLETEST2</object>
      <grantor>SIARD</grantor>
      <grantee>SIARD3</grantee>
    </privilege>
    <privilege>
      <type>SELECT</type>
      <object>TABLE "TableTest1"</object>
      <grantor>SIARD</grantor>
      <grantee>"siardrole2"</grantee>
    </privilege>
  </privileges>
</siardArchive>
```

## 5.3 Example of a Table's XML Schema Definition: table0.xsd

For each table, SIARD produces an XML schema definition which assigns the correct XML data types to the columns.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                    xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd"                          at-
tributeFormDefault="unqualified"            elementFormDefault="qualified"            target-
Namespace="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd">
  <xs:element name="table">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="row" type="rowType">
        </xs:element>
      </xs:sequence>
    </xs:complexType>
```

```
  </xs:element>
  <xs:complexType name="rowType">
    <xs:sequence>
      <xs:element name="c1" type="xs:decimal"/>
      <xs:element minOccurs="0" name="c2" type="xs:string"/>
      <xs:element name="c3" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

## 5.4 Example of a Table's Primary Data: table0.xml

Primary data are stored in an XML file that complies with the table's XML schema definition.

```
<?xml version="1.0" encoding="utf-8"?>
<table
  xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd table0.xsd"
  xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row><c1>1</c1><c2>First Name</c2><c3>2008-05-09</c3></row>
  <row><c1>2</c1><c2>Second Name</c2><c3>2008-05-10</c3></row>
</table>
```