Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Eidgenössisches Departement des Innern EDI
Schweizerisches Bundesarchiv BAR

Bern, 4. Dezember 2013

SIARD

Formatbeschreibung

Format Version 1.0

Dokumentversion 1.3

Zusammenfassung

Zusammenfassung

Dieses Dokument enthält die Beschreibung des Dateiformats, welches der Anwendung *SIARD Suite* des Schweizerischen Bundesarchivs zugrunde liegt.
SIARD (Software-Independent Archival of Relational Databases) wurde im Rahmen des ARELDA Projektes für digitale Archivierung entwickelt. Es handelt sich um eine normative Beschreibung eines Dateiformats für die langfristige Erhaltung von relationalen Datenbanken.
SIARD ist ein offener Standard. Es wird von der Anwendung *SIARD Suite* unterstützt, mit welcher man relationale Datenbanken (z. B. MS Access, Oracle und SQL Server) in ein SIARD Format konvertieren kann. Das SIARD Format basiert auf Standards – u. a. auf den ISO-Normen Unicode, XML, SQL1999 und dem Industriestandard ZIP. Die Verwendung international anerkannter Standards zielt darauf hin, die langfristige Erhaltung von und den Zugang zu dem weitverbreiteten relationalen Datenbankmodel zu gewährleisten.
Zurzeit wird SIARD im Schweizerischen Bundesarchiv sowie in mehreren Schweizer Bundesämtern eingesetzt. Im Mai 2008 wurde es als das offizielle Format des europäischen PLANETS Projekts für die Archivierung von relationalen Datenbanken akzeptiert.

Inhaltverzeichnis

1 Ausgangslage

Dies ist ein technisches Dokument für IT-Spezialisten, die im Bereich langfristige Archivierung von relationalen Datenbanken tätig sind. Das Dokument basiert auf dem Bundesratsentscheid vom 23. Januar 2008 zum Umgang mit digitalen Unterlagen. Die SIARD Formatbeschreibung stützt sich zudem auf die Archivierungsstrategie des ARELDA Projektes vom 11. April 2006.

Im Rahmen des Projekts ARELDA (ARchivierung ELektronischer DAten) hat das Schweizerische Bundesarchiv (BAR)[1] die Anwendung *SIARD Suite* entwickelt, welche dazu dient, Inhalte relationaler Datenbanken langfristig zu archivieren.

Die Anwendung wurde zunächst als Prototyp entwickelt und liegt inzwischen in einer neuen Version vor: *SIARD Suite*.

Das vorliegende Dokument beschreibt *nicht* die Anwendung sondern das Format, in dem die Datenbanken archiviert werden.

Das offene SIARD-Datenbankformat kann unabhängig von der Anwendung *SIARD Suite* dazu verwendet werden, elektronische Datenbanken langfristig zu archivieren. Wenn Struktur und Inhalt einer Datenbank ins SIARD-Format übersetzt werden, wird es später jederzeit möglich sein, auf die Daten der Datenbank zuzugreifen, selbst wenn die ursprüngliche Datenbanksoftware nicht mehr verfügbar oder nicht mehr lauffähig sein wird. Dies wurde erreicht, indem für das SIARD-Format geeignete Standards verwendet wurden, die international optimal abgestützt sind. Diese langfristige Interpretierbarkeit der Datenbankinhalte beruht im Wesentlichen auf den beiden ISO-Standards XML und SQL:1999.

SIARD wurde im Mai 2008 als das offizielle Format für die Archivierung von relationalen Datenbaken des europäischen Projektes PLANETS[2] angenommen.

Das SIARD Format wurde 2013 als schweizerischer E-Government-Standard eCH-0165[3] publiziert. Dieser Standard ist präziser als das vorliegende Dokument und weicht inhaltlich von der hier vorliegenden Formatbeschreibung nur bezüglich der URI ab, welche die *metadata.xsd* identifiziert. Ein Umstieg auf die eCH-URI ist vom Bundesarchiv vorgesehen, sobald eine Erweiterung des aktuellen Formats eingeführt wird.

---

[1] Bundesarchiv BAR [http://www.bar.admin.ch/]

[2] PLANETS = Preservation and Long-term Access via NETworked Services. Siehe dazu die Webseite von PLANETS: http://www.planets-project.eu/

[3] Der Standard eCH-0165 kann hier heruntergeladen werden:

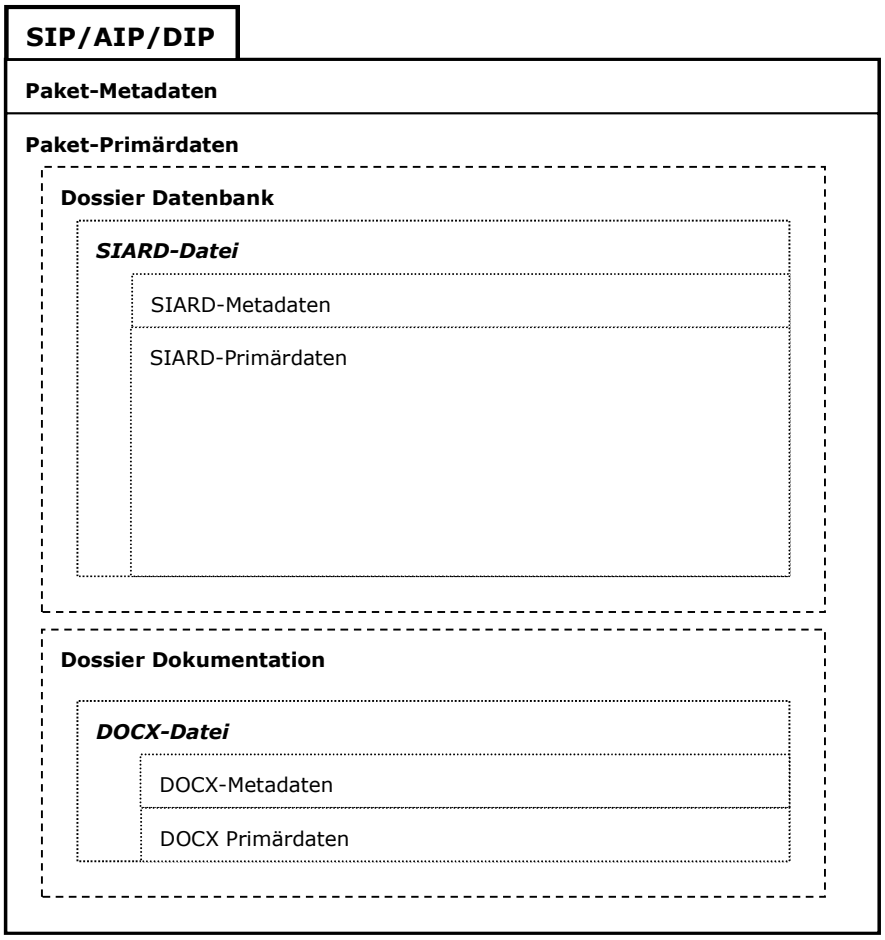
http://www.bar.admin.ch/dienstleistungen/00823/00825/.

2 Abgrenzungen

Es ist festzuhalten, dass das SIARD-Format nur das Langzeitspeicherformat für eine spezielle Sorte von digitalen Unterlagen (relationale Datenbanken) darstellt und somit völlig unabhängig von Paketstrukturen wie SIP (Submission Information Package), AIP (Archival Information Package) und DIP (Dissemination Information Package) des OAIS-Modells[4] konzipiert ist.

Es wird davon ausgegangen, dass eine Datenbank im SIARD-Format als Teil eines Archivpakets zusammen mit anderen Unterlagen (Dokumentation, für das Verständnis der Datenbank relevante Geschäftsunterlagen, ...) archiviert wird.

Ähnlich wie eine moderne Word- oder Mail-Datei eine interne Dateistruktur mit Metadaten, Primärdaten und verschiedenen Hilfsdaten enthält, enthält auch eine archivierte relationale Datenbank neben den eigentlichen Primärdaten auch eigene Metadaten, welche die Unterlage näher beschreiben – ohne Rücksicht auf den Metadatenkatalog, den ein Archiv in seinen OAIS-Paketen erfasst.



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

3 Grundsätze

3.1 Verwendung von Standards

Um die Interpretierbarkeit der Datenbankinhalte über lange Zeiträume zu gewährleisten, beruht das SIARD-Format im Wesentlichen auf den beiden ISO-Standards XML[5] sowie SQL:1999[6].

Sämtliche Datenbankinhalte werden in einer Kollektion von XML-Dateien gespeichert (Schemadefinitionen und SQL-Code jeweils SQL:1999-konform). Einzige Ausnahme sind größere BLOB- und CLOB-Daten (Binary Large OBjects und Character Large OBjects), die in separaten binären Dateien gespeichert aber in den XML-Dateien referenziert werden.

3.2 Datenbanken als Unterlagen

Eine relationale Datenbank wird von SIARD wie eine einzige zu archivierende Unterlage behandelt. Ähnlich wie eine Worddatei oder eine Maildatei kann sie intern aus vielen Teilen bestehen. Normalerweise müssen alle Tabellen in einer relationalen Datenbank zusammen archiviert werden, damit die Bezüge (Referenzen) zwischen den Daten einzelner Tabellen erhalten bleiben.

Um dieser Analogie noch mehr Nachdruck zu verleihen, wird eine im SIARD-Format archivierte Datenbank als eine einzige Datei abgelegt. Diese Datei wird als ein einziges (unkomprimiertes) ZIP[7]-Archiv gespeichert, welches die oben erwähnten XML- und Binärdateien in einer bestimmten Ordnerstruktur beinhaltet. Verschiedene Datenbanken werden in verschiedenen SIARD-Dateien archiviert.

3.3 Zeichensätze und Zeichen

Generell werden alle Daten in einem Unicode-Zeichensatz[8] gespeichert. Beim Extrahieren aus Datenbanken, welche andere Zeichensätze unterstützen, wird die Abbildung in die entsprechenden Unicode-Zeichensätze vorgenommen. Aus diesem Grund übersetzt SIARD die nationalen Zeichenketten-Typen (NCHAR, NVARCHAR, NCLOB) aus dem Datenbank-Produkt generell in nicht-nationale (CHAR, VARCHAR bzw. CLOB).

Diese Konvention wird gut von XML unterstützt, unabhängig davon, ob eine XML-Datei im UTF[9]-8 Format oder im UTF-16-Format gespeichert wird.

---

[5] XML = Extensible Markup Language, erweiterbare Auszeichnungssprache, 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 und

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

[7] ZIP-Dateien wurden ursprünglich von Phil Katz definiert. Heutzutage sind sie sehr weit verbreitet. Microsoft bietet komprimierte ZIP-Ordner als Teil des Betriebssystems Windows an, auch JAVA-Archive (JAR-Files) sind ZIP-Dateien.

Die aktuelle Version 6.3.2 der von der Firma PkWare publizierten Spezifikation findet man unter http://www.pkware.com/documents/casestudies/APPNOTE.TXT.

[8] Unicode = ein internationaler Standard für Zeichensysteme, entspricht der Norm ISO 10646, The Unicode Consortium: The Unicode Standard, Reading MA, Addison Wesley, 2000.

[9] UTF = Unicode Transformation Format

In den XML-Dateien des SIARD-Formats werden gewisse Zeichen, welche in der XML-Syntax eine spezielle Bedeutung haben, durch Einheitenreferenzen ersetzt und zwar in allen Feldern vom Typ *xs:string*.

Zusätzlich werden die Unicode-Steuerzeichen 0-31 und 127-159 mit Hilfe des Solidus („\“) codiert, damit die Gültigkeit der XML-Datei garantiert bleibt.

Zeichenersetzung

Zeichen, die nicht in UNICODE dargestellt werden können (Codes 0-8, 14-31, 127-159) sowie das Escapezeichen '\' und mehrere aufeinanderfolgende Leerschläge werden mit Escape als \u00<xx> im XML dargestellt. Doppelte Anführungszeichen, Kleiner und Et-Zeichen werden in XML als Einheitsreferenzen dargestellt.

Ursprüngliche Zeichen	Zeichen im SIARD- Format
0 bis 8	\u0000 bis \u0008
14-31	\u000E bis \u001F
32	\u0020, falls mehrere aufeinanderfolgen
&	&amp;
<	<
\	\u005c
127 bis 159	\u007F bis \u009F

3.4 Identifiers

In SQL:1999 gibt es reguläre Bezeichner ohne Leerschläge und Sonderzeichen, für welche Gross- und Kleinschreibung unwichtig ist, die aber in Grossbuchstaben gespeichert werden, und Bezeichner in Anführungszeichen, für welche die Schreibweise eindeutig ist, und die auch Sonderzeichen enthalten dürfen. Diese werden in Ausdrücken von doppelten Anführungszeichen umrahmt.

Was ein Sonderzeichen ist, bzw. was die Grossbuchstabenversion eines Buchstabens ist, wird vom Unicode Standard bestimmt.

In den Metadaten wird ein regulärer Bezeichner in Grossbuchstaben gespeichert, während ein delimitierter Bezeichner in Anführungszeichen gespeichert wird. Der SQL:1999-Standard hält fest: Sobald ein Bezeichner ein Zeichen enthält, das er als regulärer Bezeichner nicht enthalten darf, handelt es sich um einen delimitierten Bezeichner.

4 Das SIARD-Format

4.1 Ausgangspunkt relationale Datenbank

Eine „**Datenbank**“ besteht normalerweise aus einem oder mehreren Datenbank-**Schemas**[10] sowie definierten Zugriffsrechten einzelner Benutzer und Rollen auf gewisse Teile der Datenbank. In SQL:1999 können **Benutzer** (Users) und **Rollen** (Roles) Träger von **Berechtigungen** (Privilegien) sein.

Schemas sind Behälter der **Tabellen**, **Views** und **Routinen**.

Tabellen bestehen aus einer Tabellendefinition mit Feldern, die jeder **Spalte** der Tabelle einen Namen und einen Typ zuordnen, aus Datensätzen, welche die eigentlichen **Primärdaten** enthalten, aus einem optionalen **Primärschlüssel**, aus **Fremdschlüsseln**, welche die referenzielle Integrität sicherstellen, aus **Kandidatenschlüsseln**, welche zur Identifizierung eines Datensatzes dienen, und aus **Einschränkungen**, welche die Konsistenz garantieren. Optional können zu einer Tabelle sogenannte **Triggers** (Auslöser) definiert sein.

Views sind in der Datenbank gespeicherte Standardabfragen. Das Abfrageresultat ist eine Tabelle, welche ebenfalls Felder und Datensätze enthält.

SQL-**Routinen** (auch unter der Bezeichnung Stored Procedures bekannt) sind vor allem zum Verständnis der View-Abfragen wichtig, bei welchen sie in Teilausdrücken vorkommen können.

Eine relationale Datenbank besteht somit aus einer Menge strukturierter Datenbankobjekte (z.B. Schema, View etc.) sowie den Tabelleninhalten.

4.2 Struktur der SIARD-Archivdatei

Eine im SIARD-Format archivierte relationale Datenbank besteht ebenfalls aus zwei Komponenten: den Metadaten, welche die Struktur der archivierten Datenbank beschreiben, und den Primärdaten, welche die Tabelleninhalte repräsentieren. Die Metadaten geben weiterhin an, welche Primärdaten wo im Archiv zu finden sind.

Metadaten und Primärdaten einer Datenbank werden gemeinsam in einem unkomprimierten ZIP-Archiv mit der Dateierweiterung „.siard“ abgelegt. Die Primärdaten befinden sich dabei im Ordner *content* und die Metadaten im Ordner *header.*

Die folgende Darstellung soll diese Struktur verdeutlichen:

content
    schema1
        table1
            table.xsd
            table.xml

[10] Ein Datenbankschema ist eine Art Namespace-Präfix. Ein Datenbankkatalog enthält die Metadaten aller Schemas im Katalog. Die Ebene Katalog in SQL:1999 entspricht der „Unterlage Datenbank“, die man mit SIARD in ein Archivformat umwandeln kann.

lob1[11]

record1.txt / record1.bin

lob2

record1.txt / record1.bin

...

table2

table.xsd

table.xml

...

schema2

...

header

metadata.xsd[12]

metadata.xml

Die Verzeichnisse *content* und *header* werden als separate (leere) Einträge *content/* und *header/* in der ZIP-Datei gespeichert. Damit die Integrität der Primärdaten überprüft werden kann, ist es notwendig, dass der Eintrag des *header*-Verzeichnisses erst nach allen Primärdaten und vor allen anderen Metadateneinträgen eingefügt wird. Der unten erwähnte MessageDigest wird von Offset 0 bis zum Offset des *header/* Eintrags berechnet.

Details zu den Formaten der beiden Ordner und der darin enthaltenen Dateien werden im Folgenden beschrieben.

4.3 Metadaten im SIARD-Archiv

Die Metadaten im SIARD-Archiv speichern die Struktur der archivierten Datenbank und geben an, welche Primärdaten wo im Archiv zu finden sind.

Sämtliche Metadaten werden in einer einzigen Datei *metadata.xml* im Ordner *header* versammelt. Die Datei ist analog einer relationale Datenbank hierarchisch aufgebaut.

Die Inhalte der einzelnen Ebenen werden im Folgenden besprochen. Mit „ja“ in der Spalte „opt.“ sind diejenigen Angaben gekennzeichnet, die optional sind, also fehlen dürfen.

4.3.1 Metadaten auf der Ebene Datenbank

Die Datei *metadata.xml* enthält folgende globalen Angaben auf der Ebene Datenbank:

Bezeichnung	*opt.*	*Bedeutung*
version	nein	Format-Version (Version des SIARD-Formats, nicht eines SIARD-Programms!); bis auf Weiteres „1.0“

[11] LOB = Large Objects etwa BLOB = Binary Large OBjects und CLOB = Character Large OBjects, siehe Absatz 4.1

[12] XSD = XML-Schema-Definition

Bezeichnung	*opt.*	*Bedeutung*
dbname	nein	Kurze Bezeichnung der Datenbank
description	ja	Beschreibung der Bedeutung und des Inhalts der Datenbank als Ganzes.
archiver	ja	Name der Person, welche die Archivierung der Primärdaten aus der Datenbank durchführte
archiverContact	ja	Kontaktdaten (Telefon, E-Mail) zur Person, welche die Archivierung der Primärdaten aus der Datenbank durchführte
dataOwner	nein	Eigentümer der Daten in der Datenbank; die Institution oder Person, welche zum Zeitpunkt der Archivierung das Recht besitzt, Lizenzrechte an der Nutzung der Daten zu vergeben und die für die Einhaltung gesetzlicher Auflagen wie Datenschutzrichtlinien verantwortlich ist
dataOrigin-Timespan	nein	Entstehungszeitraum der Daten in der Datenbank; eine ungefähre Zeitangabe als Text
producer-Application	ja	Name und Version der Anwendung, welche die SIARD-Datei heruntergeladen hat.
archivalDate	nein	Archivierungsdatum; Datum der Archivierung der Primärdaten
messageDigest	nein	hexadezimaler Message-Digest-Code über den Ordner *content* mit Präfix, welches den Typ des Digest-Algorithums angibt („MD5[13]“ oder „SHA1[14]“) Der Message-Digest-Code ermöglicht eine schnelle Überprüfung der Integrität der Primärdaten.
clientMachine	ja	DNS[15]-Name des (Client-)Rechners, auf welchem die Archivierung durchgeführt wurde
databaseProduct	ja	Datenbank-Produkt und Version, aus welchem die Archivierung der Primärdaten erfolgte
connection	ja	Verwendeter Connection String für die Archivierung der Primärdaten
databaseUser	ja	Datenbank-UserId des Benutzers des SIARD-Werkzeugs für das Archivieren der Primärdaten aus der Datenbank
schemas	nein	Liste der Schemas in der Datenbank
users	nein	Liste der Datenbank-Benutzer
roles	ja	Liste der Datenbank-Rollen
privileges	ja	Liste der Privilegien für Benutzer und Rollen

[13] MD5 = Message-Digest Algotithmus 5, (Message-Digest Algorithm 5)

[14] SHA = sicherer Hash-Algorithmus (Secure Hash Algorithm)

[15] DNS = Domain Name System, eine verteilte Datenbank, die den Namensraum im Internet verwaltet

4.3.2 Metadaten auf der Ebene Schema

Die Schema-Metadaten werden wie schon die globalen Angaben zur Datenbank in der Datei *metadata.xml* archiviert.

Die folgenden Schema-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Schemaname in der Datenbank
folder	nein	Name des Schemaordners unter *content* im SIARD-Archiv
description	ja	Beschreibung der Bedeutung und des Inhalts des Schemas
tables	nein	Liste der Tabellen in der Datenbank
views	ja	Liste der in der Datenbank gespeicherten Queries (Abfragen)
routines	ja	Liste der Routinen (früher Stored Procedures genannt) im Schema

Die Primärdaten eines jeden Schemas werden im SIARD-Archiv in einem Unterordner von *content* archiviert. Der Schemaname der Datenbank eignet sich nicht als Ordnername, weil die SQL-Syntax Namen zulässt, die in vielen Filesystemen inakzeptabel sind.

Innerhalb des SIARD-Formats sind nur einfachste Namen für Ordner und Dateien mit ASCII - Zeichen (Ziffern und Buchstaben ohne Sonderzeichen) zugelassen, welche garantiert mit allen Filesystemen kompatibel sind. Als Ordnernamen für die Schemas generiert SIARD daher *schema1*, *schema2*, ...

4.3.3 Metadaten auf der Ebene Tabelle

Die Metadaten auf der Ebene Tabelle werden wie schon die globalen Angaben zur Datenbank und die Schema-Metadaten in der Datei *metadata.xml* archiviert.

Die folgenden Tabellen-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Tabellenname im Schema
folder	nein	Name des Tabellenordners im Schemaordner
description	ja	Beschreibung der Bedeutung und des Inhalts der Tabelle
columns	nein	Liste der Spalten der Tabelle
primaryKey	ja	Primärschlüssel der Tabelle
foreignKeys	ja	Liste der Fremdschlüssel der Tabelle
candidateKeys	ja	Liste der Kandidatenschlüssel der Tabelle
checkConstraints	ja	Liste der Einschränkungen der Tabelle
triggers	ja	Liste der Triggers der Tabelle
rows	nein	Anzahl Datensätze

Die Primärdaten einer Tabelle werden im SIARD-Archiv im Ordner *content* und dort in einem Unterordner des Schemas archiviert, zu welchem die Tabelle gehört. SIARD generiert automatisch die Namen *table1*, *table2*, *table3*, ...

4.3.4 Metadaten auf der Ebene Spalte

Die Metadaten auf der Ebene Spalte werden wie schon die globalen Angaben zur Datenbank, die Schema-Metadaten und die Metadaten auf der Ebene Tabelle in der Datei *metadata.xml* archiviert.

Die folgenden Spalten-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Spaltenname in der Tabelle
folder	ja	Name des LOB- Ordners im Tabellenordner
type	nein	SQL:1999-Typ der Spalte
typeOriginal	ja	originaler Spaltentyp
defaultValue	ja	Standardwert der Spalte
nullable	ja	Eintrag nicht erforderlich
description	ja	Beschreibung der Bedeutung und des Inhalts der Spalte

Da die verschiedenen sich SQL-konform nennenden Datenbank-Programme sehr unterschiedliche Datentypen zulassen, wird hier neben dem SQL:1999-Typ auch der *originale* Typ aufgeführt. Für jedes das SIARD-Format unterstützende Datenbank-Programm ist eine Übersetzung der proprietären Typen zu SQL:1999-Typen zu definieren und zu dokumentieren.

Der optionale LOB-Ordnername wird nur für Spalten der Large-Object-Typen (z.B BLOB oder CLOB) benötigt. SIARD generiert automatisch die Namen *lob1*, *lob2*, ...

Die Dateien, welche die Large-Object-Felder repräsentieren, werden in diesen Ordnern angelegt und heissen *record1.txt*, *record2.txt*, bzw. *record1.bin*, *record2.bin* ... Diese werden in der Daten-XML-Datei referenziert.

Alle Spalteneinschränkungen ausser der Nullabilität sind als Tabelleneinschränkungen gespeichert.

4.3.5 Metadaten des Primärschlüssels

Die folgenden Primärschlüssel-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name des Primärschlüssels
column	nein	Liste der Spalten des Primärschlüssels
description	ja	Beschreibung der Bedeutung und des Inhalts des Primärschlüssels

4.3.6 Metadaten der Fremdschlüssel

Die folgenden Fremdschlüssel-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name des Fremdschlüssels
referenced-Schema	nein	Schema der referenzierten Tabelle
referencedTable	nein	Referenzierte Tabelle
reference	nein	Referenz (Liste von Spalten und referenzierten Spalten)
matchType	ja	Matchtyp (FULL, PARTIAL oder SIMPLE)
deleteAction	ja	Löschaktion, z.B.: CASCADE
updateAction	ja	Änderungsaktion, z.B.: SET DEFAULT
description	ja	Beschreibung der Bedeutung und des Inhalts des Fremdschlüssels

Der referenzierte externe Tabellenname kann vom Typ *tabelle* oder *schema.tabelle* sein. Dabei sind delimitierte Bezeichner in Anführungszeichen gesetzt.

Der Matchtyp ist FULL, PARTIAL oder SIMPLE. Die Lösch- und Änderungsaktion enthalten die vom SQL:1999-Standard zugelassenen Aktionen.

4.3.7 Referenz-Metadaten

Die Liste der Referenzen besteht aus folgenden Elementen:

Bezeichnung	*opt.*	*Bedeutung*
column	nein	Name der Spalte
referenced	nein	Name der referenzierten Spalte

4.3.8 Metadaten der Kandidatenschlüssel

Die folgenden Kandidatenschlüssel-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name des Kandidatenschlüssels
column	nein	Liste der Spalten des Kandidatenschlüssels
description	ja	Beschreibung der Bedeutung und des Inhalts des Kandidatenschlüssels

4.3.9 Metadaten der Check-Einschränkung

Die Check-Einschränkung besteht aus einer zu prüfenden Bedingung. Diese ist als Ausdruck vom Typ BOOLEAN (mit Wert *true*, *false* oder *unknown*) in SQL:1999-Syntax angegeben.

Die folgenden Kandidatenschlüssel-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name der Check-Einschränkung
condition	nein	Bedingung der Check-Einschränkung
description	ja	Beschreibung der Bedeutung und des Inhalts der Check-Einschränkung

4.3.10 Metadaten auf der Ebene Trigger

Die folgenden Trigger-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Triggername in der Tabelle
actionTime	nein	BEFORE oder AFTER
triggerEvent	nein	INSERT, DELETE, UPDATE [OF <trigger column list>]
aliasList	ja	<old or new value alias list>
triggeredAction	nein	<triggered action>
description	ja	Beschreibung der Bedeutung und des Inhalts des Triggers

4.3.11 Metadaten auf der Ebene View

Die folgenden View-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name der View im Schema
columns	nein	Liste der Spaltennamen der View
query	ja	SQL:1999-Abfrage, welche die View definiert
queryOriginal	ja	Originale SQL-Abfrage, welche die View definiert
description	ja	Beschreibung der Bedeutung und des Inhalts der View

Da die verschiedenen sich SQL-konform nennenden Datenbank-Programme sehr unterschiedliche Abfrage-Syntax zulassen, wird hier neben der SQL:1999-Abfrage auch die *originale* Abfrage aufgeführt. Für jedes das SIARD-Format unterstützende Datenbank-Programm ist eine Übersetzung der proprietären Abfragesyntax zu SQL:1999-Typen zu definieren und zu dokumentieren.

Die Metadaten der Spalten einer View sind identisch strukturiert wie diejenigen einer Tabelle.

4.3.12 Metadaten auf der Ebene Routine

Die folgenden Routine-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Routinename im Schema
description	ja	Beschreibung der Bedeutung und des Inhalts der Routine
source	ja	originaler Quellcode der Routine (VBA, PL/SQL, JAVA)
body	ja	SQL:1999-konformer Quellcode der Routine
characteristic	ja	Charakteristik der Routine
returnType	ja	Rückgabetyp der Routine (sofern es sich um eine Funktion handelt)
parameters	ja	Liste der Parameter

Da viele Datenbank-Programme über proprietäre Routinen verfügen bei denen keine Transformation in eine Abfrage möglich ist, die streng mit SQL:1999 konform ist, kann hier der originale Quellcode der Routine (z.B. in PL/SQL bei Oracle-Datenbanken, VBA bei MS Access Modulen) archiviert werden.

4.3.13 Metadaten der Parameter

Pro Parameter sind folgende Metadaten in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name des Parameters
mode	nein	Mode des Parameters (IN, OUT oder INOUT)
type	nein	SQL:1999-Typ des Parameters
typeOriginal	ja	originaler Parametertyp
description	ja	Beschreibung der Bedeutung und der Funktion der Routine

Wie bei den Spaltenbeschreibungen kann hier der *originale* – proprietäre – Parametertyp angegeben werden.

4.3.14 Metadaten auf der Ebene des Benutzers

Die folgenden User-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name des Benutzers
description	ja	Beschreibung der Bedeutung und der Funktion des Benutzers

4.3.15 Metadaten auf der Ebene Rolle

Die folgenden Role-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
name	nein	Name der Rolle

Bezeichnung	*opt.*	*Bedeutung*
admin	nein	Administrator der Rolle (Benutzer oder Rolle)
description	ja	Beschreibung der Bedeutung und der Funktion der Rolle

4.3.16 Metadaten auf der Ebene der Privilegien

Die folgenden Privilegien-Metadaten werden in der Datei *metadata.xml* gespeichert:

Bezeichnung	*opt.*	*Bedeutung*
type	nein	eingeräumtes Privileg (z.B. SELECT)
object	ja	Objekt, auf welches das Privileg anzuwenden ist
grantor	nein	Berechtiger, der das Privileg einräumt
grantee	nein	Empfänger des Privilegs (Benutzer oder Rolle)
option	ja	Grant-Option (ADMIN oder GRANT)
description	ja	Beschreibung der Bedeutung und der Funktion des Grants

4.4 Primärdaten im SIARD-Archiv

Wie bereits beschrieben, befinden sich die Primärdaten einer archivierten relationalen Datenbank im Ordner *content* in der Dokument-Root des SIARD-Archivs. Wenn dieser Ordner leer ist (und in den Metadaten kein Digest-Code vorliegt), handelt es sich um ein *leeres* SIARD-Archiv, welches nur Metadaten-Definitionen zur Beschreibung einer Datenbank-Struktur enthält.

Die Primärdaten jeder Tabelle werden im SIARD-Archiv im Ordner *content* und dort in einem Unterordner des Schemas archiviert, zu welchem die Tabelle gehört. SIARD generiert für die Schema-Ordner automatisch die Namen *schema1*, *schema2*, *schema3*, ... und für die Tabellen-Ordner *table1*, *table2*, *table3*, ...

Die Tabellendaten (Primärdaten) sind jeweils in einer XML-Datei namens *table.xml* gespeichert.

Pro Tabelle wird eine XML-Schemadefinition (*table.xsd*) erzeugt, welche das XML-Speicherformat der Primärdaten angibt. Diese Schemadefinition spiegelt die SQL-Schema-Metadaten der Tabelle wieder und gibt an, dass die Tabelle als Sequenz von Zeilen gespeichert wird, welche eine Sequenz von Spalteneinträgen mit verschiedenen XML-Typen enthalten. Der Name des Tabellen-Tags ist *table*, derjenige des Datensatz-Tags ist *row*, das Spalten-Tag heisst *c1, c2, ...* (bzw. *siard:c1* ..., wenn der Namespace ausgeschrieben würde).

Die Werte werden gemäss SQL/XML[16] in die entsprechenden XML-Typen transformiert.

Sollte eine Tabelle Daten der Large-Object-Typen (BLOB, CLOB, ...) enthalten, die mehr als 4000 Zeichen bzw. 2000 Byte groß sind, werden hierfür separate Dateien erzeugt. SIARD generiert automatisch die Ordner-Namen *lob1*, *lob2*, ... für jede betroffene Spalte. Die Dateien, welche die Large-Object-Felder repräsentieren, werden in diesen Ordnern angelegt und heissen *record1.txt*, *record2.txt*, bzw. *record1.bin*, *record2.bin,* ... .

[16] http://www.sqlx.org/

Um zu vermeiden, dass leere Ordner entstehen, werden die Ordner nur angelegt, wenn sie notwenig sind, also Daten beinhalten.

5 Anhang: XML-Schemadefinitionen

5.1 metadata.xsd

Die XML-Schemadefinition *metadata.xsd* definiert die Struktur der Datei *metadata.xml* im Ordner *header*.

```
<?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"
maxOccurs="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"
maxOccurs="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

Eine zum XML-Schema für SIARD konforme Metadatenbeschreibung einer Datenbank sieht beispielsweise so aus:

```
<?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 Beispiel für die XML-Schemadefintion einer Tabelle: table0.xsd

Für jede Tabelle wird von SIARD eine XML-Schemadefinition erzeugt, welche den Spalten die richtigen XML-Datentypen zuordnen.

```
<?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"
attributeFormDefault="unqualified"                              elementFormDefault="qualified"
targetNamespace="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 Beispiel für die Primärdaten einer Tabelle: table0.xml

Die Primärdaten werden in einer XML-Datei gespeichert, die der XML-Schemadefinition der Tabelle genügen.

```
<?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>
```