Fachhochschule Potsdam
University of Applied Sciences
Postfach 60 06 08
14406 Potsdam

Diplomarbeit

OAIS-konforme Archivierung relationaler Datenbanken auf Basis von XML

von: **Andreas Schulz**
Studiengang: Dokumentation
Matr. Nr.: 7922
E-Mail: andreas.schulz@fh-potsdam.de

Erstgutachter: Prof. Dr. phil. Felix Sasaki
Zweitgutachter: Prof. Dr. Günther Neher
Abgabe: Juni 2010, Potsdam

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AIP	- Archival Information Package
DBML	- Database Markup Language
DIP	- Dissemination Information Package
DTD	- Dokument Typ Definition
FLWOR	- for-let-where-order by-return
HP	- Hewlett Packard
METS	- Metadata Encoding & Transmission Standard
OAIS	- Open Archival Information System
PDI	- Preservation Description Information
REMIS	- Preservation Metadata Implementation Strategies
RODA	- Repository of Authentic Digital Objects
SGML	- Standard Generalized Markup Language
SIARD	- Software Independent Archiving of Relational Databases
SIP	- Submission Information Package
SQL	- Structured Query Language
W3C	- World Wide Web Consortium
XML	- Extensible Markup Language
XSD	- XML Schema Definition
XQuery	- Extensible Query Language

Einleitung

„*Datenbanken gehören neben Tabellenkalkulation und Textverarbeitung zu den ersten Computeranwendungen überhaupt.*“[1] In der heutigen Zeit sind Datenbanken kaum wegzudenken, denn in Datenbanken werden wichtige digitale Informationen aufbewahrt und bestimmten Nutzergruppen zugänglich gemacht. Diese Informationen stellen oft wichtige Bestandteile des kulturellen Erbes oder generell wichtige Wissensquellen dar. Die Datenbanken und deren Inhalte müssen für die Nachwelt aufbewahrt werden. Dabei bedarf es spezieller Konzepte und Verfahren, um die digitalen Informationen archivgerecht und langfristig aufbewahren zu können.

Anders als die Wirtschaft, welche Datenbanken meist nur kurzfristig in Form eines Backups archiviert, benötigen Archive speziellere Formen zur langfristigen Sicherung. Die archivierte Datenbank muss revisionssicher und für Nutzer zugänglich sein. Dabei dürfen die inhaltlichen Informationen der archivierten Datenbank nicht verändert werden. Zusätzlich dazu gibt es spezielle Richtlinien für den Umgang mit inhaltlichen Informationen und Metadaten, um eine archivkonforme Weiterverarbeitung zu gewährleisten.

In dieser Diplomarbeit wird die Datenbankarchivierung anhand einer selbst erstellten relationalen Datenbank und nach dem OAIS-Modell (Open Archival Information System) erläutert. Die SQL-Testdatenbank wird durch mehrere Tools nach XML exportiert. Dabei werden die Vor- und Nachteile der jeweiligen Tools beschrieben und auf entstehende Probleme eingegangen. Für die Tests zur Datenbankarchivierung werden die Tools „Altova XML Spy“, „Altova DatabaseSpy“, „PHPMyAdmin“ und die „SQL-Kommandoebene“ verwendet. Die SQL-Abfragen aus der Datenbank werden in XQuery-Abfragen umgewandelt und für den Nutzer mit Hilfe des Tools Saxon zugänglich gemacht.

1 Quelle: Däßler, Rolf: Das Einsteigerseminar MySQL 5. Heidelberg: bhv, 2005. ISBN-10 3-8266-7292-5 S. 22

Als Ziel dieser Diplomarbeit sind folgende Ergebnisse zu erwarten:

- Getestete Toolkette für den Export einer relationalen Datenbank nach XML und somit korrekter Aufbau einer XML-Datei für eine einfache Weiterverarbeitung
- Sicherung der Datenbankabfragen als XQuery-Abfragen
- Benutzerfreundliche Bereitstellung der XQuery-Abfragen
- Vorschlag einer Prozesskette zur Archivierung einer relationalen Datenbank

Diese Diplomarbeit ist in vier Abschnitte unterteilt. Der erste Teil beschäftigt sich mit dem Stand der Forschung zum Thema Datenbankarchivierung und beschreibt einige Konzepte und Ansätze. Im zweiten Teil wird die Testdatenbank Melderegister vorgestellt. Der Aufbau der Datenbank, die Abfragen und die Testdatensätze werden ebenfalls in diesem Abschnitt gezeigt. Der dritte Teil dieser Diplomarbeit ist der praktische Teil. In diesem Abschnitt werden die einzelnen Tools getestet und es wird auf entstehende Probleme während der Tests eingegangen. Zusätzlich dazu werden in diesem Abschnitt der Umgang mit den Abfragen und der XML-Schema Datei behandelt. Der vierte Teil stellt Vorschläge für eine funktionstüchtige Prozesskette vor und offenbart die Punkte, bei denen speziellere Software notwendig ist.

1. Stand der Entwicklung und Forschung

In den folgenden Abschnitten wird auf die digitale Langzeitarchivierung sowie den Forschungsstand zur Archivierung von Datenbanken und dem OAIS-Modell (Open Archival Information System) eingegangen. Die folgenden Unterkapitel werden ausschließlich auf das Vorhaben dieser Diplomarbeit ausgelegt.

1.1. Digitale Langzeitarchivierung

Digitale Langzeitarchivierung fasst alle Methoden zusammen, welche für die langfristige Sicherung digitaler Objekte und retrodigitalisierter Ressourcen relevant sind. Die digitalen Objekte sollen mit Hilfe der Langzeitarchivierung für nachfolgende Generationen erhalten bleiben. Dieser Archivierungsprozess stellt eine Herausforderung für Bibliotheken, Archive, Museen, Forschungseinrichtungen und Wirtschaftsunternehmen dar. „*Ohne geeignete Strategien wird es zu Datenverlusten kommen, die weitreichende Folgen für alle wirtschaftlichen und wissenschaftlichen, gesellschaftlichen und politischen Bereiche haben werden.*“[2] Demnach ist es wichtig, Datenbanken - welche selbst digitale Objekte darstellen - langfristig zu sichern.

In der heutigen Zeit werden enorme Mengen an Daten und Informationen digital hergestellt und weiterverarbeitet. Von diesen digitalen Objekten gibt es oft keine analogen Fassungen, sodass man nur auf die digitalen Objekte zugreifen kann. Die digital erschaffenen Informationen haben eine hohe Bedeutung für die Informations- und Wissensgesellschaft. Daher ist es notwendig, diese Informationen für lange Zeit zugänglich zu machen. Die digitalen Informationen bestehen aus Bits und Bytes und sind eng an Hard- und Software gebunden. Nur die korrekte Hard- bzw. Softwareumgebung kann die Informationen wieder in einen für den Menschen lesbaren Zustand übersetzen. Die notwendigen Hard- bzw. Softwareumgebungen werden ständig weiterentwickelt und verbessert. Das führt dazu, dass vorhandene Technik schneller altert oder ersetzt wird.

2 Quelle: nestor zur digitalen Langzeitarchivierung: http://www.langzeitarchivierung.de/ueber_uns/lza.htm

Somit ist es wahrscheinlich, dass benötigte Software und Hardware zur Interpretation der gespeicherten Daten fehlt und dass die Gesellschaft wichtige digitale Informationen verliert.

"*Die dauerhafte Speicherung und Bereitstellung ist technisch machbar, aber aufwändig. Planung und permanente Pflege sind notwendig, um auf lange Sicht den Zugriff auf digitale Daten zu sichern.*"[3]

Das deutsche Kompetenznetzwerk "nestor"[4] befasst sich seit längerem mit der Langzeitarchivierung von digitalen Objekten und meint, dass Langzeitarchivierung für Einzelinstitutionen praktikabel wird, wenn "[...] *sie auf gemeinsames Wissen, bewährte Strategien und erfolgreich getestete Tools zurückgreifen kann.*"[5] Im Rahmen dieser Diplomarbeit soll eine getestete Toolkette zur Archivierung von Datenbanken entstehen. Mit Hilfe der so entstehenden Daten, welche am OAIS-Modell angepasst werden, soll es möglich sein, eine vereinfachte Pflege der Daten vorzunehmen und dem Langzeitarchivierungsprozess zu erleichtern. Digitale Langzeitarchivierung ist nicht "*die Abgabe einer Garantieerklärung über fünf oder fünfzig Jahre, sondern die verantwortliche Entwicklung von Strategien, die den beständigen, vom Informationsmarkt verursachten Wandel bewältigen können.*"[6] Für die Langzeitarchivierung von Datenbanken bedeutet das, dass eine Strategie entwickelt werden muss, welche es ermöglicht, die in der Datenbank enthaltenen Informationen zu erhalten und für spätere Weiterverarbeitungen kompatibel zu sein.

3 Quelle: nestor zur digitalen Langzeitarchivierung: http://www.langzeitarchivierung.de/ueber_uns/lza.htm

4 Link zur Webseite von nestor: http://www.langzeitarchivierung.de/index.htm

5 Quelle: nestor zur digitalen Langzeitarchivierung: http://www.langzeitarchivierung.de/ueber_uns/lza.htm

6 Quelle: Schwens; Liegmann: Langzeitarchivierung digitaler Ressourcen, 2004, S. 567

1.1.1. Stand der Langzeitarchivierung von Datenbanken

„*Die Datenbankarchivierung hat zum Ziel, die Datenbankinhalte in einer systemunabhängigen und lesbaren Form unter Wahrung der Authentizität und Integrität dauerhaft zu erhalten.*“[7] Damit befassen sich seit längerer Zeit unterschiedliche Institutionen und Unternehmen. In diesem Abschnitt wird auf einige bereits vorhandene Produkte zum Thema Datenbankarchivierung und auf Forschungsergebnisse eingegangen.

1.1.1.1. CSP CHRONOS von ikeep

Es gibt bereits einige kommerzielle Werkzeuge wie „CHRONOS“ von der „ikeep AG“.[8] Auf der Herstellerwebseite heißt es: „*CHRONOS ermöglicht die kontinuierliche und gesetzeskonforme Archivierung relationaler Daten- banken. Archivdaten bleiben dauerhaft verfügbar, auch wenn die produktive Datenbank-Umgebung geändert wird (z.B. Schemaänderungen, neue Produktversionen) oder nicht mehr vorhanden ist.*"[9] Doch scheint CHRONOS eher Datenbankbackups zu erstellen, welche die Arbeitsleistung der Server und Datenbanken kurzzeitig entlasten sollen. Auch ist auf der Herstellerseite nichts von Archivstandards, wie dem OAIS-Modell zu finden. Daher kann man davon ausgehen, das CHRONOS nicht die beste Lösung zur Langzeitarchivierung von Datenbanken darstellt. Unter anderem wird von einer "*CHRONOS-Skriptsprache*" berichtet. Durch diese Skriptsprache entsteht eine Abhängigkeit der archivierten Daten zur Software. Möglicherweise sind die Daten ohne CHRONOS beschränkt brauchbar oder gar gänzlich unbrauchbar. Für die korrekte archivische Verwendung der Daten nach dem OAIS-Modell müsste CHRONOS die Metadaten von den inhaltlichen Informationen trennen, damit diese gesondert weiterverarbeitet bzw. archiviert werden können.

7 Quelle: Schwarz; Däßler: Archivierung und dauerhafte Nutzung von Datenbankinhalten aus Fachverfahren – eine neue Herausforderung für die digitale Archivierung, 2010-unveröffentlicht, S. 6

8 Link zur Webseite von ikeep: http://ikeep.com/index.pc

9 Quelle: CHRONOS Funktionalität: http://ikeep.com/dba_chronos.pc

Besonders problematisch ist zudem die ausschließliche Nutzung der Server von CHRONOS. Damit haben Archive keinen direkten lokalen Zugriff auf die archivierten Daten. Für eine OAIS konforme Archivierung ist CHRONOS unbrauchbar.

1.1.1.2. Software Independent Archiving of Relational Databases (SIARD)

„*SIARD [...] 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.*“[10] SIARD wird bereits im schweizerischen Bundesarchiv verwendet und basiert auf international anerkannten Standards wie XML, Unicode, SQL und dem komprimierenden Dateiformat ZIP. In der SIARD Formatbeschreibung heißt es: "*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 [...], AIP [...] und DIP [...] des OAIS-Modells konzipiert ist.*"[11] Allerdings trennt SIARD die inhaltlichen Informationen (Primärdaten) von Metadaten, funktioniert aber nicht - wie oben erwähnt - nach dem OAIS-Modell. Am Ende der Prozesskette zur Archivierung der Datenbank mit der SIARD Suite entsteht eine einzelne Datei, die in Form eines unkomprimierten ZIP Archivs gespeichert wird. Die ursprüngliche relationale Datenbank wird im XML-Format abgelegt, dabei werden die Metadaten in einer eigenen XML-Datei gespeichert. SIARD zeichnet sich zusätzlich durch eine kostenfreie zur Verfügungsstellung vom schweizerischen Bundesarchiv aus.[12] Bis auf die fehlende Nähe zum OAIS-Modell ist SIARD ein sehr erfolgversprechendes Tool zur Langzeitarchivierung von Datenbanken.

10 Quelle: Referenzen - SIARD Formatbeschreibung: http://www.kost-ceco.ch/wiki/whelp/KaD/pages/SIARD.html ; Zusammenfassung S. 1

11 Quelle: Referenzen - SIARD Formatbeschreibung: http://www.kost-ceco.ch/wiki/whelp/KaD/pages/SIARD.html ; Abgrenzung S. 4

12 Link zur kostenfreien SIARD Suite: http://www.bar.admin.ch/dienstleistungen/00823/00825/index.html?lang=de

1.1.1.3. Repository of Authentic Digital Objects (RODA)

Bei diesem Projekt handelt es sich um ein Gemeinschaftsprojekt zwischen der „*University of Minho*“ und dem „*Portuguese National Archives*“. Im Rahmen dieser Kooperation entstand die Datenbank-Auszeichnungssprache „*DBML (database markup language)*“.[13] Diese Auszeichnungssprache wurde für die Migration von Datenbanken nach XML entwickelt. Dabei wird bzw. werden die Datenbankstruktur und die Datenbankinhalte mit einem einfachen Aufbau beschrieben:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<DB name="XXX" date="today">
<STRUCTURE>
...
</STRUCTURE>
<DATA>
...
</DATA>
</DB>
```

Innerhalb des Elementes <STRUCTURE> wird die Struktur der Datenbank beschrieben. Im Element <DATA> hingegen werden die Inhalte der Datenbank dargestellt. DBML geht auch auf die Verwendung von Fremdschlüsseln ein. Die Fremdschlüssel befinden sich dabei im Element <STRUCTURE>. Somit geht die referenzielle Integrität nicht verloren. Ein Beispiel für DBML, um die Datenbankstruktur abzubilden, befindet sich in Anhang A1. DBML eignet sich hervorragend, um Feldbezeichnungen als Attribute in die XML-Dateien einzubinden und um die Datenbankstruktur zu übernehmen.
Der Nachteil an DBML ist, dass XML-Dokumente sowie XML-Schema Dateien manuell erstellt werden müssen. Bei größeren Datenbanken wird der Aufwand somit enorm hoch.

13 Quelle: Relational Database Preservation through XML modelling: http://repositorium.sdum.uminho.pt/bitstream/1822/7120/1/EML2007-final.pdf

1.1.1.4. Hewlett-Packard Database Archiving Software

Die Firma Hewlett Packard bietet mit ihrem Produkt „*Database Archiving Software*“ ein Tool zur Archivierung von Datenbanken mit folgenden Schlüsselbesonderheiten:

- "*Accelerate application performance and improve stability for critical business processes*
- *Provide timely access to information by archiving data according to data management policies*
- *Control administrative and infrastructure costs while meeting governance and compliance archiving requirements*
- *Streamline application upgrades to rapidly enable new business capabilities*
- *Reduce costs and risks of legacy system support through data and application retirement*" [14]

Anhand dieser Besonderheiten ist erkennbar, dass HP sein Produkt speziell für Unternehmen entwickelt hat, die unter anderem von Performance Beschleunigungen und zeitlichen Zugriffen profitieren. Auf der Webseite des Produktes wird angegeben, dass die Software Daten über einen längeren Zeitraum hinweg archivieren kann. „*[...] or to standards-based XML documents for long-term retention.*“[15] HP unterstützt demnach XML als Format für die Langzeitarchivierung von Datenbanken. HP kombiniert „*database-to-database and database-to-XML archiving*", dabei werden die Daten der Datenbank zu einer lauffähigen Onlinedatenbank oder/und zu einer XML-Dateien migriert. Somit können SQL-Abfragen unverändert weiterverwendet werden. Eine Übersicht zur Funktionsweise der Software stellt Abbildung 1 dar.

14 Quelle: Hewlett Packard Database Archiving Software: key features: http://h71028.www7.hp.com/enterprise/w1/en/software/information-management-governance-ediscovery-database-archiving.html

15 Quelle: Hewlett Packard Database Archiving Software: Data sheet: http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA1-9833EEW.pdf

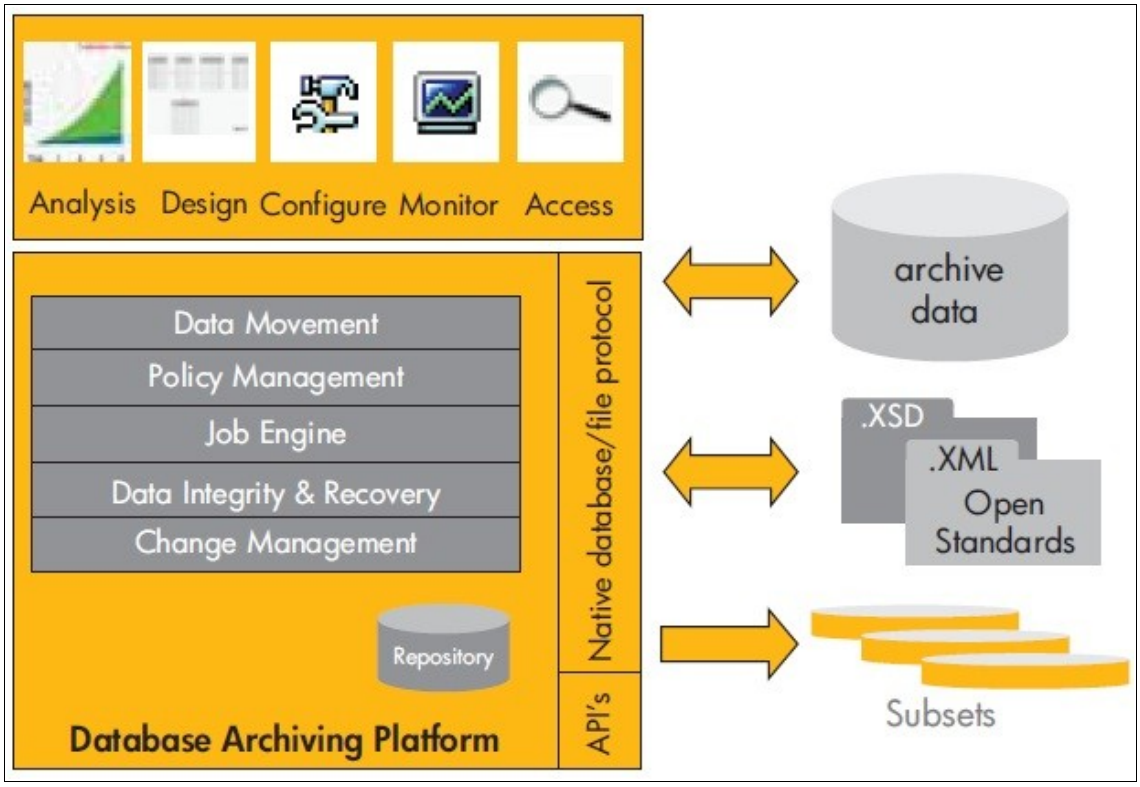


Abbildung 1: HP Database Archiving Software: Funktionsweise

Wie in Abbildung 1 dargestellt, bleibt die Datenbank lauffähig und kann weiter mit Primärdaten ergänzt werden. Die ergänzten Primärdaten werden vermutlich in den „*Subsets*“ gespeichert. Die Software bietet noch eine ganze Reihe weiterer Features, wie zum Beispiel die Unterstützung von Oracle-Datenbanken.

HP bietet damit eine vielseitige Archivierungssoftware für Unternehmen an. Allerdings gleicht der Ansatz der Software nicht dem OAIS-Modell oder lässt sich dafür verwenden. Einzig die Migrations-Methodik der Datenbanken nach XML ist möglicherweise für eine OAIS konforme Datenbankarchivierung brauchbar.

1.1.2. Das OAIS-Modell

Das „*Reference Model for an Open Archival Information System (OAIS)*" wurde 2003 in das ISO Normenwerk mit der Nummer ISO 14721:2003 aufgenommen.[16] Das vom CCDSD ("*Consultative Committee for Space Data Systems*") entwickelte OAIS-Modell *„[...] hat sich als Referenzmodell für die digitale Archivierung bei Bibliotheken und Archiven weltweit durchgesetzt.*"[17] Im OAIS-Modell wird beschrieben, wie Informationen erhalten und für einen bestimmten Nutzerkreis organisiert sowie zugänglich gemacht werden.

Wie die Informationen in das Archivsystem gelangen, welche Bearbeitungsschritte für die langfristige Archivierung vorgenommen werden müssen und wie auf die im Archiv gespeicherten Information zugegriffen werden kann, wird detailliert im OAIS-Modell beschrieben.

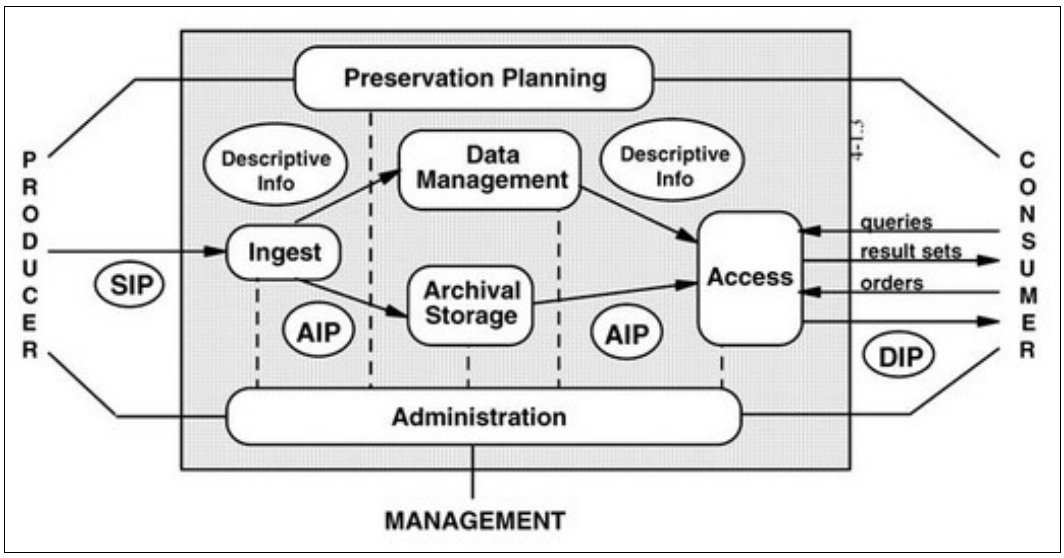


Abbildung 2: Datenflussmodell des OAIS-Modells; Einteilung in Paketen

16 Quelle: International Organization for Standardization: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24683

17 Quelle: Schweizerische Eidgenossenschaft zum OAIS-Modell: http://www.nb.admin.ch/nb_professionnel/01693/01696/01876/01878/index.html?lang=de

Auf dem Konzept des Informationspaketes aufgebaut, besteht das OAIS-Modell aus drei Pakettypen:

1. - Das SIP (Einlieferungspaket)
2. - Das AIP (Archivpaket)
3. - Das DIP (Auslieferungspaket)

Jedes dieser drei Pakettypen ist logisch voneinander abgegrenzt und wird für unterschiedliche Verwendungszwecke im digitalen Archiv genutzt. Neben der logischen Abgrenzung der einzelnen Pakete besteht jedes einzelne Paket aus entsprechenden Metadaten und Primärdaten. Die einzelnen Informationspakete sind wie folgt aufgebaut: [18]

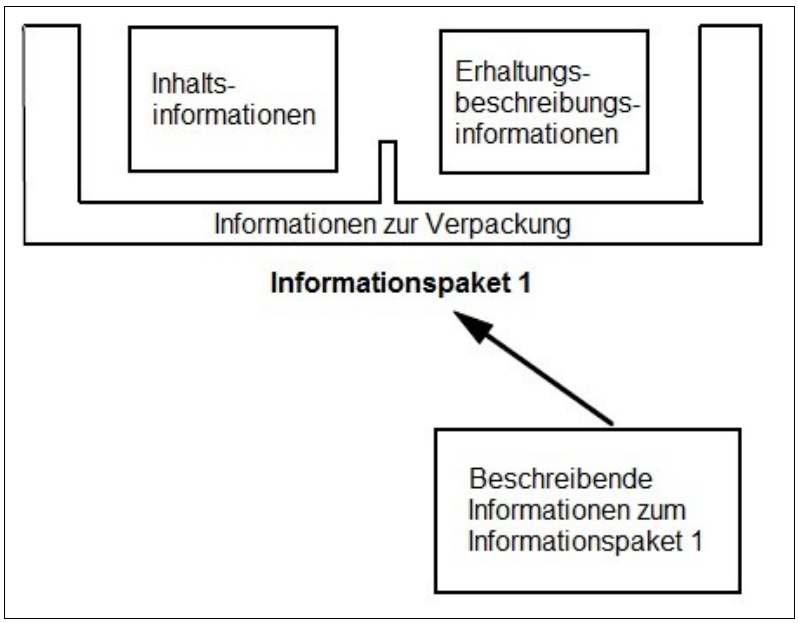


Abbildung 3: Aufbau eines Informationspaketes

Die originalen Inhaltsinformationen sind das Zielobjekt für die Bestandserhaltung. Die Erhaltungsbeschreibungsinformationen[19] beschreiben die Aufbewahrungsform der digitalen Objekte und machen die Informationen für die vorgesehene Nutzergruppe verständlich.

18 Quelle der Grafik: http://public.ccsds.org/publications/archive/650x0b1.pdf Übernahme und Übersetzung der Grafik von Seite 2-5

19 Übersetzung aus dem englischen "preservation description information" (PDI)

Erhaltungsbeschreibungsinformationen sind in vier Typen von Erhaltungsinformationen eingeteilt:

- Provenienz: Beschreibt die Herkunft der Inhaltsinformationen und deren Entstehungs- und Entwicklungsprozess.

- Kontextinformationen: Beschreibt in welchem Zusammenhang die Inhaltsinformationen zu anderen Informationen, außerhalb des Informationspaketes, stehen.

- Aktenzeichen: Ist für die eindeutige Identifizierung der Inhaltsinformation relevant.

- Beständigkeit: Ist ein schützender Schild, welcher die Inhaltsinformationen vor undokumentierten Änderungen schützen soll. [20]

Die Informationen zur Verpackung identifizieren und verbinden die einzelnen Paketkomponenten. Die beschreibenden Informationen enthalten die Informationen zur Auffindbarkeit des Informationspaketes und machen eine spätere Recherche im Archiv möglich.

1.1.2.1. Einlieferungspaket (SIP)

Das „*Submission Information Package*“ (SIP) oder auch Einlieferungspaket ist das Paket, welches vom Produzenten des Objektes an das OAIS-Modell gesendet wird. Im SIP befinden sich Primär- und Metadaten. Die Metadaten bestehen hauptsächlich aus Informationen, die die Inhalte beschreiben und für eine spätere Interpretation relevant sind. Ein einzelnes SIP kann in mehreren AIPs enthalten sein.

20 Quelle: http://public.ccsds.org/publications/archive/650x0b1.pdf Seite 2-6

1.1.2.2. Archivpaket (AIP)

Das „*Archival Information Packages*" (AIP) wird aus einem oder mehreren SIPs generiert. Dadurch entstehen ein oder mehrere AIPs für den Bestandserhalt. Das AIP enthält einen kompletten Satz aus Erhaltungsbeschreibungsinformationen, welche mit den Inhaltsinformationen verbunden sind. Für die langfristige Aufbewahrung der digitalen Informationen ist das AIP angepasst worden. Die Verpackungsinformationen sind für das AIP besonders relevant, da die beschreibenden Informationen eine langfristige Aufbewahrung und Interpretierbarkeit der Inhalte gewährleisten. Dementsprechend muss die Beschreibung der Aufbewahrungsform so vollständig und ausführlich wie möglich vorliegen.

1.1.2.3. Auslieferungspaket (DIP)

Die Suchanfragen der Nutzer werden im OAIS-Modell mittels eines „*Dissemination Information Package*“ (DIP) oder auch Auslieferungspaketes beantwortet. Das DIP beinhaltet Sammlungen von AIPs und stellt die Form dar, in der das AIP wieder zugänglich gemacht wird. Das Auslieferungspaket sollte für die unterschiedlichen Nutzergruppen besonders komfortabel und dementsprechend einfach anzupassen sein. Das DIP wird aus dem AIP generiert. Für den Nutzer sind besonders die Inhaltsinformationen relevant, daher müssen Verpackungs- und beschreibende Informationen nicht vollständig vorliegen.

1.1.2.4. Das OAIS-Funktionsmodell

Das OAIS-Modell funktioniert wie in Abbildung 2 dargestellt. Dabei beinhaltet es folgende Funktionseinheiten:

- „Ingest“/Übernahme: Das zu archivierende Objekt wird nach vereinbarten Kriterien, welche zwischen dem Archiv und dem Produzenten vereinbart werden, auf die Übernahme in das OIAS-Modell vorbereitet. In dieser Funktionseinheit laufen alle weiteren Prozesse ab, die für die Generierung des AIPs relevant sind. Dazu zählen unter anderem die Erfassung von Metadaten und die Anpassung von Primärdaten-Formate an das Archivsystem.

- „Archival Storage“/Aufbewahrung: Diese Funktionseinheit beinhaltet alle Dienste und Funktionen zur Einlagerung des AIPs. Die Funktionen der Aufbewahrung übernehmen das AIP, welches aus der Übernahme generiert wurde. Das nun übernommene AIP wird in den permanenten Archivspeicher übergeben, dabei wird eine Speicherhierarchie aufgebaut, welche das regelmäßige Warten der AIPs ermöglicht. Bei der Wartung wird ebenfalls die Lesbarkeit des archivierten Objektes geprüft.

- „Data Management“/Datenverwaltung: In dieser Einheit werden alle Prozesse zur Datenverwaltung und Archivverwaltung durchgeführt. Des Weiteren werden Updates, Datenbanksichten und Schema-Dateien verarbeitet. Die Nutzeranfragen werden durchgeführt sowie Berichte und Ergebnisse generiert.

- „Administration“/Verwaltung: Das gesamte Archivsystem wird in dieser Einheit geregelt. In der Verwaltung werden alle mit dem Produzenten vereinbarten Verträge und Verbindlichkeiten verarbeitet und abgespeichert. Hard- und Softwareprozesse werden überwacht und Arbeitsabläufe aufgezeichnet und optimiert.

Zusätzlich dazu werden die Fragen der Kunden in der Kundenbetreuung bearbeitet und die Gesetze und Standards zu den Daten abgelegt.

– „Preservation Planing“/Planung der Langzeitarchivierung: Diese Einheit befasst sich mit der Beobachtung der technischen Umgebung und den Wünschen der Nutzer. Dabei werden Anwendungsempfehlungen erstellt und Archivinhalte evaluiert, um neue Standards und Richtlinien zu entwickeln. Bei der Planung der Langzeitarchivierung werden Pläne zur Migration, Software Prototypen und Testpläne ausgearbeitet.

– „Access“/Bereitstellung bzw. Zugriff: Die Bereitstellung ermöglicht dem Nutzer den Zugriff auf die archivierten Daten mit Hilfe von Abfragen. Die Ausgabe der Informationen erfolgt über ein koordiniertes Suchverfahren und in Abhängigkeit zur Zugriffsberechtigung des Nutzers.[21]

Diese sechs Funktionseinheiten bilden das Datenflussmodell vom OAIS-Modell.

1.2. XML und Datenbanken

Für diese Diplomarbeit ist es enorm wichtig, vorab zu klären, ob sich XML in Datenbankform ablegen lässt, um es später sinnvoll weiterverarbeiten zu können. Dabei liegt der Fokus in dieser Arbeit bei den relationalen Datenbanken. „*Relationale Datenbankinhalte als XML-Dokumente darzustellen, erscheint zunächst relativ einfach, da hierbei lediglich die Aufgabe besteht, strukturierte Informationen in einer semistrukturierten Form auszugeben, die in einigen Aspekten mehr Darstellungsmöglichkeiten bietet.*“[22] Prinzipiell ist es somit möglich, eine relationale Datenbank in XML darzustellen. Es gibt sogar mehrere verschiedene Möglichkeiten, eine relationale Datenbank nach XML zu exportieren, dabei kommt es auf das Ziel der Datenrepräsentation in XML an.

21 Quelle: CCSDS: Reference Model for an Open Arcival Information System (OAIS), Januar 2020, S. 4-1 bis 4-2 http://public.ccsds.org/publications/archive/650x0b1.pdf

22 Quelle: Klettke, Meike; Meyer, Holger: XML & Datenbanken: dpunkt.verlag. 1. Auflage Heidelberg, 2003, ISBN 3-89864-148-1, S.58

Diese drei Anwendungsmöglichkeiten gibt es:

- **Vollständige Abbildung der Datenbankinhalte**, dabei wird der komplette Datenbankinhalt in das XML-Dokument übernommen. Sämtliche Inhaltsinformation müssen erhalten werden.
- **Abbildung von Abfrageergebnissen oder Views**, durch eine Sicht oder eine Abfrage auf die Datenbank werden nur bestimmte Informationen in das XML-Dokument übernommen.
- **Einsatz individueller Transformationsregeln**, dabei wird ein bestimmter Teil der Inhaltsinformationen, möglicherweise auch der gesamte Teil, in einer vordefinieren Struktur übernommen. So verändert sich nicht der Inhalt, aber die Struktur der Daten.

Bei allen drei Möglichkeiten gilt, dass jeweils die Datenbankschemainformationen und die Inhaltsinformationen nach XML transformiert werden müssen. Für die Schemainformationen gibt es spezielle XML-Formate, wie die Dokumenttypdefinition (DTD) und die XML-Schema-Definition (XSD). Die Bezeichner der Inhaltsinformationen müssen auf Element- und Attributnamen von XML-Dokumenten abgebildet werden.

Folgende Struktur würde eine Datenbank in XML abbilden können:

```
<database>
    <table>
        <row>
            <column1>...</column1>
            <column2>...</column2>
            ...
        </row>
    </table>
</database>
```
[23]

23 Quelle: http://www.rpbourret.com/xml/XMLAndDatabases.htm Abschnitt 5.1.1 Table-Based Mapping - gekürzt

Das Element <database> kommt für jede Datenbank nur einmal vor. Die Elemente <table>, <row> und <column> können beliebig oft vorkommen, um die relationale Datenbank in XML abzubilden.
Für diese Diplomarbeit wird eine vollständige Abbildung der Datenbankinhalte aus einer Testdatenbank angestrebt.

1.3. XML als Archivformat

XML ist, wie im Abschnitt 1.2. beschrieben, für die Übernahme der Daten aus relationalen Datenbanken geeignet. Ein großer Vorteil von XML besteht darin, dass es ein offener und lizenzfreier Standard des „*World Wide Web Consortium (W3C)*“ ist. Denn dieser Standard wurde für den Austausch von Daten im Internet aus der Auszeichnungssprache SGML („*Standard Generalized Markup Language*“) entwickelt. Für den Datenaustausch via Internet sind viele Entwicklungen entstanden, welche sich auch für die Langzeitarchivierung von Datenbanken nutzen lassen.
Ein Beispiel dafür ist die Validierung der Daten gegen eine Schema Datei, um Fehler zu verhindern. Dabei können vor allem fest definierte Datentypen zu keiner Abweichung führen. XML ist zudem sehr übersichtlich und gut strukturiert, was dazu führt, dass es bis zu einem gewissen Maß für den Menschen lesbar bleibt. Die hierarchische Struktur eines XML-Dokumentes lässt sich gut mit der Strukturierung von relationalen Datenbanken vergleichen. XML ist zudem textbasiert und erkennt durch die Verwendung von Unicode etliche Sonderzeichen und kann somit von vielen Textbearbeitungsprogrammen erstellt und gelesen werden. Das macht XML unabhängig zu Hard- und Software. Die weit verbreitete Verwendung und Anerkennung von XML hat viele Neuerungen für XML hervorgebracht. So zum Beispiel die Abfragesprache XQuery („*Extensible Query Language*“).

Vorteile	Nachteile
Geeignet für die Übernahme der Daten aus relationalen Datenbanken	XML-Dokumente können sehr komplex werden
Offener und lizenzfreier Standard des W3C	
Übersichtliche Strukturierung	
Für den Menschen lesbar und interpretierbar	
Hard- und Softwareunabhänig	
Große Verbreitung und Anerkennung	
Validierungsmöglichkeiten durch Schema-Dateien	
Möglichkeit für Abfragesprachen vorhanden	
Einfache Weiterverarbeitung möglich	

Diese Tabelle zeigt noch einmal alle Vor- und Nachteile von XML zur Langzeitarchivierung. XML ist demnach ein gutes Archivformat.

1.4. Validierung von XML-Dokumenten

XML-Dokumente können gegen eigens dafür entwickelte Schema Dateien validiert werden. Mit Hilfe von DTDs und XSDs lassen sich Strukturen und Datentypen definieren, um eine fehlerhafte Dateneingabe zu verhindern und um zu überprüfen, ob das XML-Dokument wohlgeformt ist. Somit werden Regeln erstellt, die auf unterschiedliche Weise mit dem XML-Dokument verknüpft sind:

- Interne DTD: Diese Variante ist besonders für kleinere XML-Dokumente zu empfehlen, die nicht für den späteren Datenaustausch oder einer Weiterverbreitung vorgesehen sind. Die Regeln werden direkt in das XML-Dokument eingebunden.

- Externe DTD: Die externe DTD ist besonders für die Weiterverbreitung des XML-Dokumentes geeignet. Beispielsweise beim Aufbau eines gemeinsamen Datenbestandes findet die externe DTD ein großes Anwendungsfeld.

Dabei können an unterschiedlichen Standorten XML-Dokumente mit den vordefinierten Regeln aus den externen DTDs erstellt werden. Die DTD wird mit Hilfe eines Links in die betreffenden XML-Dokumente eingefügt.

– XSD: Die XML-Schema Datei ist wie die externe DTD nicht in das XML-Dokument einzubinden. Die Einbindung findet aber mit Hilfe eines Links statt. Im Gegensatz zur DTD können die Regeln bei der XSD viel exakter definiert werden. Bei einer XML-Schema Datei ist es möglich, auf 44 verschiedene Datentypen zuzugreifen. Ebenfalls ist es machbar, die Reihenfolge der Elemente, die in einem XML-Dokument vorkommen können bzw. sollen, festzulegen.

Diese drei Möglichkeiten sind auch für die Langzeitarchivierung von Datenbanken interessant. Mit Hilfe der DTD oder XSD wäre es realisierbar, die Struktur der Datenbank zu speichern. Das jeweilige Dokument könnte als Speicherdatei für Metadaten dienen, da die vordefinierten Datentypen schon in der Schema Datei integriert sind.
Neben der DTD und der XSD gibt es noch spezielle Schemasprachen zur Validierung von Struktur und Inhalt einer XML-Datei. Das unter der ISO Nummer 19757-3:2006 bekannte „*Schematron*“[24] ist eine solche Schemasprache. Schematron dient nur der Validierung, anders als bei DTD oder XSD werden unter Schematron keine Regeln festgelegt. Diese Schemasprache ersetzt nicht XSD oder DTD, sondern wird sehr oft als Beigabe bzw. Ergänzung verwendet. Schematron ist sehr einfach und besteht aus nur fünf wichtigen Elementen und kann somit XML-Schema Dateien erweitern, um bestimmte Sachverhalte, die XSD nicht darstellen kann, darzustellen.
Möglicherweise ist Schematron für die spätere Validierung des XML-Dokumentes nützlich.

24 Link zur Webseite von Schematron: http://www.schematron.com/

1.5. Metadaten in XML

Metadaten sind strukturierte Daten, die Informationen über andere Daten beinhalten bzw. andere Daten beschreiben. Metadaten werden besonders bei größeren Datenmengen verwendet. Dabei sind Metadaten maschinell lesbar und auswertbar, um eine Verarbeitung der Informationsmengen zu vereinfachen.
Es gibt Formate, die es erlauben, Metadaten in XML-Dokumenten abzubilden. PREMIS[25] und METS[26] sind XML-Formate, welche Vorschläge für die Implementierung von Metadaten für die Langzeitarchivierung geben. Beide Formate basieren auf einer XML-Schema Datei, welche die Metadateneingabe vereinfacht und standardisiert. Durch das Validieren der XML-Datei gegen die Schema-Datei können Fehler vermieden werden.
PREMIS und METS eignen sich demnach gut für die Speicherung von Metadaten.

1.6. Nutzen von XQuery bei der Datenbankarchivierung und bei der Wiederherstellung der referenziellen Integrität

„*Der Umgang mit XML-Datenbanken und insbesondere die Kenntnis von XQuery wird in Zukunft den gleichen Rang wie SQL im Kontext relationaler Datenbanken einnehmen und damit fundamental sein.*“[27] Somit ist XQuery für diese Diplomarbeit sehr passend.

Die Abfragesprache XQuery kann in dieser Diplomarbeit mehrere Funktionen erfüllen. Sobald die Daten aus der Datenbank in ein XML-Dokument exportiert wurden, kann XQuery für die weitere Verarbeitung eingesetzt werden. Außerdem können mit XQuery Abfragen erstellt werden, die die referenzielle Integrität der Datenbank erhalten sollen.

25 Link zur Webseite von PREMIS: http://www.loc.gov/standards/premis/

26 Link zur Webseite von METS: http://www.loc.gov/standards/mets/

27 Quelle: Lehner, Wolfgang; Schöningen, Harald: Xquery – Grundlagen und fortgeschrittene Methoden: dpunkt.verlag 1. Auflage Heidelberg, 2004, ISBN 3-89864-266-6, S. 2

Beim Export der Datenbank kann es durchaus passieren, dass Fremdschlüsseldefinitionen verloren gehen.
Diese müssen manuell nachgetragen werden oder aber mit Abfragen wiederhergestellt werden. Und für diese Aufgabe wird XQuery verwendet. Die Abfragen oder auch „views“ werden auch nach dem OAIS-Modell in dem Auslieferungspaket benötigt.
Mit XQuery hat der Nutzer die Möglichkeit, aus vordefinierten Abfragen die gewünschten Inhaltsinformation aus dem Archivpaket herauszufiltern. Mit Hilfe der FLWOR-Ausdrücke ist es möglich, SQL Abfragen in XQuery abzubilden. Die Ergebnisse einer SQL Abfrage in der relationalen Datenbank werden so identisch mit den XQuery Abfragen in einem XML-Dokument sein.

2. Die Testdatenbank

Für diese Diplomarbeit wird eine relationale Testdatenbank entworfen, um die Tests durchzuführen. Es wird davon ausgegangen, dass diese Datenbank archivierungswürdig ist und bereits vom Datenbankproduzenten übergeben wurde.

Die Datenbank basiert auf dem Aufbau eines fiktiven Melderegisters, dabei wurden bewusst bestimmte Relationen gewählt, um spätere Probleme zu erkennen und zu beheben.

2.1. Datenbankaufbau

Die Datenbank „Melderegister“ hat folgenden Aufbau: [28]

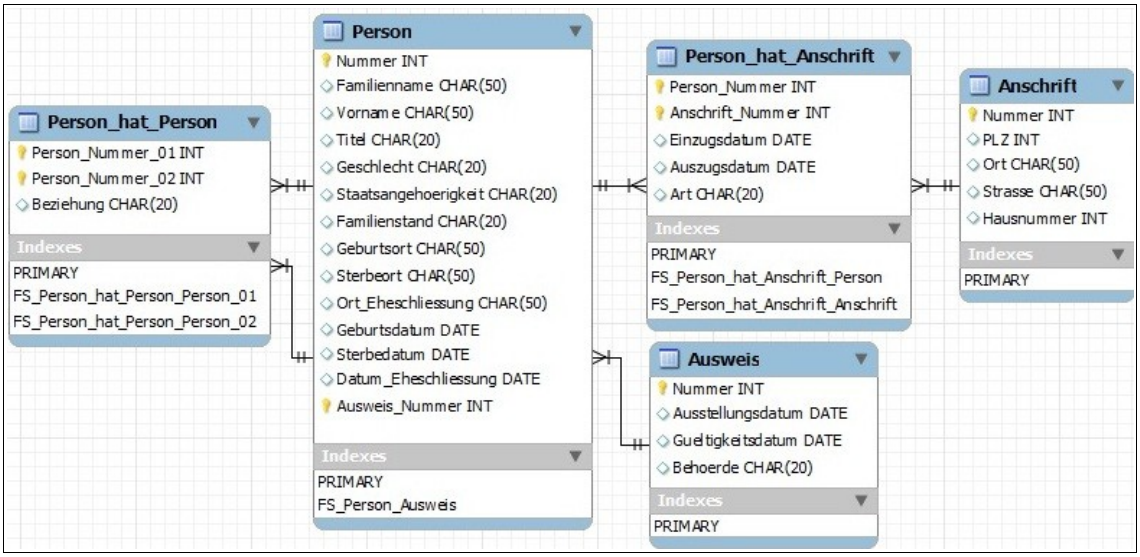


Abbildung 4: Datenbankaufbau Melderegister

Die Datenbank besteht aus fünf Tabellen mit dem Tabellentyp „InnoDB“, welcher Voraussetzung für Fremdschlüsseldefinitionen ist. Die Primärschlüssel werden in der oberen Abbildung durch ein Schlüssel-Symbol dargestellt.

28 Das Datenbankmodell Melderegister wurde mit dem Programm "MySQL Workbench 5.1. OSS" erstellt.

Die Datentypen der unterschiedlichen Spalten stehen direkt hinter dem jeweiligen Namen.

2.1.1. Datenbank Beziehungen

Abbildung 4 veranschaulicht die Beziehungen zwischen den Tabellen. Um möglichst viele unterschiedliche Beziehungstypen für die späteren Tests zur Verfügung zu haben, wurden folgende Beziehungen eingearbeitet:

- 1:1 Beziehung: Diese Beziehung besteht zwischen der Tabelle „Person“ und der Tabelle „Ausweis“. Jede Person hat genau eine Ausweisnummer und umgekehrt gehört zu jeder Ausweisnummer eine Personennummer.

- n:m Beziehung: Diese Beziehung besteht zwischen den Tabellen „Person“ und „Anschrift“ und wurde mit der Hilfstabelle „Person_hat_Anschrift“ ergänzt. Eine Person kann mehrere Wohnsitze haben und in einem Wohnsitz könnten mehrere Personen wohnen.

- rekursive Beziehung: In der Rekursion verweist die Tabelle „Person“ auf sich selbst. Dabei gehört zu einer Personennummer eine andere Personennummer. Damit werden die Beziehungen zwischen den Personen verdeutlicht, beispielsweise der Beziehungsstatus Ehepaar.

Mit Hilfe dieser unterschiedlichen Beziehungstypen ist es eventuell möglich, Fehler in der Testphase aufzudecken und Tipps für deren Beseitigung zu geben.

2.1.2. Feldbezeichnungen

In einigen Fällen ist es möglich, dass Datenbanken über keine Klartextnamen als Feldbezeichnungen verfügen. Dabei werden die Namen der einzelnen Felder und Tabellen in generische Feldbezeichnungen umgewandelt. Um diese generischen Felder wieder zu übersetzen, bedarf es einer Liste für die unterschiedlichen Feldbezeichnungen.

Generischer Feldname:	Klartext Feldname:
Tabelle_01	Person
Feld_01	Nummer
Feld_02	Name

Diese Tabelle soll das oben beschriebene Problem verdeutlichen. Mit generischen Feldnamen wird eine für den Menschen unleserliche Form erzeugt, die nur mit einer Feldbeschreibungsliste interpretierbar ist. Die Folge daraus ist, dass die Feldbeschreibungsliste während der Verarbeitung im OAIS-Modell mit übernommen werden muss. Diese Liste stellt den Bauplan zur späteren Weiterverwendung der Datenbank dar. Ohne deren Übernahme ist die archivierte Datenbank unbrauchbar.
Für die Testdatenbank Melderegister wurde auf generische Feldnamen verzichtet.

2.1.3. Datenbankinhalte

Die Testdatenbank wurde für die Tests mit fiktiven Inhalten gefüllt.

Nummer	Familienname	Vorname	Titel	Geschlecht	Staatsangehoerigkeit	Familienstand	Geburtsort
1	Lehmann	Holger	*NULL*	m	deutsch	ledig	Muenchen
2	Schmidt	Petra	*NULL*	w	deutsch	verheiratet	Berlin
3	Mueller	Sabine	Dr.	w	deutsch	ledig	Wien
4	Meier	Marcel	*NULL*	m	deutsch	ledig	Koeln
5	Schmidt	Klaus	Dr.	m	deutsch	verheiratet	Frankfurt

Sterbeort	Ort_Eheschliessung	Geburtsdatum	Sterbedatum	Datum_Eheschliessung	Ausweis_Nummer
NULL	*NULL*	1985-04-16	*NULL*	*NULL*	1
NULL	Berlin	1980-08-19	*NULL*	2006-06-06	2
NULL	*NULL*	1972-04-17	*NULL*	*NULL*	3
Koeln	*NULL*	1918-01-23	2007-04-25	*NULL*	4
NULL	Berlin	1977-12-14	*NULL*	2006-06-06	5

Abbildung 5: Melderegister - Tabelle Person

Nummer	PLZ	Ort	Strasse	Hausnummer
1	80469	München	Rumfordstraße	2
2	10247	Berlin	Frankfurter Allee	80
3	20354	Hamburg	Colonnaden	50
4	51149	Koeln	Ingeborgstrasse	100
5	10247	Berlin	Frankfurter Allee	80

Abbildung 6: Melderegister - Tabelle Anschrift

Nummer	Ausstellungsdatum	Gueltigkeitsdatum	Behoerde
1	2001-04-18	2011-04-18	München
2	2010-01-12	2020-01-12	Berlin
3	2009-09-22	2019-09-22	Hamburg
4	2002-05-02	2012-05-02	Koeln
5	2001-12-17	2011-12-17	Berlin

Abbildung 7: Melderegister - Tabelle Ausweis

Person_Nummer	Anschrift_Nummer	Einzugsdatum	Auszugsdatum	Art
1	1	1998-07-16	*NULL*	Hauptwohnsitz
2	2	1999-09-09	*NULL*	Hauptwohnsitz
3	3	1998-10-28	*NULL*	Hauptwohnsitz
4	4	2010-04-22	2007-04-25	Hauptwohnsitz
5	2	1999-09-09	*NULL*	Hauptwohnsitz

Abbildung 8: Melderegister - Tabelle Person_hat_Anschrift

Person_Nummer_01	Person_Nummer_02	Beziehung
2	5	Ehepaar

Abbildung 9: Melderegister - Tabelle Person_hat_Person

Die unterschiedliche Schreibweise, bei beispielsweise den Städtenamen mit Sonderzeichen, ist für die Testzwecke gewünscht. So ist es möglich zu testen, ob die Sonderzeichen beim Export mit übernommen werden können.
Alle nicht vorhandenen Inhalte in den Tabellen werden mit dem Wert „NULL“ gefüllt. „*Der Wert NULL ist nicht zu verwechseln mit der Zahl 0. Die Zahl 0 stellt eine Information dar, der Wert NULL jedoch nicht.*“[29] Da die Inhalte der Tabellenspalten mit festen Datentypen definiert sind, ist NULL die einzige Möglichkeit, einen fehlenden Inhalt einer Zelle darzustellen. Beim späteren Export muss NULL bzw. die Information, dass die Zelle leer ist, mit übernommen werden.
Die insgesamt fünf Testdatensätze sollten für die Darstellung der unterschiedlichen Probleme, die beim Export auftreten, ausreichen.

2.1.4. Abfragen und Sichten

Sichten[30] sind virtuelle Tabellen, die physisch nicht existieren. Sichten sind auch Abfragen, die immer wieder verwendet werden. Um die Arbeit zu vereinfachen und um wiederholtes Neueingeben der Abfragen zu verhindern, kann man die gewünschte Abfrage unter einem Namen abspeichern. Dadurch muss man nur den Namen der Sicht aufrufen, um die virtuelle Tabelle zu erzeugen. Mit Hilfe dieser Sichten ist es auch realisierbar, bestimmten Nutzern mit beschränkten Rechten nur einen Teil der inhaltlichen Informationen anzuzeigen. Mit solch einer Sicht könnten, im Beispiel des Melderegisters, nur Namen und Vornamen aber nicht Geburtsdaten und Wohnorte ausgegeben werden. Diese Zugriffsbeschränkung schützt die archivierte Datenbank vor unberechtigten Zugriffen.

29 Quelle: Däßler, Rolf: Das Einsteigerseminar MySQL 5. Heidelberg: bhv, 2005. ISBN-10 3-8266-7292-5 S. 78

30 Sichten oder auch engl. views

Die folgenden Sichten werden später mit in das OAIS-Modell übernommen und mit Hilfe von XQuery für den Nutzer im DIP zugänglich gemacht.

Abfrage_01:

```
SELECT Nummer, Familienname, Vorname FROM Person;
```

Diese Abfrage gibt nur die Vor- und Familiennamen sowie die Nummern der Personen aus der Tabelle Person aus. Die Abfrage könnte für beispielsweise Nutzer ohne höhere Zugriffsberechtigung verfügbar sein.

Abfrage_02:

```
SELECT person.Familienname, person.Vorname,
person_hat_anschrift.Einzugsdatum, person_hat_anschrift.Auszugsdatum,
anschrift.PLZ, anschrift.Ort, anschrift.Strasse, anschrift.Hausnummer
FROM person, person_hat_anschrift, anschrift
WHERE person.Nummer = person_hat_anschrift.Person_Nummer
AND anschrift.Nummer = person_hat_anschrift.Anschrift_Nummer;
```

Diese Abfrage gibt die Vor- und Familiennamen sowie die Informationen zum Wohnort aus. Dabei werden auch die Daten zum Ein- und Auszug angegeben. Die Abfrage verknüpft die Tabellen Person, Person_hat_Anschrift und Anschrift.

Abfrage_03:

```
SELECT person.Familienname, person.Vorname, ausweis.Gueltigkeitsdatum,
ausweis.Behoerde, ausweis.Ausstellungsdatum, anschrift.PLZ, anschrift.Ort
FROM person, ausweis, anschrift
WHERE person.Nummer = ausweis.Nummer AND ausweis.Nummer =
anschrift.Nummer
AND person.Ausweis_Nummer = ausweis.Nummer;
```

Mit dieser Abfrage werden Informationen zum Ausweis der Personen abgefragt. Dabei werden relevante Informationen zur Ausweisverwaltung und zur Gültigkeitsprüfung des Ausweises ausgegeben. Die Tabellen Person, Ausweis und Anschrift werden miteinander verknüpft.

Abfrage_04:

```
SELECT person.Familienname, person.Vorname,
person_hat_person.Beziehung, person.Geburtsdatum
FROM person, person_hat_person
WHERE person_hat_person.Person_Nummer_01 = person.Nummer
OR person_hat_person.Person_Nummer_02 = person.Nummer;
```

In Abfrage 04 werden die Beziehungen zwischen den Personen abgefragt. Die Tabellen Person und Person_hat_Person werden verknüpft. In der Beispieldatenbank Melderegister gibt es nur zwei Personen mit einem Status „Beziehung“. Bei mehreren Personen würde es Probleme mit der Zuordnung geben, daher wäre es dort ratsam, die jeweiligen Personennummern mit auszugeben.

Abfrage_05:

```
SELECT person.Familienname, person.Vorname,
person.Staatsangehoerigkeit, person.Geburtsort, person.Geburtsdatum,
person.Ausweis_Nummer, anschrift.PLZ, anschrift.Ort, anschrift.Strasse,
anschrift.Hausnummer, ausweis.Ausstellungsdatum,
ausweis.Gueltigkeitsdatum, ausweis.Behoerde
FROM person, anschrift, ausweis, person_hat_anschrift
WHERE person.Nummer = person_hat_anschrift.Person_Nummer
AND anschrift.Nummer = person_hat_anschrift.Anschrift_Nummer
AND person.Ausweis_Nummer = ausweis.Nummer;
```

Diese Abfrage gibt alle lesbaren Informationen aus, die sich auf einem Personalausweis befinden. Mit diesem „view“ ist es auch möglich, bestimmte Personen nach ihrer Ausweisnummer zu kontrollieren. Dazu müsste nur folgender SQL Befehl verwendet werden:

*SELECT * FROM abfrage_05 WHERE Ausweis_Nummer = 'x';*

Die Variable „x“ steht für die jeweilige Ausweisnummer.

3. Datenbankexport nach XML

In diesem Abschnitt wird der praktische Teil dieser Diplomarbeit erläutert. Dabei wird die relationale Datenbank aus Abschnitt 2. nach dem OAIS-Modell, aus Abschnitt 1.1.2., in das XML-Format exportiert. Für diesen Prozess stehen die in den folgenden Kapiteln beschriebenen Tools zur Verfügung.

3.1. Toolanalyse und Tests

Folgende Tools stehen für den Export der relationalen Datenbank nach XML zur Verfügung:

Name der Software	Version	Besonderheiten
Apache Friends XAMPP (Basis Package)	01.07.03	- simuliert einen Apache Server (Version 2.2.14) mit vorsinstalliertem MySQL 5.1.41 - beinhaltet die Testdatenbank Melderegister
phpMyAdmin	03.02.04	- Tool zur Administration der Datenbank - ist aus XAMPP vorinstalliert - ermöglicht Export als XML
MySQL 5	05.01.43	- installiert unter Windows7 - wird über die Kommandoebene aufgerufen - ermöglicht Export als XML
Saxon XSLT and XQuery Processor	9-2-1-1	- führt XQuery Dateien (.xq) aus - wird über die Kommandoebene aufgerufen
Altova DatabaseSpy	v2008 rel. 2	- greift auf Datenbanken zu - ermöglicht Import und Export von XML
Altova XML Spy	v2008 rel. 2	- Editor für u.a. XML, DTD und XSD - ermöglicht die Generierung von Schema Dateien - Validiert XML und XML-Schema Dateien

3.1.1. Kommandoebene von MySQL

Über die Windows Kommandoebene[31] wird auf den lokalen Apache Server, welcher durch XAMPP installiert wurde, zugegriffen.
Mit der Befehlskette:
mysql -u root -h localhost -p
wird als Nutzer root (-u) auf den Host localhost (-h) zugegriffen. Die darauf folgende Passwortabfrage (-p) wird mit der Enter-Taste bestätigt, da kein Passwort festgelegt wurde. Mit dem SQL-Befehl: „*show databases;*“ ist es möglich, sich die im Server abgelegten Datenbanken anzusehen.

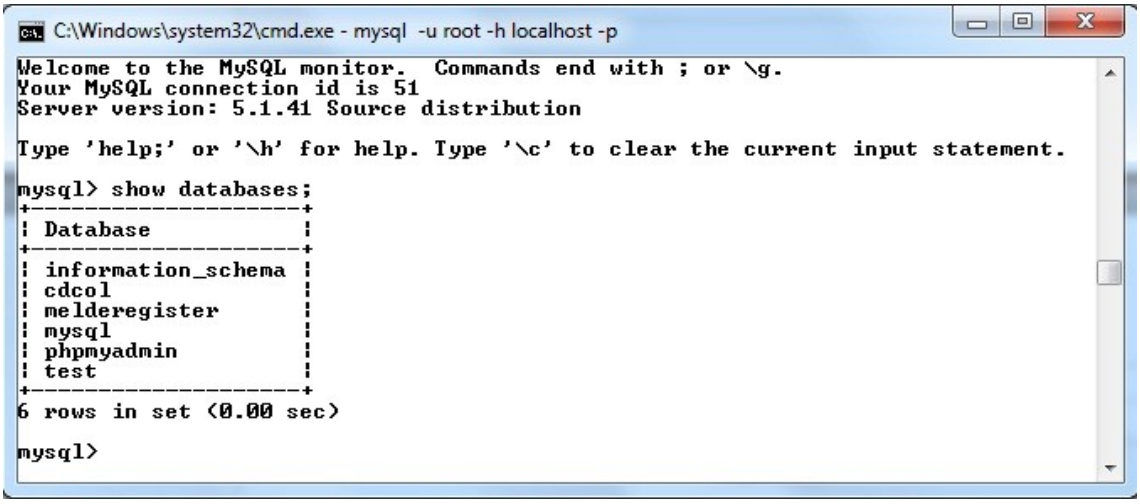


Abbildung 10: Windows Kommandoebene SQL Befehl "show databases;"

Der Zugriff auf die Datenbank war somit erfolgreich. Die Datenbank Melderegister wird angezeigt. Diese Datenbank muss nun nach XML exportiert werden. Für den nächsten Schritt muss die Kommandoebene über MySQL verlassen werden.
Nachdem die Kommandoebene neu geöffnet wurde, wird der Befehl zur Ausführung von mysqldump eingegeben.
Der Befehl lautet:
mysqldump -X -u root -p melderegister person ausweis anschrift person_hat_person person_hat_anschrift > melderegister.xml

31 Die Kommandoebene ist bei Windows unter .../Zubehör/Ausführen und der Eingabe des Kommandos cmd zu erreichen. Zusätzlich ist auch unter XAMPP eine xampp_shell.bat enthalten, welche die Kommandoebene aufruft.

Das Dienstprogramm mysqldump wird mit dem Befehl „mysqldump“ aufgerufen. Dann folgen die Optionen für die Ausgabe der dump-Datei im XML-Format (-X) und die Zugangsbefehle als Nutzer (-u) root mit einer Passwortabfrage (-p). Die Datenbank Melderegister wird ausgewählt und mit den danach folgenden Tabellen Person, Ausweis, Anschrift, Person_hat_Person und Person_hat_Anschrift ergänzt. Die so entstandene XML-Datei ist im Anhang A2 zu finden.
Nun folgt ein Ausschnitt aus dem eben erstellten XML-Dokument:

```
<table_data name="person">
        <row>
                <field name="Nummer">1</field>
                <field name="Familienname">Lehmann</field>
                <field name="Vorname">Holger</field>
                <field name="Titel" xsi:nil="true" />
                <field name="Geschlecht">m</field>
                <field name="Staatsangehoerigkeit">deutsch</field>
                <field name="Familienstand">ledig</field>
                <field name="Geburtsort">Muenchen</field>
                <field name="Sterbeort" xsi:nil="true" />
                <field name="Ort_Eheschliessung" xsi:nil="true" />
                <field name="Geburtsdatum">1985-04-16</field>
                <field name="Sterbedatum" xsi:nil="true" />
                <field name="Datum_Eheschliessung" xsi:nil="true" />
                <field name="Ausweis_Nummer">1</field>
        </row>
```

Dieser Abschnitt repräsentiert einen Datensatz aus der Tabelle „Person“ der Datenbank Melderegister. Das XML-Dokument wird in <row> und <field>[32] Elementen dargestellt. Für die Menschenlesbarkeit wäre es besser, die Elemente <row> durch den Namen der Tabelle zu ersetzen. Dann würde aus dem Element <row> das Element <person> werden. Besonders auffallend ist das *„xsi:nil=“true““* in einigen Elementen. Diese Elemente wurden aus den Zellen generiert, die in der relationalen Datenbank den Inhalt NULL enthielten. *„Wert true für das xsi:nil-Attribut in einem XML-Element gibt ausdrücklich an, dass das Element keinen Inhalt hat, d.h. weder untergeordnete Elemente noch Textkörper.“* [33]

32 row kann man mit Reihe und field mit Feld zu übersetzten

33 Quelle: Microsoft Library: http://msdn.microsoft.com/de-de/library/ybce7f69%28VS.80%29.aspx

In dem Element <table_structure> werden alle Struktur-Informationen zu der jeweiligen Tabelle abgelegt.
Diese Informationen sind zur Wiederherstellung der Tabellen in eine relationale Datenbank notwendig.

3.1.2. PHPMyAdmin

Der PHPMyAdmin ist ein Tool zur Administration von Datenbanken und wird in der Version 3.2.4. mit XAMPP mitgeliefert. Mit Hilfe dieses Tools ist es realisierbar, über eine grafische Oberfläche Datenbanken zu administrieren. Die Option „*Exportieren*“ ermöglicht das Exportieren der ausgewählten Datenbank in ein gewünschtes Exportformat.

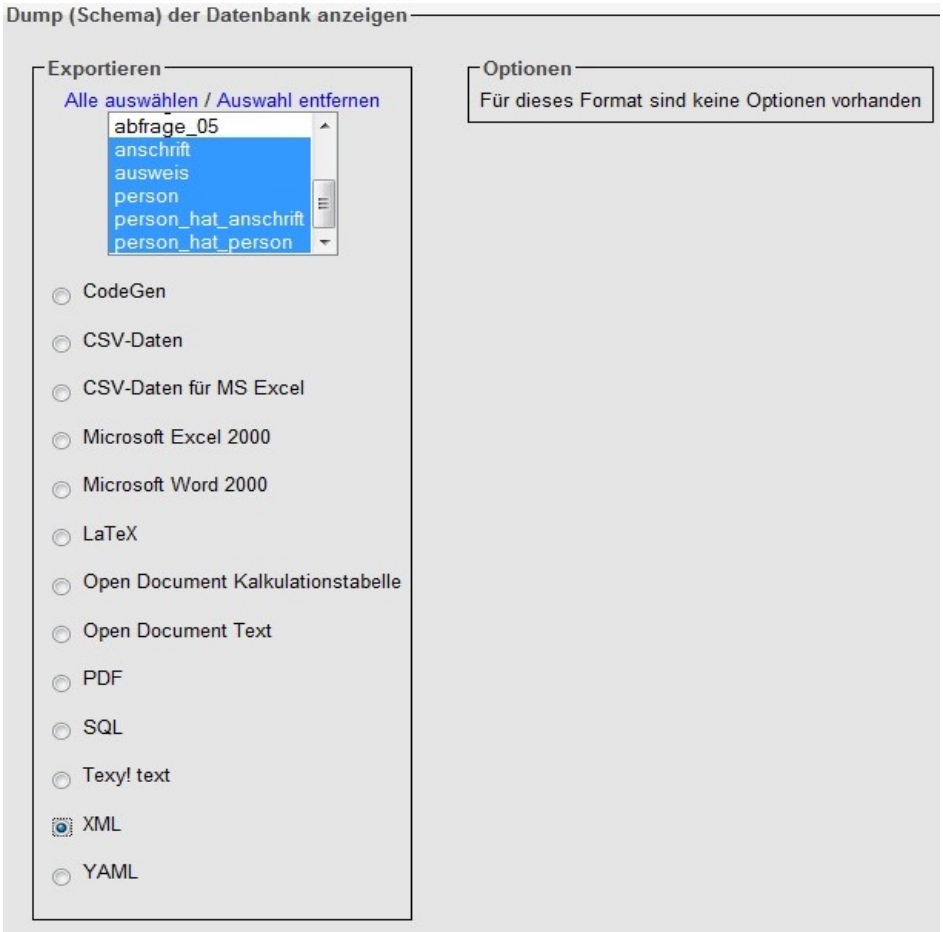


Abbildung 11: PHPMyAdmin Export Ansicht

Wie in dem Exportverfahren von der Kommandoebene aus Abschnitt 3.1.1. wird auf die Tabellen „abfrage_01“ bis „abfrage_05“ verzichtet.
Denn wie bereits in Abschnitt 2.1.4. erwähnt, sind Sichten/Abfragen virtuelle Tabellen, die physisch nicht existieren. Wenn diese Abfragen nach XML exportiert werden würden, würden diese eine physische Existenz erhalten. Wie in der Grafik zu erkennen, hat das Tool keine weiteren Optionen für den Export nach XML.
Das gesamte XML-Dokument, das mit Hilfe des PHPMyAdmins generiert wurde, ist in Anhang A3 zu finden.
Der folgende Ausschnitt ist ein Ausschnitt aus dem Anhang A3 und repräsentiert die Tabelle Person aus der Testdatenbank Melderegister.

```
<!-- Tabelle person -->
  <person>
    <Nummer>1</Nummer>
    <Familienname>Lehmann</Familienname>
    <Vorname>Holger</Vorname>
    <Geschlecht>m</Geschlecht>
    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
    <Familienstand>ledig</Familienstand>
    <Geburtsort>Muenchen</Geburtsort>
    <Geburtsdatum>1985-04-16</Geburtsdatum>
    <Ausweis_Nummer>1</Ausweis_Nummer>
  </person>
```

Bei diesem Ausschnitt werden alle Elemente mit den korrekten Bezeichnungen aus der Datenbank Melderegister dargestellt. Das Wurzelelement <person> umschließt die Unterelemente und bildet so die Tabelle Person ab. Auffallend ist das Fehlen der Zellen mit dem Inhalt NULL. Zellen, welche in der relationalen Datenbank den Inhalt NULL haben, wurden nicht generiert und fehlen als Elemente in der XML-Datei. Strukturinformationen der Tabellen und Definitionen der Datentypen fehlen in diesem XML-Dokument. Die wenigen enthaltenen Metadaten beinhalten lediglich Informationen zur PHPMyAdmin Version, dem Host, der Erstellungszeit, dem Datenbanknamen, die Server und PHP Versionen.

Allerdings sind diese Metadaten als Kommentare in die XML-Datei eingebunden.

3.1.3. Altova DatabaseSpy

Das Datenbankadministrations-Tool DatabaseSpy von Altova[34] hat sehr umfangreiche Funktionen für den Export von relationalen Datenbanken nach XML. Durch die Menüleiste gelangt man über die Option Tools zu dem Menü „*Export database data...*“.

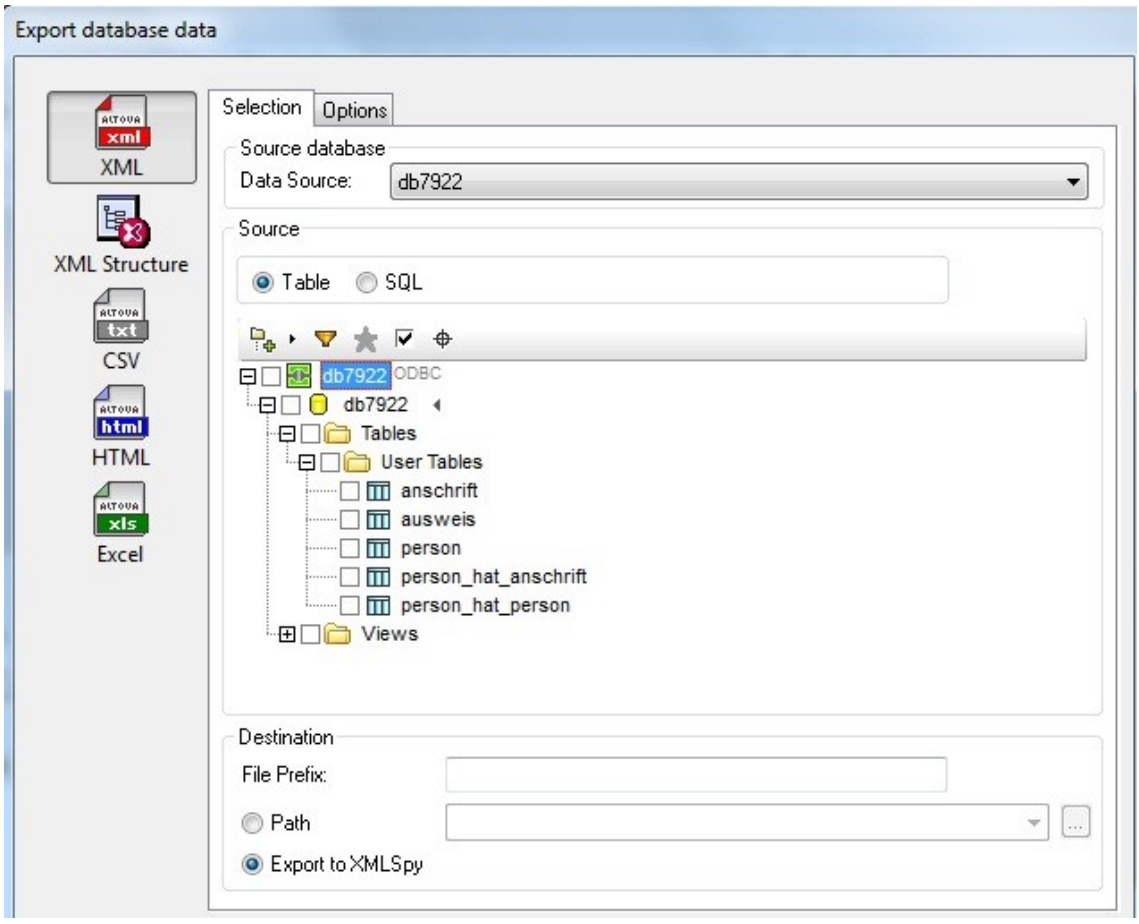


Abbildung 12: DatabaseSpy Export Menü für XML

34 Link zur Webseite von Altova und dem Tool DatabaseSpy
http://swarchive.altova.com/de/databasespy.html

Für diesen Test liegen die Tabellen der Datenbank Melderegister auf dem Server der Fachhochschule Potsdam, da XAMPP über keine funktionstüchtige ODBC Schnittstelle verfügt. Über die ODBC-Schnittstelle kann sich DatabaseSpy mit der Datenbank verbinden. Es gibt zwei Möglichkeiten, um mit DatabaseSpy ein XML-Dokument zu erzeugen.
Die erste Option ist die „XML“ Funktion. Dabei befinden sich alle Tabellen im Ordner „*User Tables*“, siehe dazu Abbildung 12. Die Sichten befinden sich im Ordner „*Views*“. Wie auch in den anderen Tests, werden die Sichten nicht mit in das XML-Dokument exportiert. Für die XML Exportfunktionen gibt es folgende Optionen:

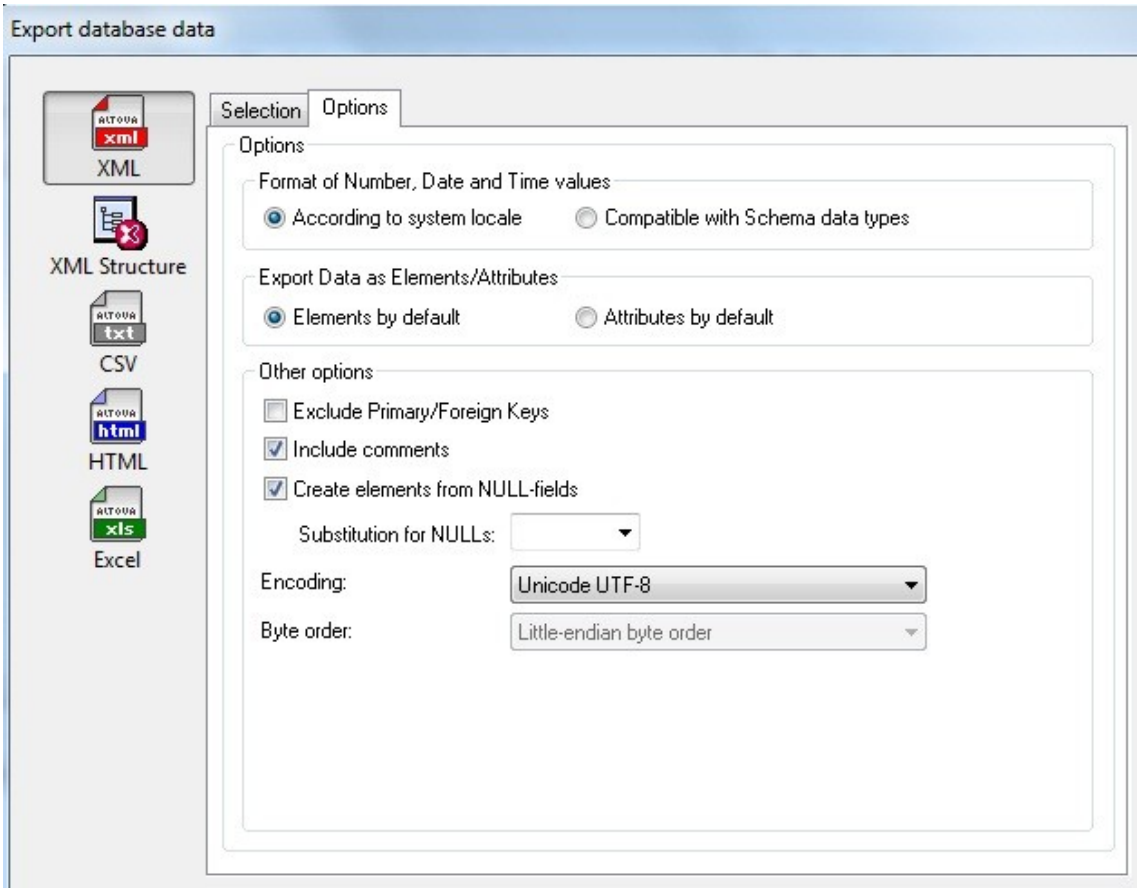


Abbildung 13: DatabaseSpy Export Menü Optionen

In den Optionen ist es möglich, die Formate von Zahlen, Datum und Zeit festzulegen. Dabei kann man zwischen einer lokalen Lösung und einer Lösung über Schema-Dateien wählen.

Zudem besteht die Möglichkeit zwischen Attributen und Elementen für den Export zu wählen. Ebenfalls können Primär- und Fremdschlüssel sowie Kommentare ausgegrenzt werden. Eine weitere Option ermöglicht die Festlegung wie die Zellen mit dem Inhalt NULL exportiert werden sollen. Die Option „*Encoding*“ ermöglicht das Festlegen eines bestimmten Zeichensatzes.
Der Export wird mit den Einstellungen aus Abbildung 13 durchgeführt. Die Tabellen der Datenbank Melderegister wurden mit Hilfe von DatabaseSpy in fünf XML-Dokumente exportiert. Jedes dieser XML-Dokumente beinhaltete eine einzelne Tabelle. Die fünf XML-Dokumente befinden sich im Anhang A4.

```
<Row>
        <Nummer>1</Nummer>
        <Familienname>Lehmann</Familienname>
        <Vorname>Holger</Vorname>
        <Titel/>
        <Geschlecht>m</Geschlecht>
        <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
        <Familienstand>ledig</Familienstand>
        <Geburtsort>Muenchen</Geburtsort>
        <Sterbeort/>
        <Ort_Eheschliessung/>
        <Geburtsdatum>1985-04-16</Geburtsdatum>
        <Sterbedatum/>
        <Datum_Eheschliessung/>
        <Ausweis_Nummer>1</Ausweis_Nummer>
</Row>
```

Der oben abgebildete XML-Code ist ein Ausschnitt aus dem XML-Dokument person.xml, welches mit DatabaseSpy aus der relationalen Datenbank exportiert wurde. Das Wurzelelement bei diesem XML-Dokument heißt <Row> und die Unterelemente besitzen die korrekten Spaltennamen aus der Datenbank. Das Element <Row> müsste <person> heißen, um die Tabelle Person eins zu eins abzubilden. Beim Export wurden aus den mit dem Wert NULL befüllte Zellen in leere Elemente umgewandelt.

Jede der fünf XML-Dateien beinhaltet Informationen zur Struktur und zu den Datentypen. Des Weiteren befinden sich in jeder zweiten Zeile jedes Dokumentes ein Import-Name mit einer SQL-Abfrage, welche alle Spaltennamen beinhaltet.

Die zweite Möglichkeit mit DatabaseSpy einen XML-Export durchzuführen lautet: „*XML Structure*“.

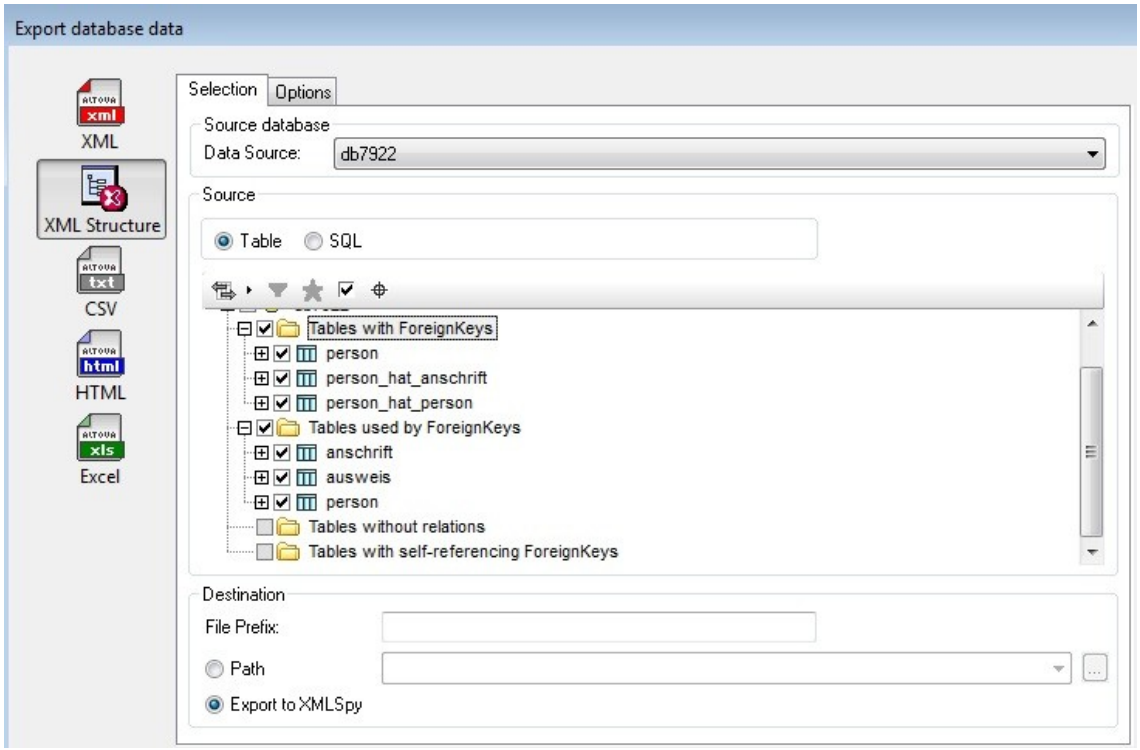


Abbildung 14: DatabaseSpy XML Export Menü für XML Structure

Wie in Abbildung 14 zu erkennen, werden die Tabellen in einer Verzeichnisstruktur sortiert. Dabei gibt es die beiden Ordner „*Tables with ForeignKeys*“ und „*Tables used by ForeignKeys*“. Diese Struktur wird anhand der Fremdschlüssel in den einzelnen Tabellen gebildet. Die Tabelle Person kommt so zweimal vor, da die Tabelle über die Spalten-Nummer und Ausweis_Nummer verfügt. Sobald eine der Tabellen selektiert wird, werden auch die anderen Tabellen ausgewählt, die mit diesem Fremdschlüssel verbunden sind.

So werden mindestens immer zwei Tabellen in eine XML-Datei exportiert. Im oberen Beispiel werden alle Tabellen ausgewählt, um diese in XML zu exportieren. Das Herausnehmen der zweiten Tabelle „Person“ ist nicht möglich. Die Tabellen werden ebenfalls mit den Einstellungen aus Abbildung 13 exportiert. Durch den Export werden zwei XML-Dokumente erzeugt, diese befinden sich im Anhang A5.

```
<ausweis>
        <Nummer>1</Nummer>
        <Ausstellungsdatum>2001-04-18</Ausstellungsdatum>
        <Gueltigkeitsdatum>2011-04-18</Gueltigkeitsdatum>
        <Behoerde>München</Behoerde>
        <person>
            <Nummer>1</Nummer>
            <Familienname>Lehmann</Familienname>
            <Vorname>Holger</Vorname>
            <Titel/>
            <Geschlecht>m</Geschlecht>
            <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
            <Familienstand>ledig</Familienstand>
            <Geburtsort>Muenchen</Geburtsort>
            <Sterbeort/>
            <Ort_Eheschliessung/>
            <Geburtsdatum>1985-04-16</Geburtsdatum>
            <Sterbedatum/>
            <Datum_Eheschliessung/>
            <Ausweis_Nummer>1</Ausweis_Nummer>
            <person_hat_anschrift>
                <Person_Nummer>1</Person_Nummer>
                <Anschrift_Nummer>1</Anschrift_Nummer>
                <Einzugsdatum>1998-07-16</Einzugsdatum>
                <Auszugsdatum/>
                <Art>Hauptwohnsitz</Art>
            </person_hat_anschrift>
        </person>
    </ausweis>
```

Der oben dargestellte XML-Code ist ein Ausschnitt aus dem mit DatabaseSpy erzeugten XML-Dokument „ausweis.xml“. Es ist gut zu erkennen, dass DatabaseSpy beim Export die Strukturen der Tabellen verändert. So ist das Element <person> ein Unterelement von <ausweis>. Die Elementbezeichnungen werden korrekt dargestellt, allerdings fehlen in diesem Dokument die Informationen aus der Tabelle „Anschrift“. Scheinbar kann DatabaseSpy keine n:m Beziehungen, wie sie zwischen den Tabellen Person und Anschrift mit der Hilfstabelle Person_hat_Anschrift besteht, darstellen.

Die NULL Elemente der Datenbank werden im XML-Dokument, den Einstellungen entsprechend mit leeren Elementen, dargestellt. Metadaten, welche als Kommentarfunktion in den Code eingebunden wurden, lassen sich nur für die Tabelle „Ausweis“ finden. Weitere Datentypen und Strukturinformationen der Tabellen fehlen in diesem XML-Dokument.

3.1.4. Auswertung der Testreihe

In den vorherigen Abschnitten wurde die Testdatenbank Melderegister mit unterschiedlichen Tools und auf unterschiedlicher Weise nach XML exportiert. Dabei lässt sich sehr schnell feststellen, dass sich alle Ergebnis-XML-Dateien voneinander unterscheiden. [35] Anhand der Tabelle, welche sich im Anhang A6 befindet, lassen sich diese Unterschiede verdeutlichen.

Für die Weiterverarbeitung des XML-Dokumentes wird ein gut strukturiertes XML-Format benötigt. Die Testdatenbank ist eine sehr kleine Datenbank, die aus nur fünf Tabellen und fünf Beispieldatensätzen besteht. Im Falle einer größeren Datenbank ist es möglich, dass das XML-Dokument sehr groß und unübersichtlich wird. Auch die Größe der XML-Datei in Bytes sollte berücksichtigt werden.

Anhand der Tests steht fest, dass keines der exportierten XML-Dokumente vollkommen fehlerfrei bzw. für die Weiterverarbeitung in dieser Diplomarbeit ohne Einschränkungen nutzbar ist.

Die MySQL-Kommandoebene eignet sich gut zur Erstellung eines Datenbankbackups, welches später wiederhergestellt werden könnte. Danach wurde auch der XML-Code aufgebaut. Die Größe der Datei sticht bei dieser Variante besonders hervor, 13,3 Kilobyte für ein XML-Dokument sind bei diesen Tests die größte Angabe. Störend wirkt zudem die Elementbezeichnung des Wurzelelementes <row>. Für eine spätere Weiterverarbeitung werden Elementbezeichnungen mit eindeutigen Namen benötigt. Der Export mit der MySQL-Kommandoebene nach XML ist für die spätere Weiterverarbeitung nicht geeignet.

35 Verglichen wurden dabei die Anhänge A2, A3, A4 und A5.

Beim Export mit dem PHPMyAdmin gibt es keine weiteren Einstellungsmöglichkeiten. Die erzeugte Datei ist dafür nur 6,24 Kilobyte groß und enthält eine korrekte Struktur mit exakten Elementbezeichnungen. Die geringe Größe der Datei entsteht dadurch, dass die Elemente mit leerem Inhalt nicht erzeugt wurden. Jedoch wäre ein leeres Element wesentlich sinnvoller für eine spätere Weiterverarbeitung. Dennoch ist der Export mit PHPMyAdmin eine Möglichkeit zur geeigneten Weiterverarbeitung.
Beide Exportvarianten von DatabaseSpy erzeugen mehrere XML-Dokumente. Das führt dazu, dass die Dokumente größer als die anderen Varianten sind. Die korrekte Elementbezeichnung ist nur beim Export über die „*XML Structure*“ Einstellung stimmig. Vorteilhaft sind die vielen Einstellungsmöglichkeiten für den Export nach XML. Bei beiden Varianten werden leere XML-Elemente erzeugt, wenn sich in der relationalen Datenbank ein NULL Wert an dieser Stelle befand. Das große Problem n:m Beziehungen korrekt darzustellen, macht DatabaseSpy mit der „*XML Structure*“ Variante unbrauchbar. Denn durch die zwei exportierten XML-Dokumente entstehen Dubletten bei dem Element <person_hat_anschrift>, welches sich in beiden XML-Dokumenten befindet. Die normale XML-Exportvariante erzeugt hingegen brauchbare Ergebnisse. Bis auf das Wurzelelement <row> und dem Export von jeder Tabelle in ein XML-Dokument, ist DatabaseSpy mit der XML-Exportvariante für eine Weiterverarbeitung geeignet.
Möglicherweise müssen die vorhandenen Programme verbessert und angepasst werden, um sinnvolle Ergebnisse zu erzielen. An dieser Stelle bedarf es besserer Software für den Export von relationalen Datenbanken nach XML. Folgende Kriterien sollte die Software beim Datenbankexport erfüllen:

- korrekte Darstellung der Wurzel- und Unterelemente
- Umgangsmöglichkeit mit dem Wert NULL
- geringe Dateigröße
- relevante Datentypen und Strukturinformationen sollten enthalten sein

Für die weiteren Tests wurde das XML-Dokument, welches von PHPMyAdmin exportiert wurde, ausgewählt. Dieses Dokument erfüllt viele der nötigen Kriterien, ist zudem gut für den Menschen lesbar und enthält wichtige Metadaten zum Datenbankexport. Auf Basis dieser XML-Datei werden die weiteren Tests aufgebaut.

3.2. Weiterverarbeitung des XML-Dokumentes

Unter dem Punkt 3.1. wurden die Tools zum Export der Datenbank in ein XML-Format erläutert. In diesem Abschnitt wird die exportierte XML-Datei weiterverarbeitet. Die Abfragen müssen mit in das Archivpaket übernommen und die referenzielle Integrität überprüft werden. Des Weiteren sind Metadaten zu erheben und gesondert abzulegen. Die Abfragen sind ebenfalls wieder im Auslieferungspaket dem Nutzer zugänglich zu machen. Wie die eben erzeugte XML-Datei weiterverarbeitet werden sollte, wird in diesem Abschnitt erläutert.

3.2.1. Wiederherstellung der referenziellen Integrität

In dem von PHPMyAdmin erzeugten XML-Dokument existieren keine Verknüpfungen und Schlüssel, welche eine Beziehung zwischen den einzelnen Elementen darstellen. Diese Verbindungen sind aber wichtig, um die Beziehungen zwischen den ehemaligen Tabellen nachzuvollziehen. Allein durch das XML-Dokument ist es nicht möglich zu erkennen, dass beispielsweise das Element <Ausweis_Nummer> von dem Wurzelelement <person> mit dem Element <Nummer> vom Wurzelelement <ausweis> verbunden ist. Eine solche Verbindung lässt sich nur mit Hilfe von XQuery Abfragen und auf Basis einer XML-Schema Datei repräsentieren. Diese beiden Möglichkeiten werden in den nächsten Abschnitten erläutert.

3.2.1.1. XML-Schema

Zunächst wird aus der Datei „*melderegister.xml*“, welche mit PHPMyAdmin exportiert wurde, eine Schema-Datei generiert. Mit Hilfe von Altova XMLSpy ist es über den Menüpunkt „*DTD/Schema*“ möglich, eine DTD oder eine XML-Schema Datei zu erzeugen. Um später keyref einzubinden, muss eine XML-Schema Datei generiert werden. Die generierte Schema-Datei befindet sich im Anhang A7. Bei dieser Schema-Datei ist eine deutliche Verschachtelung bei „person“ zu erkennen. Dadurch, dass in der melderegister.xml einige Elemente fehlen, welche in der relationalen Datenbank NULL enthielten, wurde eine „*choice*“ (Auswahl) eingefügt. Diese Auswahlen stellen Möglichkeiten für die weitere Vergabe von Elementen dar. Diese Variante ist aber unbrauchbar, da wichtige Elemente wie <Ausweis_Nummer> doppelt vorkommen. Eine brauchbare Schema-Datei befindet sich im Anhang A8. Sie wurde aus einer veränderten melderegister.xml generiert. Die Veränderung basiert auf dem händischen Nachtragen der fehlenden Elemente im Wurzelelement <person>.

Unveränderte melderegister.xml	Veränderte melderegister.xml
<person> <Nummer>1</Nummer> <Familienname>Lehmann</Familienname> <Vorname>Holger</Vorname> <Geschlecht>m</Geschlecht> <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit> <Familienstand>ledig</Familienstand> <Geburtsort>Muenchen</Geburtsort> <Geburtsdatum>1985-04-16</Geburtsdatum> <Ausweis_Nummer>1</Ausweis_Nummer> </person>	<person> <Nummer>1</Nummer> <Familienname>Lehmann</Familienname> <Vorname>Holger</Vorname> <Titel/> <Geschlecht>m</Geschlecht> <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit> <Familienstand>ledig</Familienstand> <Geburtsort>Muenchen</Geburtsort> <Sterbeort/> <Ort_Eheschliessung/> <Geburtsdatum>1985-04-16</Geburtsdatum> <Sterbedatum/> <Datum_Eheschliessung/> <Ausweis_Nummer>1</Ausweis_Nummer> </person>

Diese Tabelle zeigt die Veränderung des Elementes <person>. Die dadurch entstehende Schema-Datei aus Anhang A9 ist für eine Weiterverarbeitung geeignet. Die vorher fehlenden Elemente sind nun optional einsetzbar (Elemente mit gestricheltem Rechteck).

Dadurch, dass die Schema-Datei aus den Elementinhalten der melderegister.xml generiert wird, entstehen im Quelltext der Schema Datei folgende Strukturen:

```
<xs:element name="Familienname">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Lehmann"/>
                <xs:enumeration value="Meier"/>
                <xs:enumeration value="Mueller"/>
                <xs:enumeration value="Schmidt"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
```

Die <xs:enumeration value=""/> legt fest, dass nur der Wert zwischen den Anführungszeichen verwendet werden darf. Wenn im Beispiel der melderegister.xml eine weitere Person mit dem Nachnamen „Kunze“ hinzukommen würde, wäre die XML-Datei nicht mehr valide. Die XML-Datei darf also nach dem generieren der Schema-Datei nicht mehr verändert werden. Da XML-Schema die Elementinhalte mit den jeweiligen Inhalten der <xs:enumeration value=""/> aus der Schema-Datei validiert.

Es ist möglich mit Hilfe von keyref Schlüsselbedingungen zu definieren. Im Gegensatz zu IDREF müssen für keyref keine Attribute in der Datei melderegister.xml eingefügt werden. Denn keyref referenziert auf Schemaebene. „*Außerdem können Schlüsselbestandteile nicht nur Attributinhalte, sondern auch Elementinhalte darstellen.*“ [36] Das nachträgliche Einfügen von Attributen, um IDREF nutzen zu können, ist sehr umständlich und wird bei zunehmend größeren XML-Dateien aufwendiger. Daher ist keyref geeigneter, um die referenzielle Integrität wiederherzustellen. Aber auch bei keyref müssen nachträglich Schlüssel in die Schema-Datei eingetragen werden. Der Aufwand wäre für größere Schema-Dateien enorm hoch und könnte zu Fehlern führen, da die Eingabe manuell erfolgen muss. Dieser Aspekt macht keyref inakzeptabel für die Wiederherstellung der referenziellen Integrität.

36 Quelle: Kazakos, Wassilios; Schmidt, Andreas; Tomczyk, Peter: Datenbanken und XML: Springer Verlag. 1. Auflage Berlin Heidelberg, 2002, ISBN 3-540-41956-X, S.67

3.2.1.2. XQuery-Abfragen

Im vorherigen Abschnitt wurde festgestellt, dass keyref aufgrund der manuellen Änderung für die Wiederherstellung der referenziellen Integrität unbrauchbar ist. Die Abfragesprache XQuery hingegen kann die melderegister.xml analysieren, ohne dabei den Inhalt der XML-Datei oder der Schema-Datei zu verändern.
Für die Wiederherstellung werden die Sichten aus der relationalen Datenbank in XQuery-Abfragen umgewandelt. Diese Umwandlung erfolgt manuell, das heißt, dass die SQL-Abfragen analysiert und in XQuery abgebildet werden müssen. Gleichzeitig sind die Sichten als Abfragen im späteren Auslieferungspaket verwendbar. Die SQL Abfragen aus dem Abschnitt 2.1.4. werden nun in XQuery Dateien (.xq) umgewandelt:

abfrage_01 in SQL:

```
SELECT Nummer, Familienname, Vorname FROM Person;
```

abfrage_01 in XQuery:

```
<abfrage_01.xml>{
let $m := fn:doc("melderegister.xml")
for $p in $m//person
order by $p/Nummer
return
<person>{$p/Nummer, $p/Familienname, $p/Vorname}</person>
}</abfrage_01.xml>
```

Die SQL-Abfrage_01 wird mit der oben dargestellten XQuery-Abfrage repräsentiert. Dabei werden durch beide Abfragen die gleichen Ergebnismengen erzeugt. Alle erstellten XQuery-Abfragen befinden sich im Anhang A10. Jede Abfrage wird mit Hilfe des Programms „*Saxon*“ und dem Kommando „*java -cp c:/saxon/saxon9he.jar net.sf.saxon.Query abfrage_01.xq*“ ausgeführt. Saxon wird über die Kommandoebene ausgeführt und durch die Zugabe des Befehls „> *abfrage_01.xml*“ wird das Ergebnis als XML-Dokument in der abfrage_01.xml Datei ausgegeben.

Die SQL abfrage_01 erzeugt folgende Tabelle:

Nummer	Familienname	Vorname
1	Lehmann	Holger
2	Schmidt	Petra
3	Mueller	Sabine
4	Meier	Marcel
5	Schmidt	Klaus

Abbildung 15: SQL abfrage_01

XQuery erzeugt mit Hilfe von Saxon eine XML-Datei mit dem Namen abfrage_01.xml:

<?xml version="1.0" encoding="UTF-8"?>
<abfrage_01.xml><person><Nummer>1</Nummer><Familienname>Lehmann</Familienname>
<Vorname>Holger</Vorname></person><person><Nummer>2</Nummer><Familienname>Sch
midt</Familienname><Vorname>Petra</Vorname></person><person><Nummer>3</Nummer>
<Familienname>Mueller</Familienname><Vorname>Sabine</Vorname></person><person><N
ummer>4</Nummer><Familienname>Meier</Familienname><Vorname>Marcel</Vorname></p
erson><person><Nummer>5</Nummer><Familienname>Schmidt</Familienname><Vorname>
Klaus</Vorname></person></abfrage_01.xml>

Für eine verbesserte Menschenlesbarkeit sollte die oben dargestellte XML-Datei angepasst werden. Dabei sollten Wurzel- und Unterelemente eingerückt werden, um eine verständlichere Struktur zu erzeugen.
Sowohl die SQL-Tabelle, als auch die XML-Datei enthalten die gleichen inhaltlichen Informationen. Im Anhang A11 befinden sich die mit Saxon erstellten XML-Dokumente und werden mit den Ergebnissen der SQL-Abfragen gegenübergestellt.

Die referenzielle Integrität kann nun mit den XQuery-Abfragen überprüft werden. Mit den nötigen XQuery-Kenntnissen ist es möglich, aus den XQuery-Abfragen die Beziehungen zwischen den ehemaligen Fremdschlüsseln abzulesen.

Die Fremdschlüsselbedingungen werden bei den XQuery-Abfragen in der let Anweisung definiert.

Nach dem Beispiel der let Anweisung aus abfrage_03.xq sieht das wie folgt aus:

```
let $ausweis := doc("melderegister.xml")/melderegister/ausweis[Nummer=$person/Ausweis_Nummer]
let $anschrift := doc("melderegister.xml")/melderegister/anschrift[Nummer=$person/Nummer]
```

Die erste let Anweisung sagt aus, dass das Unterelement „Nummer“ aus dem Wurzelelement „ausweis“ in der Datei „melderegister.xml“ mit dem Unterelement „Ausweis_Nummer“ vom Wurzelelement „person“ verknüpft ist. Sobald die Werte aus beiden Unterelementen „Nummer“ und „Ausweis_Nummer“ gleich sind, können die Inhalte miteinander verknüpft werden. Dabei muss in der späteren Abfrage nur noch die Variable „$ausweis“ aufgerufen werden. Diese let Abfrage drückt sozusagen die folgende SQL Bedingung aus: „WHERE person.Nummer = ausweis.Nummer“.

Anhand dieser let Bedingungen ist es möglich, die referenzielle Integrität abzulesen. Allerdings muss dafür zuerst eine manuelle Übernahme der SQL-Abfragen nach XQuery durchgeführt werden. In diesen XQuery-Abfragen müssen auch alle Beziehungsvarianten enthalten sein, da nicht vorhandene Varianten verloren gehen könnten.

Abfragen sind demnach essentiell für die Datenbankarchivierung, denn nur mit Hilfe der Abfragen ist es möglich, die referenzielle Integrität zu rekonstruieren bzw. zu überprüfen.

Mit XQuery ist es zudem möglich, bestimmte Abfragen für die Nutzer vorzubereiten. Diese Abfragen könnten nach dem OAIS-Modell im Auslieferungspaket vorgelegt werden. Doch für diese Bedürfnisse müssten spezielle XQuery-Abfragen erstellt werden, die genau auf die Wünsche der Nutzer angepasst werden.

Diese Abfragen werden von Datenbank zu Datenbank unterschiedlich ausfallen, sodass es keine universellen Abfragen geben wird. Doch ist es möglich, einige Freiheiten für den Nutzer einzubauen.

Durch die XQuery Befehlszeile „*declare variable $Variable external;*" ist es möglich, für den Wert "*$Variable*" eine feste Variable einzubinden.

```
declare variable $Familienname external;
<abfrage_fam.xml>{
let $fam := doc("melderegister.xml")/melderegister/person[$Familienname=Familienname]
order by $fam/Familienname
return
<person>
{if($fam/$Familienname) then <Familienname>{$fam/Familienname/text()}</Familienname>
else()}
</person>
}</abfrage_fam.xml>
```

Der oben abgebildete XQuery-Code fragt alle Personen aus der „*melderegister.xml*" ab, welche den gesuchten Familiennamen beinhalten. Die Variable „*$Familienname*" wird aus der Befehlszeile der Kommandoebene eingelesen.
„*java -cp c:/saxon/saxon9he.jar net.sf.saxon.Query abfrage_fam.xq Familienname="Meier" > abfrage_fam.xml*". Dieser Befehl ruft das Tool Saxon auf und durchsucht mithilfe der „*abfrage_fam.xq*" die Datei „*melderegister.xml*" nach allen Personen mit dem Familiennamen „*Meier*". Das Ergebnis wird in die Datei „*abfrage_fam.xml*" abgelegt:

```
<?xml version="1.0" encoding="UTF-8"?>
<abfrage_fam.xml><person><Familienname>Meier</Familienname></person>
</abfrage_fam.xml>
```

Das Ergebnis enthält die Person mit dem Familiennamen „*Meier*". Über diese Variante, externe Variablen in die XQuery-Datei einzubinden, ist es möglich, spezielle Nutzerabfragen zu erstellen. Diese Abfragen müsste allerdings schon vor der Anfrage des Nutzers erstellt werden, um eine effektive Nutzung der archivierten Datenbank zu gewährleisten.

Die Abfragen können so ganz speziell für die Datenbank und für die Nutzer angepasst werden. Damit erfüllt XQuery die Anforderungen des OAIS-Modells für das Auslieferungspaket.
Der einzige Nachteil besteht darin, dass jemand die nötigen XQuery Kenntnisse besitzen muss, um die wichtigen Abfragen zu erstellen.

3.2.2. Umgang mit Metadaten und inhaltlichen Informationen

Metadaten und inhaltliche Informationen werden im OAIS-Modell gesondert verwaltet und gelagert. Die inhaltlichen Informationen befinden sich nach dem OAIS-Modell im Archivpaket. Metadaten hingegen befinden sich an vielen Stellen im OAIS-Modell, um zum Beispiel bestimmte Informationspakete zu beschreiben. Zum Suchen und Auffinden von digitalen Objekten, nach dem OAIS-Modell, sind Metadaten ebenfalls notwendig. Es gibt mehrere Arten von Metadaten:

- beschreibende Metadaten
- technische Metadaten
- administrative Metadaten
- rechtliche Metadaten
- Struktur- und Archivierungsmetadaten

Dabei unterscheiden sich die einzelnen Arten von Metadaten je nach Datenbank. Demnach gibt es für jede archivierte Datenbank andere Metadaten. Einen Überblick zu unterschiedlichen Metadaten aus der Datenbankarchivierung finden sich in der Formatbeschreibung von SIARD. [37]

37 Die SIARD Formatbeschreibung ist hier zu finden: http://www.bar.admin.ch/dienstleistungen/00823/00825/index.html

Bei SIARD werden die Metadaten in mehreren Ebenen unterteilt:

- Metadaten auf der Ebene Datenbank, Schema, Tabelle und Spalte
- Metadaten des Primär- und Fremdschlüssels
- Referenz-Metadaten
- Metadaten der Kandidatenschlüssel und der Check-Einschränkung
- Metadaten auf der Ebene Trigger, View und Routine
- Metadaten der Parameter
- Metadaten auf der Ebene des Benutzers
- Metadaten auf der Ebene Rolle
- Metadaten auf der Ebene der Privilegien

Diese Metadatenunterteilung lässt sich sehr gut für das OAIS-Modell verwenden. Somit ist es realisierbar, - für die sechs Funktionseinheiten des OAIS-Modells - die passenden Metadaten zu vergeben. Diese könnten in eines der Formate wie PREMIS oder METS implementiert werden.

Auf die genaue Analyse von PREMIS und METS wird in dieser Diplomarbeit nicht eingegangen. Die Erhebung von Metadaten ist ein sehr aufwendiger und zeitintensiver Prozess, welcher je nach Datenbank und Datenbankproduzent angepasst werden muss. Daher sollte dieser Prozess für die Archivierung einer Datenbank großzügig geplant werden.

Im Falle der Testdatenbank Melderegister müssen viele technische Metadaten erhoben werden. Durch den Export nach XML sind zudem schon einige technische und administrative Metadaten entstanden. Diese befinden sich als Kommentare in der melderegister.xml. Diese Kommentare lassen sich mit einem einfachen XQuery Befehl selektieren und in ein anderes Format transportieren.

```
let $m := fn:doc("melderegister.xml")
for $p in $m/<metaInfo>{ //comment()[1] }</metaInfo>
return
$p
```

Die Xquery-Abfrage sucht nach dem ersten Kommentar in der Datei melderegister.xml und gibt diesen aus. Es wäre auch möglich, weitere Kommentare mit Hilfe von XQuery auszugeben, ohne dabei die XML-Datei zu verändern. Diese Option rentiert sich besonders bei größeren XML-Dateien.

Die obige Xquery-Datei wird mit folgendem Befehl ausgeführt:
„*C:\Saxon>java -cp c:/saxon/saxon9he.jar net.sf.saxon.Query kommentar.xq > kommentar.txt*“.

Das folgende Ergebnis wird erzeugt:

```
<?xml version="1.0" encoding="UTF-8"?><metaInfo><!--
-
- phpMyAdmin XML Dump
- version 3.2.4
- http://www.phpmyadmin.net
-
- Host: localhost
- Erstellungszeit: 12. Mai 2010 um 14:36
- Server Version: 5.1.41
- PHP-Version: 5.3.1
--><!-- Tabelle anschrift --></metaInfo>
```

In diesem Fall wurde das .txt Format als Exportformat gewählt. Die nun exportierten Metadaten lassen sich in die XML-Datei für Metadaten eintragen und gegen die Schema Datei von PREMIS oder METS validieren. Die oben dargestellten Metadaten sind hauptsächlich technische Metadaten, welche bei weitem nicht vollständig sind. Die Erhebung der Metadaten erfolgt manuell und benötigt viele Ressourcen.

4. Auswertung und Vorstellung der Prozesskette

Datenbankarchivierung für Archive ist zwingend notwendig für den Bestandserhalt. Allerdings gibt es auf diesem Weg viele Probleme und wenige Lösungen. Für den Export einer relationalen Datenbank wird eine Software benötigt, die eine gute Umsetzung nach XML gewährleistet. Ein gut strukturiertes XML-Dokument ist besonders wichtig, um daraus eine XML-Schema Datei zu generieren. In dieser Schema Datei können einige Metadaten implementiert und andere Weiterverarbeitungen vorgenommen werden. Daher muss eine Software entwickelt bzw. angepasst werden, die eine Datenbank in ein archivgerechtes XML-Format exportieren kann.

Des Weiteren müssen die Abfragen/Views übernommen und in XQuery umgewandelt werden, dieser Prozess benötigt zusätzliche Arbeitskräfte mit XML und XQuery Kenntnissen. Zusätzliche XQuery-Abfragen für das Auslieferungspaket (DIP) müssen erstellt werden. Diese Abfragen müssen den Nutzerbedürfnissen angepasst werden. Zudem muss eine Software wie Saxon vorhanden sein, um XQuery-Dateien zu interpretieren und die Abfragen auszuführen. Saxon ist nur fähig kleine XML-Dokumente zu durchsuchen. Bei größeren XML-Dokumenten wird Saxon zusammenbrechen bzw. die Arbeitszeit des Tools wird immens hoch. Die Server von „*MarkLogic*“[38] könnten möglicherweise das Problem der Laufzeit optimieren. Die Analyse dieses Problems erfordert weitere Tests.

Für die Implementierung von Metadaten ist es möglich, sich an funktionierende Beispiele wie SIARD zu orientieren. Dennoch ist die Erhebung und Speicherung von Metadaten ein langwieriger Prozess. Bei der Erhebung muss auf die unterschiedlichen Formen und Inhalte der Datenbanken eingegangen werden. Bei der Speicherung helfen die Formate PREMIS und METS, welche eine Validierung über XML-Schema und somit fehlerfreie Eingaben erlauben. Damit steigt aber der Arbeitsaufwand bei der Metadaten-Vergabe.

Insgesamt ist die Datenbankarchivierung ein arbeitsaufwendiger Prozess der viel Anpassung, Problemlösungsorientierung und Kundenbezogenheit erfordert.

38 Webseite von MarkLogic: http://www.marklogic.com/

Der so entstehende enorme Arbeitsaufwand setzt weitere Prüfungen voraus. Archive müssen die digitalen Objekte genauestens auf deren Archivwürdigkeit überprüfen.
Die eben erläuterten Probleme und Ansätze zeigen, dass es schwierig ist, eine universelle Prozesskette zu erstellen. Für jede Datenbank müssen andere Probleme bearbeitet werden. Doch folgende Kriterien sind bei jeder Datenbankarchivierung gleich bzw. sollten vorhanden sein:

1. Datenbankexport nach XML:
   - Exportierte XML-Datei mit korrekten Elementbezeichnungen
   - XML-Datei erhält die Tabellenstruktur
   - XML-Datei beinhaltet einige Metadaten
   - XML-Dateigröße ist nicht zu groß

2. Generierung der XML-Schema Datei:
   - Benötigt wird dazu eine brauchbare XML-Datei und Software wie XMLSpy
   - Strukturierung um evtl. keyref einzubinden
   - Speicherung einiger Metadaten in der XML-Schema

3. Datenbankabfragen/Views:
   - Datenbankabfragen müssen von SQL nach XQuery umgewandelt werden
   - Erstellung neuer, für den Benutzer angepasster, XQuery-Abfragen für das Auslieferungspaket
   - Software, welche XQuery interpretieren und ausführen kann

4. Metadaten:
   - Metadaten Format wie PREMIS oder METS
   - Metadaten Formatbeschreibung wie bei beispielsweise SIARD

4.1. Pakete nach dem OAIS-Modell

Durch alle Tests und Auswertungen kommen nun folgende Informationspakete nach dem OAIS-Modell zustande:

4.1.1. Einlieferungspaket (SIP)

Das Einlieferungspaket besteht aus dem digitalen Objekt, das archiviert werden soll. In diesem Fall ist das die Testdatenbank Melderegister. Zu der Datenbank werden Metadaten erhoben, welche die Inhaltsinformationen beschreiben und für eine spätere Interpretierbarkeit der Datenbank notwendig sind. Dazu gehören auch Feldbeschreibungen, sofern die Datenbank mit generischen Feldbeschreibungen versehen wurde. Im Einlieferungspaket befinden sich demnach:

- Metadaten zur Interpretation und Beschreibung der Datenbank sowie der Inhalte
- Datenbankinhalte und Datenbankabfragen, möglicherweise in Form eines Backups
- Datenbankstrukturinformationen
- Feldbeschreibungen

Die Datenbankstrukturinformationen und die Feldbeschreibungen können in unterschiedlichen Formaten vorliegen. Diese werden vorab übernommen und später weiterverarbeitet. Es sollte zudem möglich sein, mit dem Datenbankproduzenten gängige Formate für die Daten des Einlieferungspaketes zu vereinbaren.

Die Testdatenbank Melderegister wird wie folgt in das Einlieferungspaket übernommen:

Metadaten: [39]

- dbName: Melderegister – Beinhaltet Testdatensätze für eine Diplomarbeit.
- description: Datensätze sind für diverse Testreihen vorgesehen.
- archiver: Andreas Schulz
- archiverContact: andreas.schulz@fh-potsdam.de
- dataOwner: Testdatenbank von Andreas Schulz
- dataOriginTimespan: Daten wurden im Rahmen der Diplomarbeit erstellt und entstanden in diesem Zeitraum.
- archivalDate: 01.06.2010
- databaseProduct: MySQL 5.1.41

Datenbankinhalte und Datenbankabfragen:

- Sind vorhanden als MySQL Dump, dieses beinhaltet sowohl die Tabellen mit den Inhalten, als auch die Datenbankabfragen. Der Name der Datei ist melderegister.sql.

Datenbankstrukturinformationen:

- Die Strukturinformationen befinden sich in der Abbildung 4

Feldbeschreibungen:

- In der Datenbank Melderegister gibt es keine generischen Feldbezeichnungen, daher müssen keine zusätzlichen Feldbeschreibungen übernommen werden.

Diese Ausgangsdaten werden in das Einlieferungspaket übernommen. Aus dem Einlieferungspaket wird das Archivpaket generiert.

39 Die Metadaten werden aus der SIARD Formatbeschreibung übernommen und werder hier nicht vollständig aufgeführt.

4.1.2. Archivpaket (AIP)

Das Archivpaket wird aus dem Einlieferungspaket generiert und stellt die Form dar, in der die Datenbank langfristig aufbewahrt wird. Das Format für die langfristige Aufbewahrung ist in diesem Fall XML.
Das Datenbankbackup aus dem Einlieferungspaket muss zuerst wiederhergestellt werden, damit die einzelnen Tools den Export durchführen können. In diesem Fall ist der PHPMyAdmin das Tool, welches die relationale Datenbank nach XML exportiert. Die Abfragen werden beim Export zunächst ignoriert, damit diese weiterhin virtuelle Tabellen bleiben können.
Die exportierte XML-Datei darf die Datenbankstruktur nicht verändern und muss alle notwendigen Primärdaten beinhalten und korrekt darstellen. Denn aus der XML-Datei wird eine XML-Schema Datei generiert, welche zusätzlich die Datenbankstruktur beinhaltet. Allerdings wird diese Struktur nicht die gleiche Struktur enthalten, wie die ehemalige relationale Datenbank. Die Relationen werden wahrscheinlich durch den Export nach XML verloren gehen. Daher müssen entweder keyref eingefügt oder XQuery-Abfragen erstellt werden, um die referenzielle Integrität der Daten wiederherzustellen.
Da die Abfragen der Datenbanken mit Hilfe von umgewandelten XQuery-Abfragen gesichert werden, bietet sich XQuery für die Wiederherstellung der referenziellen Integrität an.
Das Archivpaket wird mit weiteren Metadaten angereichert und die Feldbeschreibungen, sofern vorhanden, werden übernommen. Die Feldbeschreibungen könnten in verschiedenen Formaten im Archivpaket enthalten sein. Eine dieser Möglichkeiten könnte das händische Einfügen der Feldbeschreibungen als Attribute in die XML-Schema Datei sein. Im Archivpaket befinden sich demnach:

- XML-Datei, welche aus der relationalen Datenbank exportiert wird
- XML-Schema Datei, welche aus der XML-Datei generiert wird
- Abfragen, welche manuell in XQuery-Abfragen umgewandelt werden
- Feldbeschreibungen
- Metadaten

Mit Hilfe von PREMIS oder METS ist können XML-Dateien generiert werden um darin Metadaten zu speichern.
Aus dem Einlieferungspaket wird folgendes Archivpaket generiert:

XML-Datei:

- aus der melderegister.sql wird mit Hilfe des Tools PHPMyAdmin die Datei melderegister.xml generiert
- fehlende Elemente, welche in der relationalen Datenbank NULL enthielten, werden manuell nachgetragen

XML-Schema Datei:

- mit Hilfe des Tools Altova XMLSpy wird aus der Datei melderegister.xml eine XML-Schema Datei generiert

Abfragen:

- aus den fünf SQL-Abfragen werden fünf XQuery-Abfragen erstellt

Feldbeschreibungen:

- da es in der Testdatenbank keine Feldbeschreibungen gibt, müssen diese nicht übernommen werden

Metadaten:

- die Metadaten werden in eine XML-Datei implementiert, diese XML-Datei kann gegen METS oder PREMIS validiert werden
- die SIARD Feldbeschreibung liefert eine Vorlage zur Erhebung der Metadaten

Die oben genannten Inhalte bilden das Archivpaket. Die Metadaten aus dem Archivpaket werden zusätzlich für die Datenverwaltung bereitgestellt. Das Archivpaket wird mit Hilfe des Auslieferungspaketes für den Nutzer zugänglich gemacht.

4.1.3. Auslieferungspaket (DIP)

Die relationale Datenbank befindet sich nun als XML-Datei im Archivpaket. Eine Bearbeitung oder Abfragen durch SQL sind nicht mehr möglich. Daher werden XQuery-Abfragen verwendet, um das Archivpaket dem Nutzer zur Verfügung zu stellen. Dabei müssen die Abfragen auf die Bedürfnisse und auf die Interessen der Nutzer angepasst werden. Das heißt, dass im Vorfeld Nutzerabfragen erstellt werden müssen. Dabei können diese Abfragen ganze Inhalte ausgeben, wie beispielsweise die übernommenen Views oder die Abfragen werden mit Variablen, wie in Abschnitt 3.2.1.2. beschrieben, erstellt. Je nachdem um welche Datenbank es sich handelt, ist es möglich, Abfragen nach bestimmten Nutzerrechten zu erstellen. Im Fall der Datenbank Melderegister ist es so möglich, bestimmten Nutzern Abfragen zur Ausgabe von persönlichen Daten zu verbieten. Die vorgefertigten Abfragen könnten also nach Nutzerrechten sortiert und je nach Status des Nutzers freigegeben werden.
Das Archivformat kann ohne Änderungen in das Auslieferungspaket übernommen werden. Einzig die neu erstellten XQuery-Dateien müssen hinzugefügt werden. Das Auslieferungspaket besteht demnach aus:

- Datenbankdaten und dazu gehörige Datenbankstruktur
- Für den Nutzer angepasste XQuery-Abfragen
- Feldbeschreibungen

Die Feldbeschreibungen und die Datenbankstrukturinformationen können dem Nutzer ebenfalls zur Verfügung gestellt werden. Im Falle von Feldbeschreibungen ist dies sogar sinnvoll, damit der Nutzer die einzelnen Feldinhalte verstehen kann. Eine automatisierte Übersetzung der Felder in der Datenbank würde dem Nutzer in diesem Fall helfen. Denkbar wäre auch, dass die XQuery-Abfragen so gestaltet werden, dass die Attribute mit den Feldübersetzungen mit ausgegeben werden. Dafür sollte aber das DBML Format von RODA verwendet werden und ein Attribut für die einzelnen Felder gewählt werden, welches die Klartextnamen der Felder beinhaltet.

Im Fall des Melderegisters befinden sich nun folgende Inhalte im Auslieferungspaket:

Datenbankdaten und Datenbankstruktur:

- XML-Datei melderegister.xml
- Datenbankstruktur als Image Datei

XQuery-Abfragen:

- abfrage_01.xq
- abfrage_02.xq
- abfrage_03.xq
- abfrage_04.xq
- abfrage_05.xq

Feldbeschreibungen:

- in der Testdatenbank Melderegister gibt es keine generischen Feldnamen

Dem Nutzer muss die Software Saxon zugänglich gemacht und die Befehlszeile: „*java -cp c:/saxon/saxon9he.jar net.sf.saxon.Query*“ zur Verfügung gestellt werden. Zudem sollte der Nutzer eine Liste bzw. eine Übersicht zu den möglichen Eingabeoptionen bekommen. Die Ergebnisse der Nutzerabfragen sollten als XML-Dateien ausgegeben und mit einem Webbrowser wiedergegeben werden. Je nach Datenbank wird die Lesbarkeit des ausgegebenen XML-Dokumentes nur beschränkt lesbar sein. Daher sollte in dieser Richtung noch etwas weiter getestet werden, um eine optimale Wiedergabemöglichkeit für den Nutzer zu schaffen. Denkbar wäre das Hinzufügen von Style-Informationen, um die Inhalte als Formular oder ähnliches auszugeben. Diese Style-Informationen müssten aber mit in die Ergebniserzeugung von XQuery eingebunden werden.

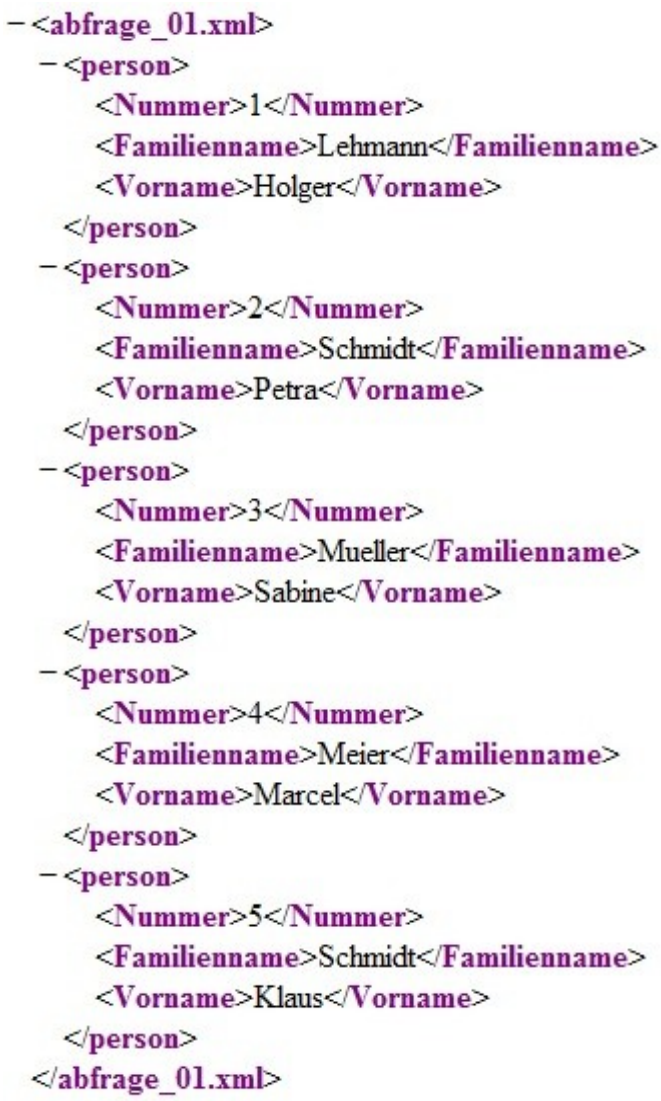


Abbildung 16: abfrage_01 - Ansicht im Mozilla Firefox

Wie Abbildung 16 am Beispiel der abfrage_01.xml verdeutlicht, ist die Ansicht im Webbrowser verständlich und strukturiert. Diese Ansicht lässt sich aber nur mit einer XML-Datei erzeugen, die korrekte Elementbezeichnungen enthält.

Fazit

In dieser Diplomarbeit wurde die Datenbankarchivierung anhand einer selbst erstellten relationalen Datenbank erläutert. Der Archivierungsprozess wurde nach den Kriterien des OAIS-Modells durchgeführt. Dabei wurde die relationale Datenbank mit mehreren Tools nach XML exportiert. Dabei entstanden Probleme, die individuelle Lösungen erforderten. Die Abfragen der Testdatenbank wurden in XQuery-Abfragen umgewandelt, um diese später wieder im Auslieferungspaket für den Nutzer bereitzustellen.

Wie die einzelnen Abschnitte dieser Diplomarbeit zeigen, ist der Prozess der Datenbankarchivierung sehr aufwendig und langwierig. Es ist sehr komplex eine genaue Prozesskette zu definieren, die alle Datenbanken nach dem OIAS-Modell archivieren kann. Jede Datenbank ist unterschiedlich aufgebaut und besitzt unterschiedliche Inhalte. Demnach entstehen während des Archivierungsprozesses immer wieder neue Probleme, die eine individuelle Lösung erfordern.
Die Datenbankarchivierung sollte grundsätzlich von Fachpersonal durchgeführt werden, das über XML, XQuery und Datenbankkenntnisse verfügt, um die entstehenden Herausforderungen sicher zu bewältigen.

Die Analyse der Tools hat gezeigt, dass jede Software ein anderes Ergebnis beim Export der relationalen Datenbank nach XML erzeugt. Des Weiteren konnte keines der Tools eine XML-Datei erschaffen, die eine einfache Weiterverarbeitung gewährleistet und zudem gut strukturiert und lesbar ist. Jedes Resultat muss noch einmal nachbearbeitet werden, um eine brauchbare XML-Datei für das Archivpaket zu erhalten. Das exportierte Ergebnis von PHPMyAdmin zeichnet sich dadurch aus, dass es die geringsten Nachbesserungen für eine Weiterverarbeitung benötigt. Einzig die fehlenden Elemente mussten manuell nachgetragen werden. Die Zellen, die in der relationalen Datenbank den Wert NULL beinhalteten, wurden vom PHPMyAdmin nicht generiert. Das heißt, dass Elemente ohne Inhalte in der XML-Datei fehlen.

Dieses Problem wird zu einem Nachteil bei der Generierung einer XML-Schema Datei aus der XML-Datei.
Allerdings wäre auch ein Export mit einer anderen Software als Lösung denkbar. Für jede Datenbank muss eine individuelle Lösung gefunden werden. Es ist zudem wichtig eine XML-Datei zu erhalten, die das Abfragen mit XQuery ermöglicht.

Diese Diplomarbeit belegt, dass XQuery für die Archivierung von Datenbanken sinnvoll einsetzbar ist.

Die Datenbankabfragen werden mit Hilfe von XQuery erhalten und bleiben funktionsfähig. Außerdem ist es mit XQuery möglich, Metadaten in Form von Kommentaren aus der XML-Datei herauszufiltern und die daraus gewonnenen Informationen weiterzuverarbeiten. Anhand der Abfragen ist es ebenso möglich, die referenzielle Integrität der Datenbank zu bewahren. Die dazu benötigten Informationen können aus den Ergebnissen und aus den Abfragen selbst herausgelesen werden. Mit Hilfe des Tools Saxon ist es realisierbar, die Abfragen für den Nutzer zugänglich zu machen. Dabei können Abfragen nach mehreren Zugriffsberechtigungen und Variablen erschaffen werden. Somit kann das Archiv kontrollieren, welche Informationen spezifische Nutzer abfragen können. Zu diesem Zweck müssen aber individuelle Abfragen erstellt werden. Die Abfrageergebnisse können in XML-Dateien gespeichert und in einem Webbrowser angezeigt werden.

Allerdings ist Saxon keine Software, die sehr große XML-Dokumente durchsuchen kann. Für umfangreiche Datenbanken bedarf es spezieller Server und Software zur Interpretation von XQuery-Abfragen. Es sind zudem weitere Tests mit den nötigen Komponenten notwendig, um eine optimale Abfrage-Zeit zu erreichen.

Für die Vergabe von Metadaten ist es möglich, sich an bereits vorhandene Standards zu orientieren, bzw. diese zu verwenden. So gibt die Formatbeschreibung von SIARD eine Übersicht zur Vergabe von Metadaten her. Die Formate PREMIS und METS eigenen sich gut, um die Metadaten in ein XML-Format zu implementieren. Diese XML-Datei kann gegen die Schema-Datei von METS oder PREMIS validiert werden.

Damit wurden alle Ziele bis auf den Vorschlag einer Prozesskette zur Archivierung einer relationalen Datenbank erreicht. Eine genaue Prozesskette ist aufgrund der Vielfalt der einzelnen Datenbanken schwer zu ermitteln. Allerdings funktioniert die in dieser Diplomarbeit angewandte Prozesskette, bis auf einige Mängel, gut.

Für eine funktionstüchtige Toolkette müssten weitere Datenbanken und Datenbankformate getestet werden. Es ist nicht bekannt, wie sich die Toolkette bei mehreren Gigabyte großen Datenbanken verhält. Genauso konnte nicht getestet werden, wie sich eine Access oder Oracle Datenbank archivieren lässt. Bei diesen Varianten ist es durchaus möglich, dass neue Probleme entstehen würden.

In der Zukunft sollte bessere Software entwickelt werden, welche die Datenbankarchivierung nach dem OAIS-Modell und XML erleichtert sowie standardisiert.

Anhang

Anhang A1:

DBML: Beschreibung der Datenbankstruktur

```
<?xml version="1.0" ?>
<DB>
<STRUCTURE>
<TABLE NAME="products">
<COLUMNS>
<COLUMN NAME="code" TYPE="nvarchar"
SIZE="10" NULL="no"/>
<COLUMN NAME="description" TYPE="nvarchar"
SIZE="50" NULL="no"/>
...
</COLUMNS>
<KEYS>
<PKEY TYPE="simple">
<FIELD NAME="code"/>
</PKEY>
</KEYS>
</TABLE>
<TABLE NAME="p2s">
<COLUMNS>
<COLUMN NAME="cod-p" TYPE="nvarchar"
SIZE="10" NULL="no"/>
<COLUMN NAME="cod-s" TYPE="nvarchar"
SIZE="10" NULL="no"/>
</COLUMNS>
<KEYS>
<PKEY TYPE="composite">
<FIELD NAME="cod-p"/>
<FIELD NAME="cod-s"/>
</PKEY>
<FKEY NAME="cod-p" IN="products"
REF="code"/>
<FKEY NAME="cod-s" IN="suppliers"
REF="code"/>
</KEYS>
</TABLE>
<TABLE NAME="suppliers">
<COLUMNS>
<COLUMN NAME="code" TYPE="nvarchar"
SIZE="10" NULL="no"/>
<COLUMN NAME="name" TYPE="nvarchar"
SIZE="60" NULL="no"/>
...
</COLUMNS>
<KEYS>
<PKEY TYPE="simple">
<FIELD NAME="code"/>
</PKEY>
</KEYS>
</TABLE>
</STRUCTURE>
<DATA>
...
</DATA>
</DB>
```

Anhang A2:

MySQL Dump – erstellt mit der Windows Kommandoebene und dem Befehl:

mysqldump -X -u root -p melderegister person ausweis anschrift person_hat_person person_hat_anschrift > melderegister.xml

```
<?xml version="1.0"?>
<mysqldump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<database name="melderegister">
	<table_structure name="person">
		<field Field="Nummer" Type="int(11)" Null="NO" Key="PRI" Extra="" />
		<field Field="Familienname" Type="char(50)" Null="YES" Key="" Extra="" />
		<field Field="Vorname" Type="char(50)" Null="YES" Key="" Extra="" />
		<field Field="Titel" Type="char(20)" Null="YES" Key="" Extra="" />
		<field Field="Geschlecht" Type="char(20)" Null="YES" Key="" Extra="" />
		<field Field="Staatsangehoerigkeit" Type="char(20)" Null="YES" Key="" Extra="" />
		<field Field="Familienstand" Type="char(20)" Null="YES" Key="" Extra="" />
		<field Field="Geburtsort" Type="char(50)" Null="YES" Key="" Extra="" />
		<field Field="Sterbeort" Type="char(50)" Null="YES" Key="" Extra="" />
		<field Field="Ort_Eheschliessung" Type="char(50)" Null="YES" Key="" Extra="" />
		<field Field="Geburtsdatum" Type="date" Null="YES" Key="" Extra="" />
		<field Field="Sterbedatum" Type="date" Null="YES" Key="" Extra="" />
		<field Field="Datum_Eheschliessung" Type="date" Null="YES" Key="" Extra="" />
		<field Field="Ausweis_Nummer" Type="int(11)" Null="NO" Key="PRI" Extra="" />
		<key Table="person" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1"
Column_name="Nummer" Collation="A" Cardinality="5" Null="" Index_type="BTREE" Comment="" />
		<key Table="person" Non_unique="0" Key_name="PRIMARY" Seq_in_index="2"
Column_name="Ausweis_Nummer" Collation="A" Cardinality="5" Null="" Index_type="BTREE"
Comment="" />
		<key Table="person" Non_unique="1" Key_name="FS_Person_Ausweis"
Seq_in_index="1" Column_name="Ausweis_Nummer" Collation="A" Cardinality="5" Null=""
Index_type="BTREE" Comment="" />
		<options Name="person" Engine="InnoDB" Version="10" Row_format="Compact"
Rows="5" Avg_row_length="3276" Data_length="16384" Max_data_length="0" Index_length="16384"
Data_free="3145728" Create_time="2010-05-10 11:11:18" Collation="latin1_swedish_ci"
Create_options="" Comment="" />
	</table_structure>
	<table_data name="person">
	<row>
		<field name="Nummer">1</field>
		<field name="Familienname">Lehmann</field>
		<field name="Vorname">Holger</field>
		<field name="Titel" xsi:nil="true" />
		<field name="Geschlecht">m</field>
		<field name="Staatsangehoerigkeit">deutsch</field>
		<field name="Familienstand">ledig</field>
		<field name="Geburtsort">Muenchen</field>
		<field name="Sterbeort" xsi:nil="true" />
		<field name="Ort_Eheschliessung" xsi:nil="true" />
		<field name="Geburtsdatum">1985-04-16</field>
		<field name="Sterbedatum" xsi:nil="true" />
		<field name="Datum_Eheschliessung" xsi:nil="true" />
		<field name="Ausweis_Nummer">1</field>
	</row>
	<row>
		<field name="Nummer">2</field>
```

```
            <field name="Familienname">Schmidt</field>
            <field name="Vorname">Petra</field>
            <field name="Titel" xsi:nil="true" />
            <field name="Geschlecht">w</field>
            <field name="Staatsangehoerigkeit">deutsch</field>
            <field name="Familienstand">verheiratet</field>
            <field name="Geburtsort">Berlin</field>
            <field name="Sterbeort" xsi:nil="true" />
            <field name="Ort_Eheschliessung">Berlin</field>
            <field name="Geburtsdatum">1980-08-19</field>
            <field name="Sterbedatum" xsi:nil="true" />
            <field name="Datum_Eheschliessung">2006-06-06</field>
            <field name="Ausweis_Nummer">2</field>
    </row>
    <row>
            <field name="Nummer">3</field>
            <field name="Familienname">Mueller</field>
            <field name="Vorname">Sabine</field>
            <field name="Titel">Dr.</field>
            <field name="Geschlecht">w</field>
            <field name="Staatsangehoerigkeit">deutsch</field>
            <field name="Familienstand">ledig</field>
            <field name="Geburtsort">Wien</field>
            <field name="Sterbeort" xsi:nil="true" />
            <field name="Ort_Eheschliessung" xsi:nil="true" />
            <field name="Geburtsdatum">1972-04-17</field>
            <field name="Sterbedatum" xsi:nil="true" />
            <field name="Datum_Eheschliessung" xsi:nil="true" />
            <field name="Ausweis_Nummer">3</field>
    </row>
    <row>
            <field name="Nummer">4</field>
            <field name="Familienname">Meier</field>
            <field name="Vorname">Marcel</field>
            <field name="Titel" xsi:nil="true" />
            <field name="Geschlecht">m</field>
            <field name="Staatsangehoerigkeit">deutsch</field>
            <field name="Familienstand">ledig</field>
            <field name="Geburtsort">Koeln</field>
            <field name="Sterbeort">Koeln</field>
            <field name="Ort_Eheschliessung" xsi:nil="true" />
            <field name="Geburtsdatum">1918-01-23</field>
            <field name="Sterbedatum">2007-04-25</field>
            <field name="Datum_Eheschliessung" xsi:nil="true" />
            <field name="Ausweis_Nummer">4</field>
    </row>
    <row>
            <field name="Nummer">5</field>
            <field name="Familienname">Schmidt</field>
            <field name="Vorname">Klaus</field>
            <field name="Titel">Dr.</field>
            <field name="Geschlecht">m</field>
            <field name="Staatsangehoerigkeit">deutsch</field>
            <field name="Familienstand">verheiratet</field>
            <field name="Geburtsort">Frankfurt</field>
            <field name="Sterbeort" xsi:nil="true" />
            <field name="Ort_Eheschliessung">Berlin</field>
            <field name="Geburtsdatum">1977-12-14</field>
            <field name="Sterbedatum" xsi:nil="true" />
```

```
                <field name="Datum_Eheschliessung">2006-06-06</field>
                <field name="Ausweis_Nummer">5</field>
        </row>
        </table_data>
        <table_structure name="ausweis">
                <field Field="Nummer" Type="int(11)" Null="NO" Key="PRI" Extra="auto_increment" />
                <field Field="Ausstellungsdatum" Type="date" Null="YES" Key="" Extra="" />
                <field Field="Gueltigkeitsdatum" Type="date" Null="YES" Key="" Extra="" />
                <field Field="Behoerde" Type="char(20)" Null="YES" Key="" Extra="" />
                <key Table="ausweis" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1"
Column_name="Nummer" Collation="A" Cardinality="5" Null="" Index_type="BTREE" Comment="" />
                <options Name="ausweis" Engine="InnoDB" Version="10" Row_format="Compact"
Rows="5" Avg_row_length="3276" Data_length="16384" Max_data_length="0" Index_length="0"
Data_free="3145728" Auto_increment="6" Create_time="2010-05-10 11:11:18"
Collation="latin1_swedish_ci" Create_options="" Comment="" />
        </table_structure>
        <table_data name="ausweis">
        <row>
                <field name="Nummer">1</field>
                <field name="Ausstellungsdatum">2001-04-18</field>
                <field name="Gueltigkeitsdatum">2011-04-18</field>
                <field name="Behoerde">München</field>
        </row>
        <row>
                <field name="Nummer">2</field>
                <field name="Ausstellungsdatum">2010-01-12</field>
                <field name="Gueltigkeitsdatum">2020-01-12</field>
                <field name="Behoerde">Berlin</field>
        </row>
        <row>
                <field name="Nummer">3</field>
                <field name="Ausstellungsdatum">2009-09-22</field>
                <field name="Gueltigkeitsdatum">2019-09-22</field>
                <field name="Behoerde">Hamburg</field>
        </row>
        <row>
                <field name="Nummer">4</field>
                <field name="Ausstellungsdatum">2002-05-02</field>
                <field name="Gueltigkeitsdatum">2012-05-02</field>
                <field name="Behoerde">Koeln</field>
        </row>
        <row>
                <field name="Nummer">5</field>
                <field name="Ausstellungsdatum">2001-12-17</field>
                <field name="Gueltigkeitsdatum">2011-12-17</field>
                <field name="Behoerde">Berlin</field>
        </row>
        </table_data>
        <table_structure name="anschrift">
                <field Field="Nummer" Type="int(11)" Null="NO" Key="PRI" Extra="auto_increment" />
                <field Field="PLZ" Type="int(11)" Null="YES" Key="" Extra="" />
                <field Field="Ort" Type="char(50)" Null="YES" Key="" Extra="" />
                <field Field="Strasse" Type="char(50)" Null="YES" Key="" Extra="" />
                <field Field="Hausnummer" Type="int(11)" Null="YES" Key="" Extra="" />
                <key Table="anschrift" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1"
Column_name="Nummer" Collation="A" Cardinality="5" Null="" Index_type="BTREE" Comment="" />
                <options Name="anschrift" Engine="InnoDB" Version="10" Row_format="Compact"
Rows="5" Avg_row_length="3276" Data_length="16384" Max_data_length="0" Index_length="0"
Data_free="3145728" Auto_increment="6" Create_time="2010-05-10 11:11:18"
```

```
Collation="latin1_swedish_ci" Create_options="" Comment="" />
        </table_structure>
        <table_data name="anschrift">
        <row>
                <field name="Nummer">1</field>
                <field name="PLZ">80469</field>
                <field name="Ort">München</field>
                <field name="Strasse">Rumfordstraße</field>
                <field name="Hausnummer">2</field>
        </row>
        <row>
                <field name="Nummer">2</field>
                <field name="PLZ">10247</field>
                <field name="Ort">Berlin</field>
                <field name="Strasse">Frankfurter Allee</field>
                <field name="Hausnummer">80</field>
        </row>
        <row>
                <field name="Nummer">3</field>
                <field name="PLZ">20354</field>
                <field name="Ort">Hamburg</field>
                <field name="Strasse">Colonnaden</field>
                <field name="Hausnummer">50</field>
        </row>
        <row>
                <field name="Nummer">4</field>
                <field name="PLZ">51149</field>
                <field name="Ort">Koeln</field>
                <field name="Strasse">Ingeborgstrasse</field>
                <field name="Hausnummer">100</field>
        </row>
        <row>
                <field name="Nummer">5</field>
                <field name="PLZ">10247</field>
                <field name="Ort">Berlin</field>
                <field name="Strasse">Frankfurter Allee</field>
                <field name="Hausnummer">80</field>
        </row>
        </table_data>
        <table_structure name="person_hat_person">
                <field Field="Person_Nummer_01" Type="int(11)" Null="NO" Key="PRI" Extra="" />
                <field Field="Person_Nummer_02" Type="int(11)" Null="NO" Key="PRI" Extra="" />
                <field Field="Beziehung" Type="char(20)" Null="YES" Key="" Extra="" />
                <key Table="person_hat_person" Non_unique="0" Key_name="PRIMARY"
Seq_in_index="1" Column_name="Person_Nummer_01" Collation="A" Cardinality="1" Null=""
Index_type="BTREE" Comment="" />
                <key Table="person_hat_person" Non_unique="0" Key_name="PRIMARY"
Seq_in_index="2" Column_name="Person_Nummer_02" Collation="A" Cardinality="1" Null=""
Index_type="BTREE" Comment="" />
                <key Table="person_hat_person" Non_unique="1"
Key_name="FS_Person_hat_Person_Person_01" Seq_in_index="1"
Column_name="Person_Nummer_01" Collation="A" Cardinality="1" Null="" Index_type="BTREE"
Comment="" />
                <key Table="person_hat_person" Non_unique="1"
Key_name="FS_Person_hat_Person_Person_02" Seq_in_index="1"
Column_name="Person_Nummer_02" Collation="A" Cardinality="1" Null="" Index_type="BTREE"
Comment="" />
                <options Name="person_hat_person" Engine="InnoDB" Version="10"
Row_format="Compact" Rows="1" Avg_row_length="16384" Data_length="16384" Max_data_length="0"
```

```
Index_length="32768"  Data_free="3145728"  Create_time="2010-05-10  11:11:19"
Collation="latin1_swedish_ci" Create_options="" Comment="" />
        </table_structure>
        <table_data name="person_hat_person">
        <row>
                <field name="Person_Nummer_01">2</field>
                <field name="Person_Nummer_02">5</field>
                <field name="Beziehung">Ehepaar</field>
        </row>
        </table_data>
        <table_structure name="person_hat_anschrift">
                <field Field="Person_Nummer" Type="int(11)" Null="NO" Key="PRI" Extra="" />
                <field Field="Anschrift_Nummer" Type="int(11)" Null="NO" Key="PRI" Extra="" />
                <field Field="Einzugsdatum" Type="date" Null="YES" Key="" Extra="" />
                <field Field="Auszugsdatum" Type="date" Null="YES" Key="" Extra="" />
                <field Field="Art" Type="char(20)" Null="YES" Key="" Extra="" />
                <key  Table="person_hat_anschrift"  Non_unique="0"  Key_name="PRIMARY"
Seq_in_index="1"  Column_name="Person_Nummer"  Collation="A"  Cardinality="5"  Null=""
Index_type="BTREE" Comment="" />
                <key  Table="person_hat_anschrift"  Non_unique="0"  Key_name="PRIMARY"
Seq_in_index="2"  Column_name="Anschrift_Nummer"  Collation="A"  Cardinality="5"  Null=""
Index_type="BTREE" Comment="" />
                <key  Table="person_hat_anschrift"  Non_unique="1"
Key_name="FS_Person_hat_Anschrift_Person"  Seq_in_index="1"  Column_name="Person_Nummer"
Collation="A" Cardinality="5" Null="" Index_type="BTREE" Comment="" />
                <key  Table="person_hat_anschrift"  Non_unique="1"
Key_name="FS_Person_hat_Anschrift_Anschrift"  Seq_in_index="1"  Column_name="Anschrift_Nummer"
Collation="A" Cardinality="5" Null="" Index_type="BTREE" Comment="" />
                <options  Name="person_hat_anschrift"  Engine="InnoDB"  Version="10"
Row_format="Compact"  Rows="5"  Avg_row_length="3276"  Data_length="16384"  Max_data_length="0"
Index_length="32768"  Data_free="3145728"  Create_time="2010-05-10  11:11:18"
Collation="latin1_swedish_ci" Create_options="" Comment="" />
        </table_structure>
        <table_data name="person_hat_anschrift">
        <row>
                <field name="Person_Nummer">1</field>
                <field name="Anschrift_Nummer">1</field>
                <field name="Einzugsdatum">1998-07-16</field>
                <field name="Auszugsdatum" xsi:nil="true" />
                <field name="Art">Hauptwohnsitz</field>
        </row>
        <row>
                <field name="Person_Nummer">2</field>
                <field name="Anschrift_Nummer">2</field>
                <field name="Einzugsdatum">1999-09-09</field>
                <field name="Auszugsdatum" xsi:nil="true" />
                <field name="Art">Hauptwohnsitz</field>
        </row>
        <row>
                <field name="Person_Nummer">3</field>
                <field name="Anschrift_Nummer">3</field>
                <field name="Einzugsdatum">1998-10-28</field>
                <field name="Auszugsdatum" xsi:nil="true" />
                <field name="Art">Hauptwohnsitz</field>
        </row>
        <row>
                <field name="Person_Nummer">4</field>
                <field name="Anschrift_Nummer">4</field>
                <field name="Einzugsdatum">2010-04-22</field>
```

```
                <field name="Auszugsdatum">2007-04-25</field>
                <field name="Art">Hauptwohnsitz</field>
        </row>
        <row>
                <field name="Person_Nummer">5</field>
                <field name="Anschrift_Nummer">2</field>
                <field name="Einzugsdatum">1999-09-09</field>
                <field name="Auszugsdatum" xsi:nil="true" />
                <field name="Art">Hauptwohnsitz</field>
        </row>
        </table_data>
</database>
</mysqldump>
```

Anhang A3:

Funktion „Exportieren“ von PHPMyAdmin:

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
-
- phpMyAdmin XML Dump
- version 3.2.4
- http://www.phpmyadmin.net
-
- Host: localhost
- Erstellungszeit: 25. Mai 2010 um 16:13
- Server Version: 5.1.41
- PHP-Version: 5.3.1
-->

<!--
- Datenbank: 'melderegister'
-->
<melderegister>
  <!-- Tabelle anschrift -->
    <anschrift>
        <Nummer>1</Nummer>
        <PLZ>80469</PLZ>
        <Ort>München</Ort>
        <Strasse>Rumfordstraße</Strasse>
        <Hausnummer>2</Hausnummer>
    </anschrift>
    <anschrift>
        <Nummer>2</Nummer>
        <PLZ>10247</PLZ>
        <Ort>Berlin</Ort>
        <Strasse>Frankfurter Allee</Strasse>
        <Hausnummer>80</Hausnummer>
    </anschrift>
    <anschrift>
        <Nummer>3</Nummer>
        <PLZ>20354</PLZ>
        <Ort>Hamburg</Ort>
        <Strasse>Colonnaden</Strasse>
        <Hausnummer>50</Hausnummer>
    </anschrift>
```

```
  <anschrift>
    <Nummer>4</Nummer>
    <PLZ>51149</PLZ>
    <Ort>Koeln</Ort>
    <Strasse>Ingeborgstrasse</Strasse>
    <Hausnummer>100</Hausnummer>
  </anschrift>
  <anschrift>
    <Nummer>5</Nummer>
    <PLZ>10247</PLZ>
    <Ort>Berlin</Ort>
    <Strasse>Frankfurter Allee</Strasse>
    <Hausnummer>80</Hausnummer>
  </anschrift>
<!-- Tabelle ausweis -->
  <ausweis>
    <Nummer>1</Nummer>
    <Ausstellungsdatum>2001-04-18</Ausstellungsdatum>
    <Gueltigkeitsdatum>2011-04-18</Gueltigkeitsdatum>
    <Behoerde>München</Behoerde>
  </ausweis>
  <ausweis>
    <Nummer>2</Nummer>
    <Ausstellungsdatum>2010-01-12</Ausstellungsdatum>
    <Gueltigkeitsdatum>2020-01-12</Gueltigkeitsdatum>
    <Behoerde>Berlin</Behoerde>
  </ausweis>
  <ausweis>
    <Nummer>3</Nummer>
    <Ausstellungsdatum>2009-09-22</Ausstellungsdatum>
    <Gueltigkeitsdatum>2019-09-22</Gueltigkeitsdatum>
    <Behoerde>Hamburg</Behoerde>
  </ausweis>
  <ausweis>
    <Nummer>4</Nummer>
    <Ausstellungsdatum>2002-05-02</Ausstellungsdatum>
    <Gueltigkeitsdatum>2012-05-02</Gueltigkeitsdatum>
    <Behoerde>Koeln</Behoerde>
  </ausweis>
  <ausweis>
    <Nummer>5</Nummer>
    <Ausstellungsdatum>2001-12-17</Ausstellungsdatum>
    <Gueltigkeitsdatum>2011-12-17</Gueltigkeitsdatum>
    <Behoerde>Berlin</Behoerde>
  </ausweis>
<!-- Tabelle person -->
  <person>
    <Nummer>1</Nummer>
    <Familienname>Lehmann</Familienname>
    <Vorname>Holger</Vorname>
    <Geschlecht>m</Geschlecht>
    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
    <Familienstand>ledig</Familienstand>
    <Geburtsort>Muenchen</Geburtsort>
    <Geburtsdatum>1985-04-16</Geburtsdatum>
    <Ausweis_Nummer>1</Ausweis_Nummer>
  </person>
  <person>
    <Nummer>2</Nummer>
```

```
    <Familienname>Schmidt</Familienname>
    <Vorname>Petra</Vorname>
    <Geschlecht>w</Geschlecht>
    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
    <Familienstand>verheiratet</Familienstand>
    <Geburtsort>Berlin</Geburtsort>
    <Ort_Eheschliessung>Berlin</Ort_Eheschliessung>
    <Geburtsdatum>1980-08-19</Geburtsdatum>
    <Datum_Eheschliessung>2006-06-06</Datum_Eheschliessung>
    <Ausweis_Nummer>2</Ausweis_Nummer>
  </person>
  <person>
    <Nummer>3</Nummer>
    <Familienname>Mueller</Familienname>
    <Vorname>Sabine</Vorname>
    <Titel>Dr.</Titel>
    <Geschlecht>w</Geschlecht>
    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
    <Familienstand>ledig</Familienstand>
    <Geburtsort>Wien</Geburtsort>
    <Geburtsdatum>1972-04-17</Geburtsdatum>
    <Ausweis_Nummer>3</Ausweis_Nummer>
  </person>
  <person>
    <Nummer>4</Nummer>
    <Familienname>Meier</Familienname>
    <Vorname>Marcel</Vorname>
    <Geschlecht>m</Geschlecht>
    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
    <Familienstand>ledig</Familienstand>
    <Geburtsort>Koeln</Geburtsort>
    <Sterbeort>Koeln</Sterbeort>
    <Geburtsdatum>1918-01-23</Geburtsdatum>
    <Sterbedatum>2007-04-25</Sterbedatum>
    <Ausweis_Nummer>4</Ausweis_Nummer>
  </person>
  <person>
    <Nummer>5</Nummer>
    <Familienname>Schmidt</Familienname>
    <Vorname>Klaus</Vorname>
    <Titel>Dr.</Titel>
    <Geschlecht>m</Geschlecht>
    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
    <Familienstand>verheiratet</Familienstand>
    <Geburtsort>Frankfurt</Geburtsort>
    <Ort_Eheschliessung>Berlin</Ort_Eheschliessung>
    <Geburtsdatum>1977-12-14</Geburtsdatum>
    <Datum_Eheschliessung>2006-06-06</Datum_Eheschliessung>
    <Ausweis_Nummer>5</Ausweis_Nummer>
  </person>
<!-- Tabelle person_hat_anschrift -->
  <person_hat_anschrift>
    <Person_Nummer>1</Person_Nummer>
    <Anschrift_Nummer>1</Anschrift_Nummer>
    <Einzugsdatum>1998-07-16</Einzugsdatum>
    <Art>Hauptwohnsitz</Art>
  </person_hat_anschrift>
  <person_hat_anschrift>
    <Person_Nummer>2</Person_Nummer>
```

```
      <Anschrift_Nummer>2</Anschrift_Nummer>
      <Einzugsdatum>1999-09-09</Einzugsdatum>
      <Art>Hauptwohnsitz</Art>
   </person_hat_anschrift>
   <person_hat_anschrift>
      <Person_Nummer>3</Person_Nummer>
      <Anschrift_Nummer>3</Anschrift_Nummer>
      <Einzugsdatum>1998-10-28</Einzugsdatum>
      <Art>Hauptwohnsitz</Art>
   </person_hat_anschrift>
   <person_hat_anschrift>
      <Person_Nummer>4</Person_Nummer>
      <Anschrift_Nummer>4</Anschrift_Nummer>
      <Einzugsdatum>2010-04-22</Einzugsdatum>
      <Auszugsdatum>2007-04-25</Auszugsdatum>
      <Art>Hauptwohnsitz</Art>
   </person_hat_anschrift>
   <person_hat_anschrift>
      <Person_Nummer>5</Person_Nummer>
      <Anschrift_Nummer>2</Anschrift_Nummer>
      <Einzugsdatum>1999-09-09</Einzugsdatum>
      <Art>Hauptwohnsitz</Art>
   </person_hat_anschrift>
  <!-- Tabelle person_hat_person -->
   <person_hat_person>
      <Person_Nummer_01>2</Person_Nummer_01>
      <Person_Nummer_02>5</Person_Nummer_02>
      <Beziehung>Ehepaar</Beziehung>
   </person_hat_person>
</melderegister>
```

Anhang A4:

Fünf XML Dokumente: exportiert mit DatabaseSpy und den Optionen aus Abbildung 13.

```
<?xml version="1.0" encoding="UTF-8"?>
<Import name="anschrift" query="SELECT `Nummer`, `PLZ`, `Ort`, `Strasse`, `Hausnummer` FROM
`melderegister`.`anschrift` ">
        <!--MySQL-->
        <!--Column name:Nummer type:integer-->
        <!--Column name:PLZ type:integer-->
        <!--Column name:Ort type:char length:50 maxlength:255-->
        <!--Column name:Strasse type:char length:50 maxlength:255-->
        <!--Column name:Hausnummer type:integer-->
        <Row>
                <Nummer>1</Nummer>
                <PLZ>80469</PLZ>
                <Ort>München</Ort>
                <Strasse>Rumfordstraße</Strasse>
                <Hausnummer>2</Hausnummer>
        </Row>
        <Row>
                <Nummer>2</Nummer>
                <PLZ>10247</PLZ>
                <Ort>Berlin</Ort>
```

```
                <Strasse>Frankfurter Allee</Strasse>
                <Hausnummer>80</Hausnummer>
        </Row>
        <Row>
                <Nummer>3</Nummer>
                <PLZ>20354</PLZ>
                <Ort>Hamburg</Ort>
                <Strasse>Colonnaden</Strasse>
                <Hausnummer>50</Hausnummer>
        </Row>
        <Row>
                <Nummer>4</Nummer>
                <PLZ>51149</PLZ>
                <Ort>Koeln</Ort>
                <Strasse>Ingeborgstrasse</Strasse>
                <Hausnummer>100</Hausnummer>
        </Row>
        <Row>
                <Nummer>5</Nummer>
                <PLZ>10247</PLZ>
                <Ort>Berlin</Ort>
                <Strasse>Frankfurter Allee</Strasse>
                <Hausnummer>80</Hausnummer>
        </Row>
</Import>

<?xml version="1.0" encoding="UTF-8"?>
<Import name="ausweis" query="SELECT `Nummer`, `Ausstellungsdatum`, `Gueltigkeitsdatum`,
`Behoerde` FROM `melderegister`.`ausweis` ">
        <!--MySQL-->
        <!--Column name:Nummer type:integer-->
        <!--Column name:Ausstellungsdatum type:date-->
        <!--Column name:Gueltigkeitsdatum type:date-->
        <!--Column name:Behoerde type:char length:20 maxlength:255-->
        <Row>
                <Nummer>1</Nummer>
                <Ausstellungsdatum>2001-04-18</Ausstellungsdatum>
                <Gueltigkeitsdatum>2011-04-18</Gueltigkeitsdatum>
                <Behoerde>München</Behoerde>
        </Row>
        <Row>
                <Nummer>2</Nummer>
                <Ausstellungsdatum>2010-01-12</Ausstellungsdatum>
                <Gueltigkeitsdatum>2020-01-12</Gueltigkeitsdatum>
                <Behoerde>Berlin</Behoerde>
        </Row>
        <Row>
                <Nummer>3</Nummer>
                <Ausstellungsdatum>2009-09-22</Ausstellungsdatum>
                <Gueltigkeitsdatum>2019-09-22</Gueltigkeitsdatum>
                <Behoerde>Hamburg</Behoerde>
        </Row>
        <Row>
                <Nummer>4</Nummer>
                <Ausstellungsdatum>2002-05-02</Ausstellungsdatum>
                <Gueltigkeitsdatum>2012-05-02</Gueltigkeitsdatum>
                <Behoerde>Koeln</Behoerde>
        </Row>
        <Row>
```

```
                <Nummer>5</Nummer>
                <Ausstellungsdatum>2001-12-17</Ausstellungsdatum>
                <Gueltigkeitsdatum>2011-12-17</Gueltigkeitsdatum>
                <Behoerde>Berlin</Behoerde>
        </Row>
</Import>

<?xml version="1.0" encoding="UTF-8"?>
<Import name="person" query="SELECT `Nummer`, `Familienname`, `Vorname`, `Titel`, `Geschlecht`,
`Staatsangehoerigkeit`, `Familienstand`, `Geburtsort`, `Sterbeort`, `Ort_Eheschliessung`, `Geburtsdatum`,
`Sterbedatum`, `Datum_Eheschliessung`, `Ausweis_Nummer` FROM `melderegister`.`person` ">
        <!--MySQL-->
        <!--Column name:Nummer type:integer-->
        <!--Column name:Familienname type:char length:50 maxlength:255-->
        <!--Column name:Vorname type:char length:50 maxlength:255-->
        <!--Column name:Titel type:char length:20 maxlength:255-->
        <!--Column name:Geschlecht type:char length:20 maxlength:255-->
        <!--Column name:Staatsangehoerigkeit type:char length:20 maxlength:255-->
        <!--Column name:Familienstand type:char length:20 maxlength:255-->
        <!--Column name:Geburtsort type:char length:50 maxlength:255-->
        <!--Column name:Sterbeort type:char length:50 maxlength:255-->
        <!--Column name:Ort_Eheschliessung type:char length:50 maxlength:255-->
        <!--Column name:Geburtsdatum type:date-->
        <!--Column name:Sterbedatum type:date-->
        <!--Column name:Datum_Eheschliessung type:date-->
        <!--Column name:Ausweis_Nummer type:integer-->
        <Row>
                <Nummer>1</Nummer>
                <Familienname>Lehmann</Familienname>
                <Vorname>Holger</Vorname>
                <Titel/>
                <Geschlecht>m</Geschlecht>
                <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                <Familienstand>ledig</Familienstand>
                <Geburtsort>Muenchen</Geburtsort>
                <Sterbeort/>
                <Ort_Eheschliessung/>
                <Geburtsdatum>1985-04-16</Geburtsdatum>
                <Sterbedatum/>
                <Datum_Eheschliessung/>
                <Ausweis_Nummer>1</Ausweis_Nummer>
        </Row>
        <Row>
                <Nummer>2</Nummer>
                <Familienname>Schmidt</Familienname>
                <Vorname>Petra</Vorname>
                <Titel/>
                <Geschlecht>w</Geschlecht>
                <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                <Familienstand>verheiratet</Familienstand>
                <Geburtsort>Berlin</Geburtsort>
                <Sterbeort/>
                <Ort_Eheschliessung>Berlin</Ort_Eheschliessung>
                <Geburtsdatum>1980-08-19</Geburtsdatum>
                <Sterbedatum/>
                <Datum_Eheschliessung>2006-06-06</Datum_Eheschliessung>
                <Ausweis_Nummer>2</Ausweis_Nummer>
        </Row>
        <Row>
```

```
                <Nummer>3</Nummer>
                <Familienname>Mueller</Familienname>
                <Vorname>Sabine</Vorname>
                <Titel>Dr.</Titel>
                <Geschlecht>w</Geschlecht>
                <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                <Familienstand>ledig</Familienstand>
                <Geburtsort>Wien</Geburtsort>
                <Sterbeort/>
                <Ort_Eheschliessung/>
                <Geburtsdatum>1972-04-17</Geburtsdatum>
                <Sterbedatum/>
                <Datum_Eheschliessung/>
                <Ausweis_Nummer>3</Ausweis_Nummer>
        </Row>
        <Row>
                <Nummer>4</Nummer>
                <Familienname>Meier</Familienname>
                <Vorname>Marcel</Vorname>
                <Titel/>
                <Geschlecht>m</Geschlecht>
                <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                <Familienstand>ledig</Familienstand>
                <Geburtsort>Koeln</Geburtsort>
                <Sterbeort>Koeln</Sterbeort>
                <Ort_Eheschliessung/>
                <Geburtsdatum>1918-01-23</Geburtsdatum>
                <Sterbedatum>2007-04-25</Sterbedatum>
                <Datum_Eheschliessung/>
                <Ausweis_Nummer>4</Ausweis_Nummer>
        </Row>
        <Row>
                <Nummer>5</Nummer>
                <Familienname>Schmidt</Familienname>
                <Vorname>Klaus</Vorname>
                <Titel>Dr.</Titel>
                <Geschlecht>m</Geschlecht>
                <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                <Familienstand>verheiratet</Familienstand>
                <Geburtsort>Frankfurt</Geburtsort>
                <Sterbeort/>
                <Ort_Eheschliessung>Berlin</Ort_Eheschliessung>
                <Geburtsdatum>1977-12-14</Geburtsdatum>
                <Sterbedatum/>
                <Datum_Eheschliessung>2006-06-06</Datum_Eheschliessung>
                <Ausweis_Nummer>5</Ausweis_Nummer>
        </Row>
</Import>

<?xml version="1.0" encoding="UTF-8"?>
<Import name="person_hat_anschrift" query="SELECT `Person_Nummer`, `Anschrift_Nummer`,
`Einzugsdatum`, `Auszugsdatum`, `Art` FROM `melderegister`.`person_hat_anschrift` ">
        <!--MySQL-->
        <!--Column name:Person_Nummer type:integer-->
        <!--Column name:Anschrift_Nummer type:integer-->
        <!--Column name:Einzugsdatum type:date-->
        <!--Column name:Auszugsdatum type:date-->
        <!--Column name:Art type:char length:20 maxlength:255-->
        <Row>
```

```
                <Person_Nummer>1</Person_Nummer>
                <Anschrift_Nummer>1</Anschrift_Nummer>
                <Einzugsdatum>1998-07-16</Einzugsdatum>
                <Auszugsdatum/>
                <Art>Hauptwohnsitz</Art>
        </Row>
        <Row>
                <Person_Nummer>2</Person_Nummer>
                <Anschrift_Nummer>2</Anschrift_Nummer>
                <Einzugsdatum>1999-09-09</Einzugsdatum>
                <Auszugsdatum/>
                <Art>Hauptwohnsitz</Art>
        </Row>
        <Row>
                <Person_Nummer>3</Person_Nummer>
                <Anschrift_Nummer>3</Anschrift_Nummer>
                <Einzugsdatum>1998-10-28</Einzugsdatum>
                <Auszugsdatum/>
                <Art>Hauptwohnsitz</Art>
        </Row>
        <Row>
                <Person_Nummer>4</Person_Nummer>
                <Anschrift_Nummer>4</Anschrift_Nummer>
                <Einzugsdatum>2010-04-22</Einzugsdatum>
                <Auszugsdatum>2007-04-25</Auszugsdatum>
                <Art>Hauptwohnsitz</Art>
        </Row>
        <Row>
                <Person_Nummer>5</Person_Nummer>
                <Anschrift_Nummer>2</Anschrift_Nummer>
                <Einzugsdatum>1999-09-09</Einzugsdatum>
                <Auszugsdatum/>
                <Art>Hauptwohnsitz</Art>
        </Row>
</Import>

<?xml version="1.0" encoding="UTF-8"?>
<Import name="person_hat_person" query="SELECT `Person_Nummer_01`, `Person_Nummer_02`,
`Beziehung` FROM `melderegister`.`person_hat_person` ">
        <!--MySQL-->
        <!--Column name:Person_Nummer_01 type:integer-->
        <!--Column name:Person_Nummer_02 type:integer-->
        <!--Column name:Beziehung type:char length:20 maxlength:255-->
        <Row>
                <Person_Nummer_01>2</Person_Nummer_01>
                <Person_Nummer_02>5</Person_Nummer_02>
                <Beziehung>Ehepaar</Beziehung>
        </Row>
</Import>
```

Anhang A5:

Zwei XML-Dokumente: exportiert mit DatabaseSpy und den Optionen aus Abbildung 13.

```
<?xml version="1.0" encoding="UTF-8"?>
<Import>
	<anschrift>
		<Nummer>1</Nummer>
		<PLZ>80469</PLZ>
		<Ort>München</Ort>
		<Strasse>Rumfordstraße</Strasse>
		<Hausnummer>2</Hausnummer>
		<person_hat_anschrift>
			<Person_Nummer>1</Person_Nummer>
			<Anschrift_Nummer>1</Anschrift_Nummer>
			<Einzugsdatum>1998-07-16</Einzugsdatum>
			<Auszugsdatum/>
			<Art>Hauptwohnsitz</Art>
		</person_hat_anschrift>
	</anschrift>
	<anschrift>
		<Nummer>2</Nummer>
		<PLZ>10247</PLZ>
		<Ort>Berlin</Ort>
		<Strasse>Frankfurter Allee</Strasse>
		<Hausnummer>80</Hausnummer>
		<person_hat_anschrift>
			<Person_Nummer>2</Person_Nummer>
			<Anschrift_Nummer>2</Anschrift_Nummer>
			<Einzugsdatum>1999-09-09</Einzugsdatum>
			<Auszugsdatum/>
			<Art>Hauptwohnsitz</Art>
		</person_hat_anschrift>
		<person_hat_anschrift>
			<Person_Nummer>5</Person_Nummer>
			<Anschrift_Nummer>2</Anschrift_Nummer>
			<Einzugsdatum>1999-09-09</Einzugsdatum>
			<Auszugsdatum/>
			<Art>Hauptwohnsitz</Art>
		</person_hat_anschrift>
	</anschrift>
	<anschrift>
		<Nummer>3</Nummer>
		<PLZ>20354</PLZ>
		<Ort>Hamburg</Ort>
		<Strasse>Colonnaden</Strasse>
		<Hausnummer>50</Hausnummer>
		<person_hat_anschrift>
			<Person_Nummer>3</Person_Nummer>
			<Anschrift_Nummer>3</Anschrift_Nummer>
			<Einzugsdatum>1998-10-28</Einzugsdatum>
			<Auszugsdatum/>
			<Art>Hauptwohnsitz</Art>
		</person_hat_anschrift>
	</anschrift>
	<anschrift>
```

```
                <Nummer>4</Nummer>
                <PLZ>51149</PLZ>
                <Ort>Koeln</Ort>
                <Strasse>Ingeborgstrasse</Strasse>
                <Hausnummer>100</Hausnummer>
                <person_hat_anschrift>
                        <Person_Nummer>4</Person_Nummer>
                        <Anschrift_Nummer>4</Anschrift_Nummer>
                        <Einzugsdatum>2010-04-22</Einzugsdatum>
                        <Auszugsdatum>2007-04-25</Auszugsdatum>
                        <Art>Hauptwohnsitz</Art>
                </person_hat_anschrift>
        </anschrift>
        <anschrift>
                <Nummer>5</Nummer>
                <PLZ>10247</PLZ>
                <Ort>Berlin</Ort>
                <Strasse>Frankfurter Allee</Strasse>
                <Hausnummer>80</Hausnummer>
        </anschrift>
</Import>


<?xml version="1.0" encoding="UTF-8"?>
<Import name="ausweis">
        <!--MySQL-->
        <!--Column name:Nummer type:int-->
        <!--Column name:Ausstellungsdatum type:date-->
        <!--Column name:Gueltigkeitsdatum type:date-->
        <!--Column name:Behoerde type:char length:20 maxlength:255-->
        <ausweis>
                <Nummer>1</Nummer>
                <Ausstellungsdatum>2001-04-18</Ausstellungsdatum>
                <Gueltigkeitsdatum>2011-04-18</Gueltigkeitsdatum>
                <Behoerde>München</Behoerde>
                <person>
                        <Nummer>1</Nummer>
                        <Familienname>Lehmann</Familienname>
                        <Vorname>Holger</Vorname>
                        <Titel/>
                        <Geschlecht>m</Geschlecht>
                        <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                        <Familienstand>ledig</Familienstand>
                        <Geburtsort>Muenchen</Geburtsort>
                        <Sterbeort/>
                        <Ort_Eheschliessung/>
                        <Geburtsdatum>1985-04-16</Geburtsdatum>
                        <Sterbedatum/>
                        <Datum_Eheschliessung/>
                        <Ausweis_Nummer>1</Ausweis_Nummer>
                        <person_hat_anschrift>
                                <Person_Nummer>1</Person_Nummer>
                                <Anschrift_Nummer>1</Anschrift_Nummer>
                                <Einzugsdatum>1998-07-16</Einzugsdatum>
                                <Auszugsdatum/>
                                <Art>Hauptwohnsitz</Art>
                        </person_hat_anschrift>
                </person>
        </ausweis>
```

```
<ausweis>
        <Nummer>2</Nummer>
        <Ausstellungsdatum>2010-01-12</Ausstellungsdatum>
        <Gueltigkeitsdatum>2020-01-12</Gueltigkeitsdatum>
        <Behoerde>Berlin</Behoerde>
        <person>
                <Nummer>2</Nummer>
                <Familienname>Schmidt</Familienname>
                <Vorname>Petra</Vorname>
                <Titel/>
                <Geschlecht>w</Geschlecht>
                <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                <Familienstand>verheiratet</Familienstand>
                <Geburtsort>Berlin</Geburtsort>
                <Sterbeort/>
                <Ort_Eheschliessung>Berlin</Ort_Eheschliessung>
                <Geburtsdatum>1980-08-19</Geburtsdatum>
                <Sterbedatum/>
                <Datum_Eheschliessung>2006-06-06</Datum_Eheschliessung>
                <Ausweis_Nummer>2</Ausweis_Nummer>
                <person_hat_anschrift>
                        <Person_Nummer>2</Person_Nummer>
                        <Anschrift_Nummer>2</Anschrift_Nummer>
                        <Einzugsdatum>1999-09-09</Einzugsdatum>
                        <Auszugsdatum/>
                        <Art>Hauptwohnsitz</Art>
                </person_hat_anschrift>
                <person_hat_person>
                        <Person_Nummer_01>2</Person_Nummer_01>
                        <Person_Nummer_02>5</Person_Nummer_02>
                        <Beziehung>Ehepaar</Beziehung>
                </person_hat_person>
        </person>
</ausweis>
<ausweis>
        <Nummer>3</Nummer>
        <Ausstellungsdatum>2009-09-22</Ausstellungsdatum>
        <Gueltigkeitsdatum>2019-09-22</Gueltigkeitsdatum>
        <Behoerde>Hamburg</Behoerde>
        <person>
                <Nummer>3</Nummer>
                <Familienname>Mueller</Familienname>
                <Vorname>Sabine</Vorname>
                <Titel>Dr.</Titel>
                <Geschlecht>w</Geschlecht>
                <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                <Familienstand>ledig</Familienstand>
                <Geburtsort>Wien</Geburtsort>
                <Sterbeort/>
                <Ort_Eheschliessung/>
                <Geburtsdatum>1972-04-17</Geburtsdatum>
                <Sterbedatum/>
                <Datum_Eheschliessung/>
                <Ausweis_Nummer>3</Ausweis_Nummer>
                <person_hat_anschrift>
                        <Person_Nummer>3</Person_Nummer>
                        <Anschrift_Nummer>3</Anschrift_Nummer>
                        <Einzugsdatum>1998-10-28</Einzugsdatum>
                        <Auszugsdatum/>
```

```
                        <Art>Hauptwohnsitz</Art>
                    </person_hat_anschrift>
            </person>
    </ausweis>
    <ausweis>
            <Nummer>4</Nummer>
            <Ausstellungsdatum>2002-05-02</Ausstellungsdatum>
            <Gueltigkeitsdatum>2012-05-02</Gueltigkeitsdatum>
            <Behoerde>Koeln</Behoerde>
            <person>
                    <Nummer>4</Nummer>
                    <Familienname>Meier</Familienname>
                    <Vorname>Marcel</Vorname>
                    <Titel/>
                    <Geschlecht>m</Geschlecht>
                    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                    <Familienstand>ledig</Familienstand>
                    <Geburtsort>Koeln</Geburtsort>
                    <Sterbeort>Koeln</Sterbeort>
                    <Ort_Eheschliessung/>
                    <Geburtsdatum>1918-01-23</Geburtsdatum>
                    <Sterbedatum>2007-04-25</Sterbedatum>
                    <Datum_Eheschliessung/>
                    <Ausweis_Nummer>4</Ausweis_Nummer>
                    <person_hat_anschrift>
                            <Person_Nummer>4</Person_Nummer>
                            <Anschrift_Nummer>4</Anschrift_Nummer>
                            <Einzugsdatum>2010-04-22</Einzugsdatum>
                            <Auszugsdatum>2007-04-25</Auszugsdatum>
                            <Art>Hauptwohnsitz</Art>
                    </person_hat_anschrift>
            </person>
    </ausweis>
    <ausweis>
            <Nummer>5</Nummer>
            <Ausstellungsdatum>2001-12-17</Ausstellungsdatum>
            <Gueltigkeitsdatum>2011-12-17</Gueltigkeitsdatum>
            <Behoerde>Berlin</Behoerde>
            <person>
                    <Nummer>5</Nummer>
                    <Familienname>Schmidt</Familienname>
                    <Vorname>Klaus</Vorname>
                    <Titel>Dr.</Titel>
                    <Geschlecht>m</Geschlecht>
                    <Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit>
                    <Familienstand>verheiratet</Familienstand>
                    <Geburtsort>Frankfurt</Geburtsort>
                    <Sterbeort/>
                    <Ort_Eheschliessung>Berlin</Ort_Eheschliessung>
                    <Geburtsdatum>1977-12-14</Geburtsdatum>
                    <Sterbedatum/>
                    <Datum_Eheschliessung>2006-06-06</Datum_Eheschliessung>
                    <Ausweis_Nummer>5</Ausweis_Nummer>
                    <person_hat_anschrift>
                            <Person_Nummer>5</Person_Nummer>
                            <Anschrift_Nummer>2</Anschrift_Nummer>
                            <Einzugsdatum>1999-09-09</Einzugsdatum>
                            <Auszugsdatum/>
                            <Art>Hauptwohnsitz</Art>
```

```
                    </person_hat_anschrift>
                </person>
            </ausweis>
</Import>
```

Anhang A6:

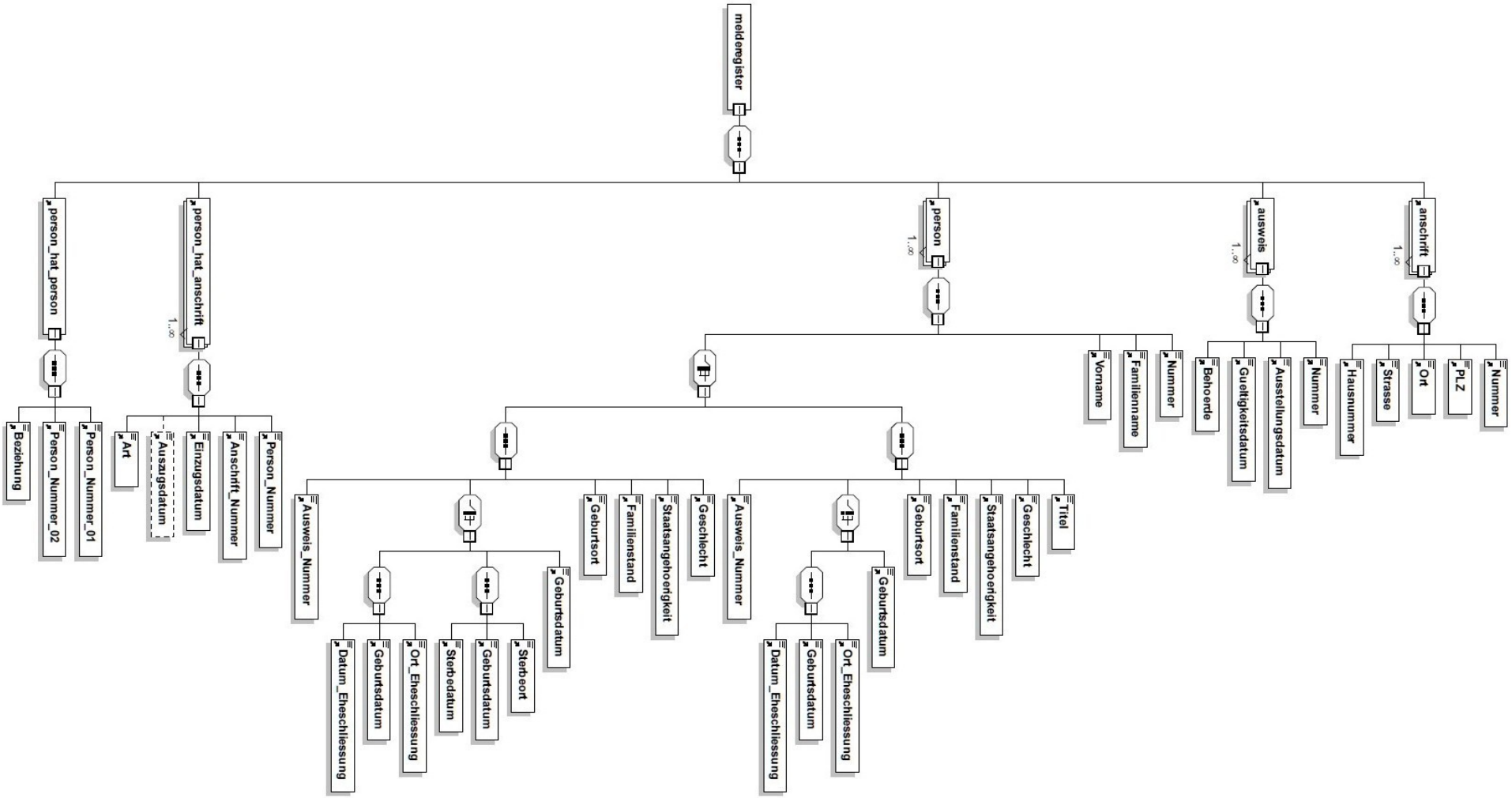
Vergleich der einzelnen Exportvarrianten:

	Kommandoebene mit MySQL	PHPMyAdmin	DatabaseSpy "XML" Export	DatabaseSpy "XML Structure" Export
Anzahl der exportierten Dateien	1	1	5	2
Größe der Dateien in KB	13,3	6,24	7,84	6,96
Bezeichnung vom Wurzelelement Anhand der Tabelle Person	<row>	<person>	<Row>	<person>
Bezeichnung vom Unterelement Anhand der Tabelle Person	<field name="Nummer">	<Nummer>	<Nummer>	<Nummer>
Strukturinformationen und Datentypen	Vorhanden – mit im XML Code eingebunden	Nicht vorhanden	Vorhanden – als Kommentar eingebunden	Nur sehr gering Vorhanden
Umgang mit NULL	Verwendet xsi:nil="true" für NULL	Elemente werden nicht erzeugt	Leere Elemente werden erzeugt	Leere Elemente werden erzeugt
Abbildung der Tabellen Struktur in XML	Bildet die Struktur ab	Bildet die Struktur ab	Bildet die Struktur ab	Verändert die Strukturen
Metadaten wie Erstellungsdatum	Vorhanden	Wenige Vorhanden	Keine Vorhanden	Keine Vorhanden
Export Optionen einstellbar	Ja, wenige	Nein	Ja, einige	Ja, einige
Kosten der Software	Kostenfrei	Kostenfrei	ca. 149€ [40]	ca. 149€

40 Quelle: Preisliste von Altova https://shop.altova.com/pricelist.asp

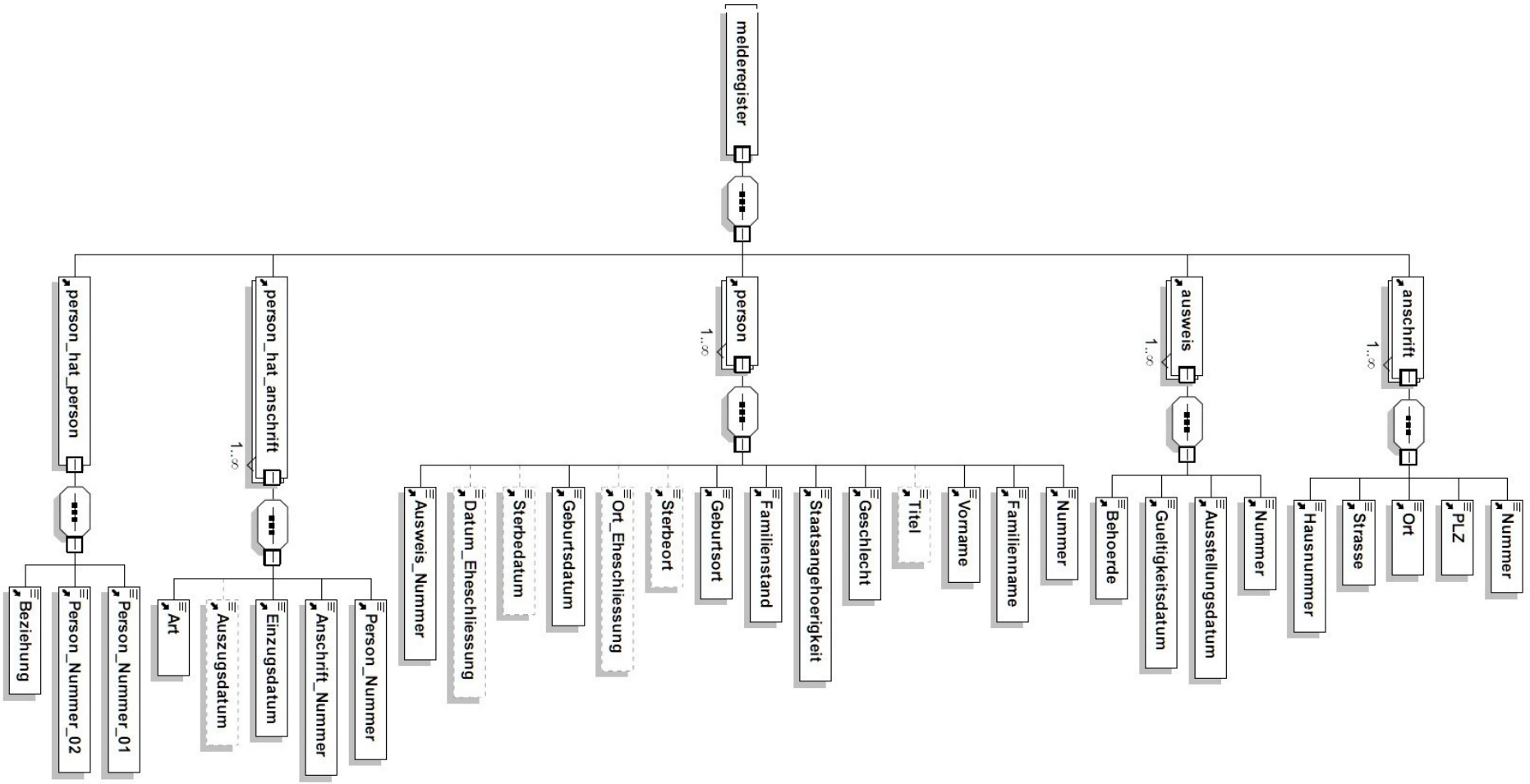
Anhang A7:

XML-Schema Datei, generiert aus der melderegister.xml, welche von PHPMyAdmin exportiert wurde. Schema/WSDL Ansicht:



Anhang A8:

XML-Schema Datei, generiert aus der veränderten melderegister.xml. Schema/WSDL Ansicht:



Anhang A9:

XML-Schema Datei, generiert aus der veränderten melderegister.xml. Quelltext:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSpy v2008 rel. 2 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="person_hat_person">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="Person_Nummer_01"/>
				<xs:element ref="Person_Nummer_02"/>
				<xs:element ref="Beziehung"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="person_hat_anschrift">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="Person_Nummer"/>
				<xs:element ref="Anschrift_Nummer"/>
				<xs:element ref="Einzugsdatum"/>
				<xs:element ref="Auszugsdatum" minOccurs="0"/>
				<xs:element ref="Art"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="person">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="Nummer"/>
				<xs:element ref="Familienname"/>
				<xs:element ref="Vorname"/>
				<xs:element ref="Titel"/>
				<xs:element ref="Geschlecht"/>
				<xs:element ref="Staatsangehoerigkeit"/>
				<xs:element ref="Familienstand"/>
				<xs:element ref="Geburtsort"/>
				<xs:element ref="Sterbeort"/>
				<xs:element ref="Ort_Eheschliessung"/>
				<xs:element ref="Geburtsdatum"/>
				<xs:element ref="Sterbedatum"/>
				<xs:element ref="Datum_Eheschliessung"/>
				<xs:element ref="Ausweis_Nummer"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="melderegister">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="anschrift" maxOccurs="unbounded"/>
				<xs:element ref="ausweis" maxOccurs="unbounded"/>
				<xs:element ref="person" maxOccurs="unbounded"/>
				<xs:element ref="person_hat_anschrift" maxOccurs="unbounded"/>
				<xs:element ref="person_hat_person"/>
			</xs:sequence>
		</xs:complexType>
```

```
</xs:element>
<xs:element name="ausweis">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="Nummer"/>
			<xs:element ref="Ausstellungsdatum"/>
			<xs:element ref="Gueltigkeitsdatum"/>
			<xs:element ref="Behoerde"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>
<xs:element name="anschrift">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="Nummer"/>
			<xs:element ref="PLZ"/>
			<xs:element ref="Ort"/>
			<xs:element ref="Strasse"/>
			<xs:element ref="Hausnummer"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>
<xs:element name="Vorname">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Holger"/>
			<xs:enumeration value="Klaus"/>
			<xs:enumeration value="Marcel"/>
			<xs:enumeration value="Petra"/>
			<xs:enumeration value="Sabine"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="Titel">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value=""/>
			<xs:enumeration value="Dr."/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="Strasse">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Colonnaden"/>
			<xs:enumeration value="Frankfurter Allee"/>
			<xs:enumeration value="Ingeborgstrasse"/>
			<xs:enumeration value="Rumfordstraße"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="Sterbeort">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value=""/>
			<xs:enumeration value="Koeln"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

```
<xs:element name="Sterbedatum">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value=""/>
                        <xs:enumeration value="2007-04-25"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Staatsangehoerigkeit">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="deutsch"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Person_Nummer_02">
        <xs:simpleType>
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="5"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Person_Nummer_01">
        <xs:simpleType>
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="2"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Person_Nummer">
        <xs:simpleType>
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="2"/>
                        <xs:enumeration value="3"/>
                        <xs:enumeration value="4"/>
                        <xs:enumeration value="5"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="PLZ">
        <xs:simpleType>
                <xs:restriction base="xs:int">
                        <xs:enumeration value="10247"/>
                        <xs:enumeration value="20354"/>
                        <xs:enumeration value="51149"/>
                        <xs:enumeration value="80469"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Ort_Eheschliessung">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value=""/>
                        <xs:enumeration value="Berlin"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Ort">
```

```
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Berlin"/>
                    <xs:enumeration value="Hamburg"/>
                    <xs:enumeration value="Koeln"/>
                    <xs:enumeration value="München"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Nummer">
            <xs:simpleType>
                <xs:restriction base="xs:byte">
                    <xs:enumeration value="1"/>
                    <xs:enumeration value="2"/>
                    <xs:enumeration value="3"/>
                    <xs:enumeration value="4"/>
                    <xs:enumeration value="5"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Hausnummer">
            <xs:simpleType>
                <xs:restriction base="xs:byte">
                    <xs:enumeration value="100"/>
                    <xs:enumeration value="2"/>
                    <xs:enumeration value="50"/>
                    <xs:enumeration value="80"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Gueltigkeitsdatum">
            <xs:simpleType>
                <xs:restriction base="xs:date">
                    <xs:enumeration value="2011-04-18"/>
                    <xs:enumeration value="2011-12-17"/>
                    <xs:enumeration value="2012-05-02"/>
                    <xs:enumeration value="2019-09-22"/>
                    <xs:enumeration value="2020-01-12"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Geschlecht">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="m"/>
                    <xs:enumeration value="w"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Geburtsort">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Berlin"/>
                    <xs:enumeration value="Frankfurt"/>
                    <xs:enumeration value="Koeln"/>
                    <xs:enumeration value="Muenchen"/>
                    <xs:enumeration value="Wien"/>
                </xs:restriction>
            </xs:simpleType>
```

```
</xs:element>
<xs:element name="Geburtsdatum">
        <xs:simpleType>
                <xs:restriction base="xs:date">
                        <xs:enumeration value="1918-01-23"/>
                        <xs:enumeration value="1972-04-17"/>
                        <xs:enumeration value="1977-12-14"/>
                        <xs:enumeration value="1980-08-19"/>
                        <xs:enumeration value="1985-04-16"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Familienstand">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="ledig"/>
                        <xs:enumeration value="verheiratet"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Familienname">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Lehmann"/>
                        <xs:enumeration value="Meier"/>
                        <xs:enumeration value="Mueller"/>
                        <xs:enumeration value="Schmidt"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Einzugsdatum">
        <xs:simpleType>
                <xs:restriction base="xs:date">
                        <xs:enumeration value="1998-07-16"/>
                        <xs:enumeration value="1998-10-28"/>
                        <xs:enumeration value="1999-09-09"/>
                        <xs:enumeration value="2010-04-22"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Datum_Eheschliessung">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value=""/>
                        <xs:enumeration value="2006-06-06"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Beziehung">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Ehepaar"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Behoerde">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Berlin"/>
```

```
                                <xs:enumeration value="Hamburg"/>
                                <xs:enumeration value="Koeln"/>
                                <xs:enumeration value="München"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="Auszugsdatum">
                <xs:simpleType>
                        <xs:restriction base="xs:date">
                                <xs:enumeration value="2007-04-25"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="Ausweis_Nummer">
                <xs:simpleType>
                        <xs:restriction base="xs:byte">
                                <xs:enumeration value="1"/>
                                <xs:enumeration value="2"/>
                                <xs:enumeration value="3"/>
                                <xs:enumeration value="4"/>
                                <xs:enumeration value="5"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="Ausstellungsdatum">
                <xs:simpleType>
                        <xs:restriction base="xs:date">
                                <xs:enumeration value="2001-04-18"/>
                                <xs:enumeration value="2001-12-17"/>
                                <xs:enumeration value="2002-05-02"/>
                                <xs:enumeration value="2009-09-22"/>
                                <xs:enumeration value="2010-01-12"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="Art">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumeration value="Hauptwohnsitz"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="Anschrift_Nummer">
                <xs:simpleType>
                        <xs:restriction base="xs:byte">
                                <xs:enumeration value="1"/>
                                <xs:enumeration value="2"/>
                                <xs:enumeration value="3"/>
                                <xs:enumeration value="4"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
</xs:schema>
```

Anhang A10:

Diese XQuery-Abfragen wurden aus den SQL Abfragen von Kapitel 2.1.4. erstellt:

abfrage_01.xq:

```
<abfrage_01.xml>{
let $m := fn:doc("melderegister.xml")
for $p in $m//person
order by $p/Nummer
return
<person>{$p/Nummer, $p/Familienname, $p/Vorname}</person>
}</abfrage_01.xml>
```

abfrage_02.xq:

```
<abfrage_02.xml>{
for $person in doc("melderegister.xml")/melderegister/person
let $hatAnschrift := doc("melderegister.xml")/melderegister/person_hat_anschrift[Person_Nummer=$person/Nummer]
let $anschrift := doc("melderegister.xml")/melderegister/anschrift[Nummer=$person/Nummer]
order by $person/Familienname, $person/Vorname
return
<person>
<Familienname>{$person/Familienname/text()}</Familienname><Vorname>{$person/Vorname/text()}</Vorname>
<Einzugsdatum>{$hatAnschrift/Einzugsdatum/text()}</Einzugsdatum>
{if($hatAnschrift/Auszugsdatum) then <Auszugsdatum>{$hatAnschrift/Auszugsdatum/text()}</Auszugsdatum> else()}
<anschrift>{$anschrift/PLZ, $anschrift/Ort, $anschrift/Strasse, $anschrift/Hausnummer}</anschrift>
</person>
}</abfrage_02.xml>
```

abfrage_03.xq:

```
<abfrage_03.xml>{
for $person in doc("melderegister.xml")/melderegister/person
let $ausweis := doc("melderegister.xml")/melderegister/ausweis[Nummer=$person/Ausweis_Nummer]
let $anschrift := doc("melderegister.xml")/melderegister/anschrift[Nummer=$person/Nummer]
order by $person/Familienname, $person/Vorname
return
<person>
<Familienname>{$person/Familienname/text()}</Familienname><Vorname>{$person/Vorname/text()}</Vorname>
<ausweis>{$ausweis/Gueltigkeitsdatum, $ausweis/Behoerde, $ausweis/Ausstellungsdatum}</ausweis>
<anschrift>{$anschrift/PLZ, $anschrift/Ort}</anschrift>
</person>
}</abfrage_03.xml>
```

abfrage_04.xq:

```
<abfrage_04.xml>{
for $person in doc("melderegister.xml")/melderegister/person
let $hatPerson := doc("melderegister.xml")/melderegister/person_hat_person[Person_Nummer_01=$person/Nummer]
let $bezugsperson := doc("melderegister.xml")/melderegister/person[Nummer=$hatPerson/Person_Nummer_02]
return(
if($bezugsperson) then
(<person>{
($person/Familienname, $person/Vorname, $person/Geburtsdatum),
<person_hat_person>{$hatPerson/Beziehung}
</person_hat_person>,
($bezugsperson/Familienname, $bezugsperson/Vorname, $person/Geburtsdatum),
<person_hat_person>{$hatPerson/Beziehung}
</person_hat_person>
}</person>) else())
}</abfrage_04.xml>
```

abfrage_05.xq:

```
<abfrage_05.xml>{
for $person in doc("melderegister.xml")/melderegister/person
let $anschrift := doc("melderegister.xml")/melderegister/anschrift[Nummer=$person/Nummer]
let $ausweis := doc("melderegister.xml")/melderegister/ausweis[Nummer=$person/Ausweis_Nummer]
order by $person/Familienname, $person/Vorname
return
<person>
<Familienname>{$person/Familienname/text()}</Familienname><Vorname>{$person/Vorname/text()}</Vorname>
<Staatsangehoerigkeit>{$person/Staatsangehoerigkeit/text()}</Staatsangehoerigkeit>
<Geburtsort>{$person/Geburtsort/text()}</Geburtsort>
<Geburtsdatum>{$person/Geburtsdatum/text()}</Geburtsdatum>
<Ausweis_Nummer>{$person/Ausweis_Nummer/text()}</Ausweis_Nummer>
<anschrift>{$anschrift/PLZ, $anschrift/Ort, $anschrift/Strasse, $anschrift/Hausnummer}</anschrift>
<ausweis>{$ausweis/Ausstellungsdatum, $ausweis/Gueltigkeitsdatum, $ausweis/Behoerde}</ausweis>
</person>
}</abfrage_05.xml>
```

Anhang A11:

Die Ergebnisse von den XQuery-Abfragen und den SQL-Abfragen werden gegenübergestellt:

abfrage_01.xml:

<?xml version="1.0" encoding="UTF-8"?
><abfrage_01.xml><person><Nummer>1</Nummer><Familienname>Lehmann</Familienname><Vorname>Holger</Vorname></person><person><Nummer>2</Nummer><Familien
name>Schmidt</Familienname><Vorname>Petra</Vorname></person><person><Nummer>3</Nummer><Familienname>Mueller</Familienname><Vorname>Sabine</Vorname></p
erson><person><Nummer>4</Nummer><Familienname>Meier</Familienname><Vorname>Marcel</Vorname></person><person><Nummer>5</Nummer><Familienname>Schmidt<
/Familienname><Vorname>Klaus</Vorname></person></abfrage_01.xml>

Nummer	Familienname	Vorname
1	Lehmann	Holger
2	Schmidt	Petra
3	Mueller	Sabine
4	Meier	Marcel
5	Schmidt	Klaus

abfrage_02.xml:

<?xml version="1.0" encoding="UTF-8"?><abfrage_02.xml><person><Familienname>Lehmann</Familienname><Vorname>Holger</Vorname><Einzugsdatum>1998-07-
16</Einzugsdatum><anschrift><PLZ>80469</PLZ><Ort>München</Ort><Strasse>Rumfordstraße</Strasse><Hausnummer>2</Hausnummer></anschrift></person><person><Famili
enname>Meier</Familienname><Vorname>Marcel</Vorname><Einzugsdatum>2010-04-22</Einzugsdatum><Auszugsdatum>2007-04-
25</Auszugsdatum><anschrift><PLZ>51149</PLZ><Ort>Koeln</Ort><Strasse>Ingeborgstrasse</Strasse><Hausnummer>100</Hausnummer></anschrift></person><person><Famil
ienname>Mueller</Familienname><Vorname>Sabine</Vorname><Einzugsdatum>1998-10-
28</Einzugsdatum><anschrift><PLZ>20354</PLZ><Ort>Hamburg</Ort><Strasse>Colonnaden</Strasse><Hausnummer>50</Hausnummer></anschrift></person><person><Familie
nname>Schmidt</Familienname><Vorname>Klaus</Vorname><Einzugsdatum>1999-09-09</Einzugsdatum><anschrift><PLZ>10247</PLZ><Ort>Berlin</Ort><Strasse>Frankfurter
Allee</Strasse><Hausnummer>80</Hausnummer></anschrift></person><person><Familienname>Schmidt</Familienname><Vorname>Petra</Vorname><Einzugsdatum>1999-09-
09</Einzugsdatum><anschrift><PLZ>10247</PLZ><Ort>Berlin</Ort><Strasse>Frankfurter
Allee</Strasse><Hausnummer>80</Hausnummer></anschrift></person></abfrage_02.xml>

Familienname	Vorname	Einzugsdatum	Auszugsdatum	PLZ	Ort	Strasse	Hausnummer
Lehmann	Holger	1998-07-16	*NULL*	80469	München	Rumfordstraße	2
Schmidt	Petra	1999-09-09	*NULL*	10247	Berlin	Frankfurter Allee	80
Mueller	Sabine	1998-10-28	*NULL*	20354	Hamburg	Colonnaden	50
Meier	Marcel	2010-04-22	2007-04-25	51149	Koeln	Ingeborgstrasse	100
Schmidt	Klaus	1999-09-09	*NULL*	10247	Berlin	Frankfurter Allee	80

abfrage_03.xml:

<?xml version="1.0" encoding="UTF-8"?><abfrage_03.xml><person><Familienname>Lehmann</Familienname><Vorname>Holger</Vorname><ausweis><Gueltigkeitsdatum>2011-04-18</Gueltigkeitsdatum><Behoerde>München</Behoerde><Ausstellungsdatum>2001-04-18</Ausstellungsdatum></ausweis><anschrift><PLZ>80469</PLZ><Ort>München</Ort></anschrift></person><person><Familienname>Meier</Familienname><Vorname>Marcel</Vorname><ausweis><Gueltigkeitsdatum>2012-05-02</Gueltigkeitsdatum><Behoerde>Koeln</Behoerde><Ausstellungsdatum>2002-05-02</Ausstellungsdatum></ausweis><anschrift><PLZ>51149</PLZ><Ort>Koeln</Ort></anschrift></person><person><Familienname>Mueller</Familienname><Vorname>Sabine</Vorname><ausweis><Gueltigkeitsdatum>2019-09-22</Gueltigkeitsdatum><Behoerde>Hamburg</Behoerde><Ausstellungsdatum>2009-09-22</Ausstellungsdatum></ausweis><anschrift><PLZ>20354</PLZ><Ort>Hamburg</Ort></anschrift></person><person><Familienname>Schmidt</Familienname><Vorname>Klaus</Vorname><ausweis><Gueltigkeitsdatum>2011-12-17</Gueltigkeitsdatum><Behoerde>Berlin</Behoerde><Ausstellungsdatum>2001-12-17</Ausstellungsdatum></ausweis><anschrift><PLZ>10247</PLZ><Ort>Berlin</Ort></anschrift></person><person><Familienname>Schmidt</Familienname><Vorname>Petra</Vorname><ausweis><Gueltigkeitsdatum>2020-01-12</Gueltigkeitsdatum><Behoerde>Berlin</Behoerde><Ausstellungsdatum>2010-01-12</Ausstellungsdatum></ausweis><anschrift><PLZ>10247</PLZ><Ort>Berlin</Ort></anschrift></person></abfrage_03.xml>

Familienname	Vorname	Gueltigkeitsdatum	Behoerde	Ausstellungsdatum	PLZ	Ort
Lehmann	Holger	2011-04-18	München	2001-04-18	80469	München
Schmidt	Petra	2020-01-12	Berlin	2010-01-12	10247	Berlin
Mueller	Sabine	2019-09-22	Hamburg	2009-09-22	20354	Hamburg
Meier	Marcel	2012-05-02	Koeln	2002-05-02	51149	Koeln
Schmidt	Klaus	2011-12-17	Berlin	2001-12-17	10247	Berlin

abfrage_04.xml:

<?xml version="1.0" encoding="UTF-8"?><abfrage_04.xml><person><Familienname>Schmidt</Familienname><Vorname>Petra</Vorname><Geburtsdatum>1980-08-19</Geburtsdatum><person_hat_person><Beziehung>Ehepaar</Beziehung></person_hat_person><Familienname>Schmidt</Familienname><Vorname>Klaus</Vorname><Geburtsdatum>1980-08-19</Geburtsdatum><person_hat_person><Beziehung>Ehepaar</Beziehung></person_hat_person></person></abfrage_04.xml>

Familienname	Vorname	Beziehung	Geburtsdatum
Schmidt	Petra	Ehepaar	1980-08-19
Schmidt	Klaus	Ehepaar	1977-12-14

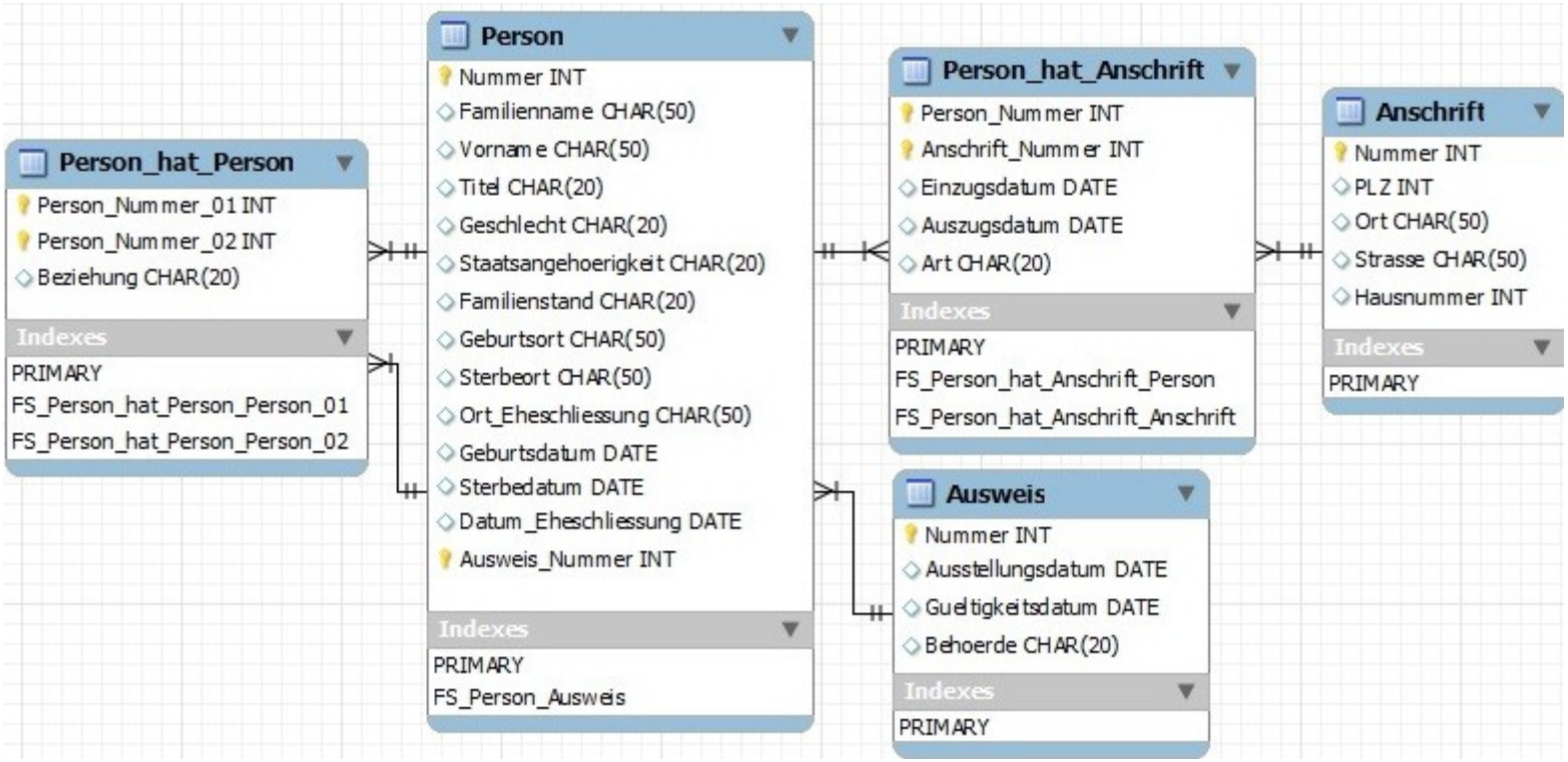
abfrage_05.xml:

<?xml version="1.0" encoding="UTF-8"?
><abfrage_05.xml><person><Familienname>Lehmann</Familienname><Vorname>Holger</Vorname><Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit><Geburtsort>Muenchen</Geburtsort><Geburtsdatum>1985-04-16</Geburtsdatum><Ausweis_Nummer>1</Ausweis_Nummer><anschrift><PLZ>80469</PLZ><Ort>München</Ort><Strasse>Rumfordstraße</Strasse><Hausnummer>2</Hausnummer></anschrift><ausweis><Ausstellungsdatum>2001-04-18</Ausstellungsdatum><Gueltigkeitsdatum>2011-04-18</Gueltigkeitsdatum><Behoerde>München</Behoerde></ausweis></person><person><Familienname>Meier</Familienname><Vorname>Marcel</Vorname><Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit><Geburtsort>Koeln</Geburtsort><Geburtsdatum>1918-01-23</Geburtsdatum><Ausweis_Nummer>4</Ausweis_Nummer><anschrift><PLZ>51149</PLZ><Ort>Koeln</Ort><Strasse>Ingeborgstrasse</Strasse><Hausnummer>100</Hausnummer></anschrift><ausweis><Ausstellungsdatum>2002-05-02</Ausstellungsdatum><Gueltigkeitsdatum>2012-05-02</Gueltigkeitsdatum><Behoerde>Koeln</Behoerde></ausweis></person><person><Familienname>Mueller</Familienname><Vorname>Sabine</Vorname><Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit><Geburtsort>Wien</Geburtsort><Geburtsdatum>1972-04-17</Geburtsdatum><Ausweis_Nummer>3</Ausweis_Nummer><anschrift><PLZ>20354</PLZ><Ort>Hamburg</Ort><Strasse>Colonnaden</Strasse><Hausnummer>50</Hausnummer></anschrift><ausweis><Ausstellungsdatum>2009-09-22</Ausstellungsdatum><Gueltigkeitsdatum>2019-09-22</Gueltigkeitsdatum><Behoerde>Hamburg</Behoerde></ausweis></person><person><Familienname>Schmidt</Familienname><Vorname>Klaus</Vorname><Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit><Geburtsort>Frankfurt</Geburtsort><Geburtsdatum>1977-12-14</Geburtsdatum><Ausweis_Nummer>5</Ausweis_Nummer><anschrift><PLZ>10247</PLZ><Ort>Berlin</Ort><Strasse>Frankfurter Allee</Strasse><Hausnummer>80</Hausnummer></anschrift><ausweis><Ausstellungsdatum>2001-12-17</Ausstellungsdatum><Gueltigkeitsdatum>2011-12-17</Gueltigkeitsdatum><Behoerde>Berlin</Behoerde></ausweis></person><person><Familienname>Schmidt</Familienname><Vorname>Petra</Vorname><Staatsangehoerigkeit>deutsch</Staatsangehoerigkeit><Geburtsort>Berlin</Geburtsort><Geburtsdatum>1980-08-19</Geburtsdatum><Ausweis_Nummer>2</Ausweis_Nummer><anschrift><PLZ>10247</PLZ><Ort>Berlin</Ort><Strasse>Frankfurter Allee</Strasse><Hausnummer>80</Hausnummer></anschrift><ausweis><Ausstellungsdatum>2010-01-12</Ausstellungsdatum><Gueltigkeitsdatum>2020-01-12</Gueltigkeitsdatum><Behoerde>Berlin</Behoerde></ausweis></person></abfrage_05.xml>

Familienname	Vorname	Staatsangehoerigkeit	Geburtsort	Geburtsdatum	Ausweis_Nummer	PLZ	Ort	Strasse	Hausnummer	Ausstellungsdatum	Gueltigkeitsdatum	Behoerde
Lehmann	Holger	deutsch	Muenchen	1985-04-16	1	80469	München	Rumfordstraße	2	2001-04-18	2011-04-18	München
Schmidt	Petra	deutsch	Berlin	1980-08-19	2	10247	Berlin	Frankfurter Allee	80	2010-01-12	2020-01-12	Berlin
Mueller	Sabine	deutsch	Wien	1972-04-17	3	20354	Hamburg	Colonnaden	50	2009-09-22	2019-09-22	Hamburg
Meier	Marcel	deutsch	Koeln	1918-01-23	4	51149	Koeln	Ingeborgstrasse	100	2002-05-02	2012-05-02	Koeln
Schmidt	Klaus	deutsch	Frankfurt	1977-12-14	5	10247	Berlin	Frankfurter Allee	80	2001-12-17	2011-12-17	Berlin

Anhang A12:

Datenbank Melderegister – Datenbankstruktur:



Literaturverzeichnis

Hochschulschriften

1) Glöde, Julia:
Archivierung relationaler Datenbanken auf der Grundlage von XML.
Potsdam, Fachhochschule Potsdam – Informationswissenschaften,
Diplomarbeit, 2009

2) Schwarz, Karin; Däßler, Rolf:
Archivierung und dauerhafte Nutzung von Datenbankinhalten aus
Fachverfahren – eine neue Herausforderung für die digitale Archivierung.
Potsdam, 2010- unveröffentlicht, S. 6

Internetquellen

1) Altova:
DatabaseSpy: http://swarchive.altova.com/de/databasespy.html
(letzter Zugriff am 19.06.2010)

2) consultative committee for space data systems (CCSDS):
Reference Model for an Open Archival Information System (OAIS):
http://public.ccsds.org/publications/archive/650x0b1.pdf
(letzter Zugriff am 19.06.2010)

3) Hewlett Packard:
Database Archiving Software:
http://h71028.www7.hp.com/enterprise/w1/en/software/information-management-governance-ediscovery-database-archiving.html
(letzter Zugriff am 19.06.2010)
Database Archiving Software: Data sheet:
http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA1-9833EEW.pdf
(letzter Zugriff am 19.06.2010)

4) ikeep:
CSP CHRONOS Funktionalität: http://ikeep.com/dba_chronos.pc
(letzter Zugriff am 19.06.2010)

5) International Organization for Standardization:
ISO 14721:2003:
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24683 (letzter Zugriff am 19.06.2010)

6) Koordinationsstelle für die dauerhafte Archivierung elektronischer Unterlagen (KOST):
SIARD: http://www.kost-ceco.ch/wiki/whelp/KaD/index.php?Id=http://www.kost-ceco.ch/wiki/whelp/KaD/pages/SIARD.html
(letzter Zugriff am 19.06.2010)

7) MarkLogic:
http://www.marklogic.com/ (letzter Zugriff am 19.06.2010)

8) METS:
http://www.loc.gov/standards/mets/ (letzter Zugriff am 19.06.2010)

9) Microsoft Library:
Bindungsunterstützung für das xsi:nil-Attribut:
http://msdn.microsoft.com/de-de/library/ybce7f69%28VS.80%29.aspx
(letzter Zugriff am 19.06.2010)

10) nestor:
Digitale Langzeitarchivierung:
http://www.langzeitarchivierung.de/ueber_uns/lza.htm
(letzter Zugriff am 19.06.2010)

11) PREMIS:
http://www.loc.gov/standards/premis/ (letzter Zugriff am 19.06.2010)

12) Ronald Bourret:
XML and Databases:
http://www.rpbourret.com/xml/XMLAndDatabases.htm
(letzter Zugriff am 19.06.2010)

13) Schematron:
http://www.schematron.com/ (letzter Zugriff am 19.06.2010)

14) Schweizerische Eidgenossenschaft:
Datenbankarchivierung: SIARD Suite:
http://www.bar.admin.ch/dienstleistungen/00823/00825/index.html?lang=de (letzter Zugriff am 19.06.2010)
Das OAIS-Modell:
http://www.nb.admin.ch/nb_professionnel/01693/01696/01876/01878/index.html?lang=de (letzter Zugriff am 19.06.2010)

15) TEIA AG:
XML und Datenbanken:
http://www.teialehrbuch.de/Kostenlose-Kurse/XML/7911-Einleitung.html
(letzter Zugriff am 19.06.2010)

16) University of Minho:
Relational Database Preservation through XML modelling:
http://repositorium.sdum.uminho.pt/bitstream/1822/7120/1/EML2007-final.pdf (letzter Zugriff am 19.06.2010)

Monografien

1) Däßler, Rolf:
Das Einsteigerseminar MySQL 5. Heidelberg : bhv, 2005, ISBN-10 3-8266-7397-2

2) Kazakos, Wassilios; Schmidt, Andreas; Tomczyk, Peter:
XML und Datenbanken: Konzepte, Anwendungen, Systeme. Berlin: Springer, 2002, ISBN-10: 354041956X

3) Klettke, Meike; Meyer, Holger:
XML & Datenbanken.1. Auflage, Heidelberg: dpunkt.verlag, 2003, ISBN 3-89864-148-1, S.58

4) Lehner, Wolfgang; Schöningen, Harald:
Xquery – Grundlagen und fortgeschrittene Methoden. 1. Auflage, Heidelberg: dpunkt.verlag, 2004, ISBN 3-89864-266-6, S. 2

5) Regionales Rechenzentrum für Niedersachsen (Hrsg.):XML 1.1 : Grundlagen. 6., unveränd. Aufl. Hannover: Leibniz Universität, 2006

Sammelwerkbeiträge

1) Schwens, Ute; Liegmann, Hans:
Langzeitarchivierung digitaler Ressourcen. In: Rainer Kuhlen [u. a.] (Hrsg.): Handbuch zur Einführung in die Informationswissenschaft und -praxis. 5., völlig neu gefasste Ausg.. München : Saur, 2004, S. 567-570 (Grundlagen der praktischen Information und Dokumentation, Bd. 1)

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.
Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Potsdam, 27. Juni 2010

Andreas Schulz