# XML – ein Format zur dauerhaften Aufbewahrung und Nutzung einfacher, relationaler Datenbanken?

Von Björn Dehms, Andreas Engel und Ulrich Meyer

Die wachsende Verbreitung elektronischer Aufzeichnungen erfordert die Entwicklung von organisatorisch-technischen Verfahren, mit denen auch diese Unterlagen ausgesondert, dauerhaft aufbewahrt, gesichert und genutzt werden können. Während für elektronische Akten bereits Verfahren entwickelt wurden,[1] existieren zur dauerhaften Aufbewahrung, Sicherung und Nutzung von Datenbanken bislang noch keine neuen Ansätze, die über die klassische Flat-file-Archivierung hinausgehen.[2] Dabei scheint die Sicherung von Datenbank-Informationen zeitlich gesehen das drängendere Problem zu sein. Datenbanken sind bereits seit vielen Jahren im Einsatz, sie werden in der Regel kontinuierlich fortgeführt und aktualisiert, so dass gespeicherte Informationen schnell verloren gehen, wenn sie nicht rechtzeitig gesichert werden.

An der Forschungsstelle für Verwaltungsinformatik der Universität Koblenz wurde deshalb mit Untersuchungen begonnen, wie auch Datenbanken auf Dauer gesichert, aufbewahrt und genutzt werden können. In einem ersten Schritt wurde daher im Rahmen einer Diplomarbeit ein softwaretechnisches Verfahren zur dauerhaften elektronischen Aussonderung, Aufbewahrung, Sicherung und Nutzung *einfacher, relationaler Datenbanken unter Verwendung von XML als Archivierungs- und Nutzungsformat* entwickelt. Als Kooperationspartner für die Entwicklung der Verfahrenslösung wirkte das Bundesarchiv mit,[3] aus dessen Bestand eine Datei über „Grenzzwischenfälle an den Grenzen der DDR“ bereitgestellt wurde. Das im Rahmen der Studie entwickelte Migrationsprogramm ist jedoch von der Beispielanwendung unabhängig und kann auch für die Migration beliebiger tabellenartiger Datenbanken eingesetzt werden, sofern sie nur als ASCII-Flat-file aufbereitet werden können. Im vorliegenden Beitrag wird das entwickelte Migrationsprogramm vorgestellt und eine Bewertung von XML als Archivierungs- und Nutzungsformat vorgenommen.

## 1. Übersicht über das entwickelte Verfahren zur Migration von einfachen, relationalen Datenbanken nach XML

Der Migrationsprozess einer einfachen, relationalen Datenbank nach XML kann aus softwaretechnischer Sicht in zwei Phasen eingeteilt werden, die Vorbereitungs- und die eigentliche Migrationsphase, wobei die Phasenübergänge durch die jeweils bearbeiteten Dateien beschrieben werden können. Die am Migrationsprozess beteiligten Dateien lassen sich bezüglich ihrer Funktion in Eingabedateien, Zwischendateien und Ausgabedateien gruppieren. Die folgende Abbildung gibt einen Überblick über den Ablauf des Migrationsprozesses.

---

[1] Vgl. Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt) (Hrsg.), Konzept zur Aussonderung elektronischer Akten, Bonn 1998 (Schriftenreihe der KBSt 40). – Engel, Andreas, Elektronische Archivierung und Nutzung (ELAN) – Eine Projektidee zur Entwicklung von Verfahren für die dauerhafte Archivierung und Nutzung von elektronischen Geschäftsunterlagen aus Behörden und Betrieben. In: Ulrich Nieß (Hrsg.), Auf der Suche nach archivischen Lösungsstrategien im digitalen Zeitalter. Beiträge zur 4. Jahrestagung des Arbeitskreises „Archivierung von Unterlagen aus digitalen Systemen“ im Stadtarchiv Mannheim, 10.–11.4.2000 (Sonderveröffentlichungen des Stadtarchivs Mannheim 26), Mannheim 2001, S. 73–84.

[2] Michael Wettengel, Archivierung digitaler Datenbestände aus der DDR nach der Wiedervereinigung. In: Udo Schäfer – Nicole Bickhoff (Hrsg.), Archivierung elektronischer Unterlagen, Stuttgart 1999, S. 223–239

[3] Seitens des Bundesarchivs waren Frau Bettina Martin-Weber, Ulf Rathje und Dr. Michael Wettengel beteiligt.

Ausgangspunkt der Migration ist eine Datenbank, die in der Regel in einem proprietären Datenbankspeicherformat vorliegt. Damit das entwickelte Migrationsprogramm möglichst unabhängig von bestimmten Datenbankformaten eingesetzt werden kann, wird vorausgesetzt, dass die Datenbankinformationen als ASCII-Flat-file exportiert wurden, was von den gebräuchlichen Datenbankmanagement-Systemen durch eine Standardexport-Funktion unterstützt wird. Dabei muss jeder Datensatz der Datenbanktabelle eine Zeile des ASCII-Flat-Files ausfüllen. Die einzelnen Felder eines Datensatzes können formatiert oder durch Trennzeichen voneinander separiert werden. Werden keine Trennzeichen verwendet, unterstützt das Migrationsprogramm die Erfassung der Feldbeschreibungen (im wesentlichen die Länge der einzelnen Felder) per Hand.

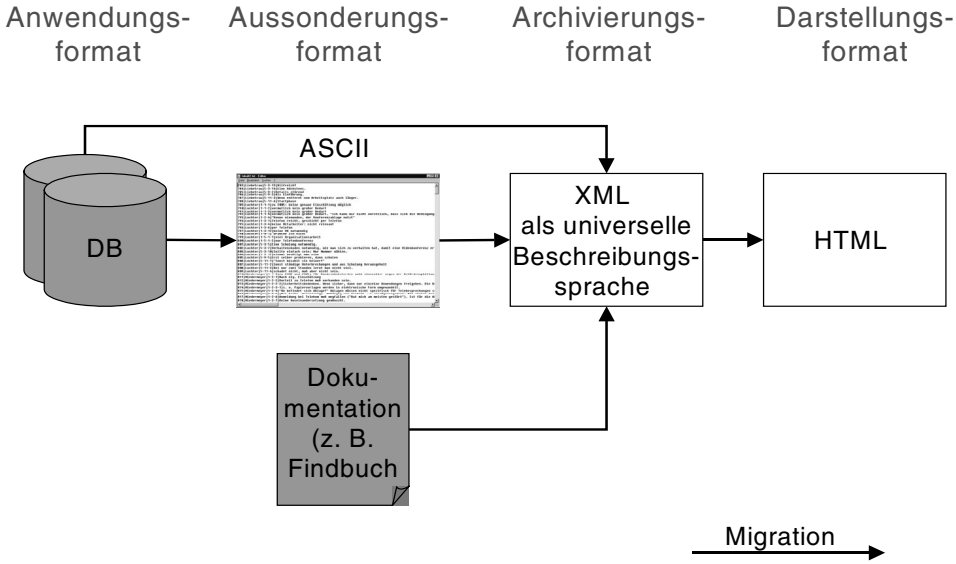


**Abbildung 1: Überblick zum Ablauf des Migrationsprozesses**

Häufig werden Informationen in Datenbanken in kodierter Form abgespeichert, so dass zusätzlich zur Datenbank auch ein Schlüsselverzeichnis (meist in nicht elektronischer Form) vorliegt, mit dem die Datenbankinformationen interpretiert werden können. Auch ein evtl. vorliegendes Schlüsselverzeichnis muss in das Migrationsverfahren einbezogen werden. Zu diesem Zweck muss für das entwickelte Migrationsverfahren eine Schlüssel-DTD erfasst werden. Die Schlüssel-DTD beschreibt die Klartexte der Schlüsselfelder, damit zur Präsentation der Daten auf die Klartexte der verschlüsselten Informationen zurückgegriffen werden kann. Datenbank-Schlüssel werden in der Schlüssel-DTD als Entitätsnamen, der zum Schlüssel gehörende Klartext als Inhalt der Entität abgebildet. Abb. 2 beschreibt die Dateien im Migrationsprozess.

Ergebnis des Migrationsprozesses sind drei Dateien im Archivierungsformat. Den wichtigsten Part spielt hierbei das XML-Dokument, das aus dem ASCII-Flat-File hervorgeht. Aus unverschlüsselten Feldern werden die Klartexte aus dem ASCII-Flat-file unverändert in das XML-Dokument übertragen, bei verschlüsselten Datenbankfeldern werden die Schlüssel des ASCII-Flat-Files im XML-Dokument durch Entitätsreferenzen ersetzt, die auf die in der Schlüssel-DTD befindlichen Entitäten verweisen. Der Browser ersetzt diese Entitätsreferenzen bei der Darstellung des XML-Dokuments automatisch durch den Inhalt der entsprechenden Entitäten, d. h. den Klartext.

Die dritte Ergebnisdatei des Migrationsprozesses ist die Erschließungsdatendatei. Sie enthält die Erschließungsdaten zur archivierten Datenbank, die vor dem Beginn des Migrationsverfahrens über Tastatur eingegeben werden müssen. XML-Dokument, Schlüssel-DTD und Erschließungsdatendatei sind über XML-Link-Strukturen miteinander verknüpft, so dass sie

bei Betrachtung mit einem XML-fähigen Webbrowser als ein logisches Dokument erscheinen und auch von anderen Applikationen als solches verarbeitet werden können.

Anzumerken ist noch, dass alle drei Daten ohne weitere Software von einem Standardbrowser dargestellt werden können. Bei Verwendung spezieller Algorithmen zur Umsetzung der Schlüssel in Klartext müssten diese unter Umständen an veränderte Hard-/Softwareumgebungen angepasst werden.

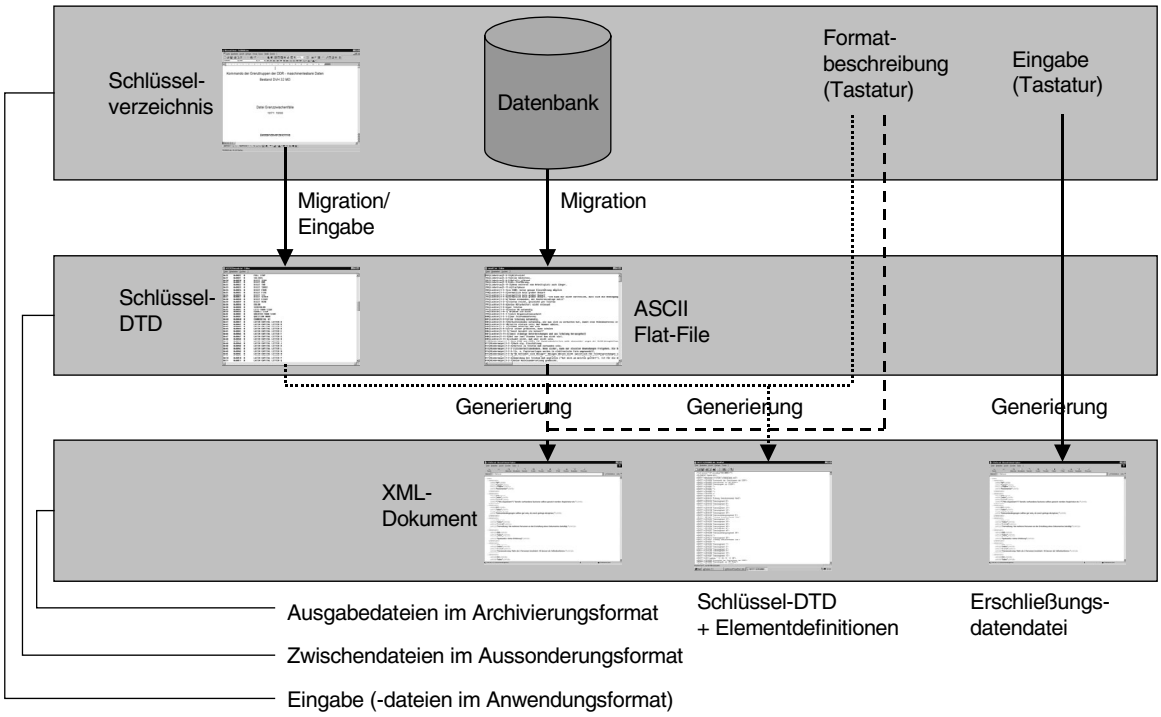


**Abbildung 2: Ablauf des Migrationsprozesses mit den beteiligten Dateien**

## 2. Grundentscheidung zur Migration von Datenbank-Inhalt in ein XML-Format

Zur Speicherung von Informationen aus einfachen, relationalen Datenbanken unter Zuhilfenahme der Extensible Markup Language bieten sich zwei Möglichkeiten an. Bei der ersten Variante werden die Feldinhalte der Datenbank als Inhalt eines Elements abgespeichert. Diese sind Kindelemente eines einen einzelnen Datensatz einrahmenden Elements, z. B.:

```
<Datensatz>
<Feld_1>
6105
</Feld_1>
<Feld_2>
000
</Feld_2>
[...]
</Datensatz>
```

Bei der zweiten Variante werden die Feldinhalte der Datenbank in Attributen abgelegt. Die Attribute gehören zu einem Element, das einen ganzen Datensatz der Datenbank aufnimmt, z.B.:

```
<Datensatz Feld_1="6105" Feld_2="000" [...] ></Datensatz>
```

Diese Lösung besitzt gegenüber der ersten Variante einen geringeren Plattenspeicherbedarf aufgrund der Verwendung einer kleineren Zahl von XML-Konstrukten zur Beschreibung der Primärinformationen. Hieraus resultiert wiederum ein geringerer Hauptspeicherbedarf bei der

Anzeige der Daten: Beim Laden in den Speicher werden XML-Dokumente intern in eine Baumstruktur gewandelt, wobei sich die zweite Variante im Hauptspeicher nicht so stark vergrößert wie Variante 1. Zudem befinden sich bei der Speicherung der Datenbankinformationen als Attributinhalt alle Primärinformationen eines Datensatzes in einer Zeile des XML-Dokuments, was auch von der äußeren Form weitestgehend dem Format des ASCII-Flat-Files entspricht.

Unverschlüsselte Feldinhalte werden bei beiden Varianten unverändert in das XML-Dokument übernommen. Zur Darstellung verschlüsselter Datenbankinhalte mit XML musste jedoch eine Lösung entworfen werden, mit der sich eine Umsetzung der Schlüssel in Klartexte auf einfache Weise vollziehen lässt. Verschiedene Alternativen mit Vor- und Nachteilen werden von Dehms[4] beschrieben, an dieser Stelle wird nur die schließlich verwendete Version vorgestellt: Bei dieser werden die von XML zur Verfügung gestellten Entitäten verwendet, die in der entwickelten Lösung als Entitätsnamen den Feldnamen aus der Datenbank gefolgt vom Schlüssel und als Inhalt den dem Schlüssel entsprechenden Klartext besitzen, z.B.

<!ENTITY Feld1_1101 " Überfall mit Schußwaffen">

Im XML-Dokument wird anstelle des Schlüssels eine Entitätsreferenz eingetragen, die der XML-Parser durch den Inhalt der Entität, d. h. den Klartext, ersetzt. Zur Anzeige der Klartexte ist auf diese Weise kein Programmieraufwand mehr zu leisten. Sämtliche Entitäten werden in der zum XML-Dokument gehörenden Document Type Definition, der Schlüssel-DTD, definiert. Dies ist notwendig, da sich auf die in einem proprietären Format vorliegende Originaldatei des Schlüsselverzeichnisses noch nicht mit den von XML zur Verfügung gestellten Funktionen zugreifen lässt. Die Generierung der Schlüssel-DTD aus dem Schlüsselverzeichnis kann jedoch nur in geringem Umfang automatisiert werden! Wie die Schlüssel-DTD erzeugt wird, bleibt letztlich dem Archivar überlassen. Nicht zu unterschätzen ist dabei der Authentizitätsverlust, der sich durch die Umwandlung des Schlüsselverzeichnisses in die Schlüssel-DTD ergibt. Verschiedene Probleme sind bei Dehms[5] ausführlich beschrieben, von denen eines hier exemplarisch angeführt wird:

Im ursprünglichen Schlüsselverzeichnis besitzen beispielsweise die Schlüssel des Feldes mit dem Feldnamen KZ06 folgenden Aufbau, wobei die erste Zeile die Überschrift der darauf folgenden Schlüssel darstellt:

Anschläge gegenüber der Zivilbevölkerung der DDR

1301 Überfall
1302 Beschießen
[...]

Würden diese in der bereits beschriebenen Form umgesetzt, so erhielten die Schlüssel die folgende Form:

<!ENTITY KZ6_1301 " Überfall">
<!ENTITY KZ6_1302 " Beschießen">

Um die Informationen aus den Überschriften des Schlüsselverzeichnisses nicht zu verlieren, wurden diese in der Schlüssel-DTD zum Inhalt der Entität hinzugefügt. Ohne dieses Vorgehen sähe der Benutzer bei einer Abfrage nur die Wörter „Überfall“ und „Beschießen“, was einen erheblichen Informationsverlust gegenüber der ursprünglichen Aussage darstellen würde. Aus dem obigen Auszug der Codeliste wurden somit untenstehende Entitäten:

<!ENTITY KZ6_1301 " Anschläge der Zivilbevölkerung der DDR: Überfall">
<!ENTITY KZ6_1302 " Anschläge der Zivilbevölkerung der DDR: Beschießen">

---

[4] Björn Dehms, Langzeitarchivierung einfacher, relationaler Datenbanken. Entwicklung eines Prototypen zur Migration nach XML. Diplomarbeit. Universität Koblenz, Fachbereich Informatik, Koblenz 2000.
[5] Ebd.

Besondere Bedeutung kommt auch den fehlerhaften Datenbankeingaben (z. B. durch Zahlendreher) zu, wodurch Schlüssel (und später Entitäten) entstehen können, die nicht in der Schlüssel-DTD wieder zu finden sind und eine Fehlermeldung des XML-Parsers hervorrufen. Zu der entwickelten Fehlerbehandlung kann stark verkürzt gesagt werden, dass zu jedem fehlerhaft eingegebenen Schlüssel eine leere Entität definiert wird, z. B.

<!ENTITY Feld1_9999 " " >

Handelt es sich bei dem Schlüssel „9999“ um eine fehlerhafte Eingabe, so wird durch den XML-Parser an dieser Stelle der leere String ausgegeben, was gegenüber der Original-Datenbank einen weiteren Authentizitätsverlust darstellt. Das exakte Format zur Fehlerbehandlung ist ebenfalls bei Dehms[6] zu finden.

Zur Beschreibung der in der entwickelten Lösung erzielten Authentizität wurde der Begriff der „Bedienungsauthentizität“ geprägt: Im besten Fall sollte nach der Migration der relationalen Datenbank in das XML-Dateiformat ein Unterschied zur Abfrage und Manipulation relationaler Datenbanken mittels SQL nicht mehr erkennbar sein. Dem Benutzer soll im besten Fall verborgen bleiben, dass es sich bei der verwendeten Datenbank um keine relationale Datenbank, sondern vielmehr um eine Datenbank im XML-Dateiformat handelt. Diese Form der Authentizität war aus Gründen, die Dehms näher erläutert[7], nicht vollständig zu erreichen.

Die drei denkbaren Möglichkeiten zur Umsetzung der Schlüssel in Klartexte sollen nun vor allem bezüglich des Speicherbedarfs auf der Festplatte betrachtet werden: Die erste Möglichkeit wäre die Speicherung des Klartextes anstelle des Schlüssels im XML-Dokument. Hierbei würde der ursprüngliche Schlüssel nicht im Dokument abgelegt und ginge verloren. Der Platzbedarf dieser Variante wäre enorm hoch, da die u. U. sehr langen Klartexte an verschiedenen Stellen des XML-Dokumentes wiederholt abgelegt würden, wodurch sich die Menge redundanter Daten stark vergrößerte. Zweitens könnte ein Verweis auf den Klartext abgespeichert werden, was der von Dehms[8] schließlich favorisierten Lösung entspricht. Der Klartext würde nicht direkt im XML-Dokument, sondern erst zur Anzeige vom XML-Prozessor an der Stelle des Verweises eingefügt und auf dem Ausgabegerät ausgegeben. Es müssten nicht mehr die Klartexte an verschiedenen Stellen des XML-Dokumentes redundant gespeichert werden, sondern nur noch die Verweise, die jedoch erheblich kürzer sind als die zu ihnen gehörenden Klartexte. Gegenüber Lösung 1 reduzierte sich der Speicherbedarf hierbei um fast die Hälfte (s. Berechnung in Dehms[9]). Bei der dritten Variante würden nur die Schlüssel, d.h. die Daten in ihrer ursprünglichen Form, im XML-Dokument abgelegt. Die unter Umständen sehr aufwendige Interpretation der Schlüssel fände dann während der Aufbereitung der Daten zur Anzeige statt. Gegenüber Lösung 2 würde sich der Speicherbedarf um weitere 15,5 % reduzieren.

Lösung 1 kommt somit nicht mehr in Betracht. Lösung 3 ist jedoch nur unter dem Gesichtspunkt des Speicherbedarfs auf der Festplatte die optimale Lösung und benötigt gegenüber der Lösung 2 bei der Anzeige eine aufwändigere Bearbeitung.

## 3. Vorstellung des Migrationsprogramms mit einer Kurzdarstellung der 7 Schritte

Das von Dehms[10] entwickelte Migrationsprogramm überträgt die Informationen eines aus einer Datenbank ausgesonderten ASCII-Flat-Files in ein XML-Dokument. Da ASCII-Flat-Files in den unterschiedlichsten Formaten vorliegen können, muss der Benutzer die Struktur

[6] Ebd.
[7] Ebd.
[8] Ebd.
[9] Ebd.
[10] Ebd.

der jeweiligen Datei möglichst präzise beschreiben, wozu das Programm sieben aufeinander folgende Dialoge anbietet. Als Eingabe in den Migrationsprozess erhält das Migrationsprogramm drei Dateien:

die aus der Datenbank erzeugte Aussonderungsdatei, das ASCII-Flat-File,

die Schlüssel-DTD,

die Datei mit den Namen der Felder der Datenbanktabelle, falls diese nicht über die Tastatur eingegeben werden sollen.

Zu diesen drei Dateien müssen im Startdialog Angaben gemacht werden: Das oberste Editierfeld verlangt zuerst nach einem frei wählbaren Projektnamen. In den beiden darunter liegenden Editierfeldern müssen Name und Position des ASCII-Flat-Files sowie des zu erzeugenden XML-Dokumentes im MS-DOS-Format angegeben werden.

Das darunter liegende Panel betrifft Angaben zu den Namen der im XML-Dokument zu speichernden Felder der Datenbanktabelle. Diese können sowohl aus einer bereits erstellten Datei geladen werden oder über einen Dialog von Hand eingegeben werden. Über eine Checkbox kann weiterhin festgelegt werden, ob das ASCII-Flat-File Trennzeichen zwischen den einzelnen Feldern besitzt oder ob die Felder an deren Länge erkannt werden müssen.

Im untersten Editierfeld des Dialoges „Schritt 1 von 7“ müssen Ort und Name der Schlüssel-DTD angegeben werden, damit diese vom Migrationsprogramm zum XML-Dokument hinzugefügt werden kann. Das auf den Startdialog folgende Fenster listet die Feldbezeichner der Datenbank untereinander auf. Durch Markieren einer Checkbox wird das entsprechende Feld in das erzeugte XML-Dokument übertragen, anderenfalls erscheint es nicht in der Ausgabe. Der darauf folgende Dialog besitzt den gleichen Aufbau wie Schritt 2, hier müssen jedoch die verschlüsselten Datenbankfelder markiert werden.

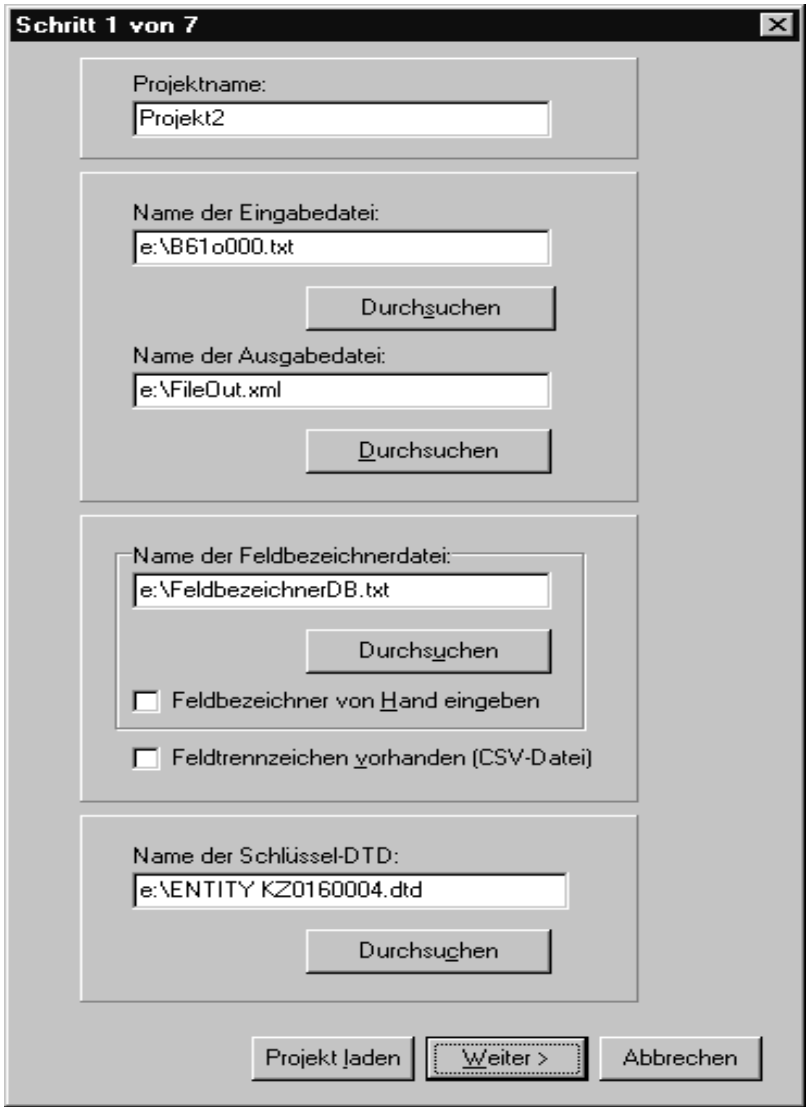


**Abbildung 3: Schritt 1 von 7**

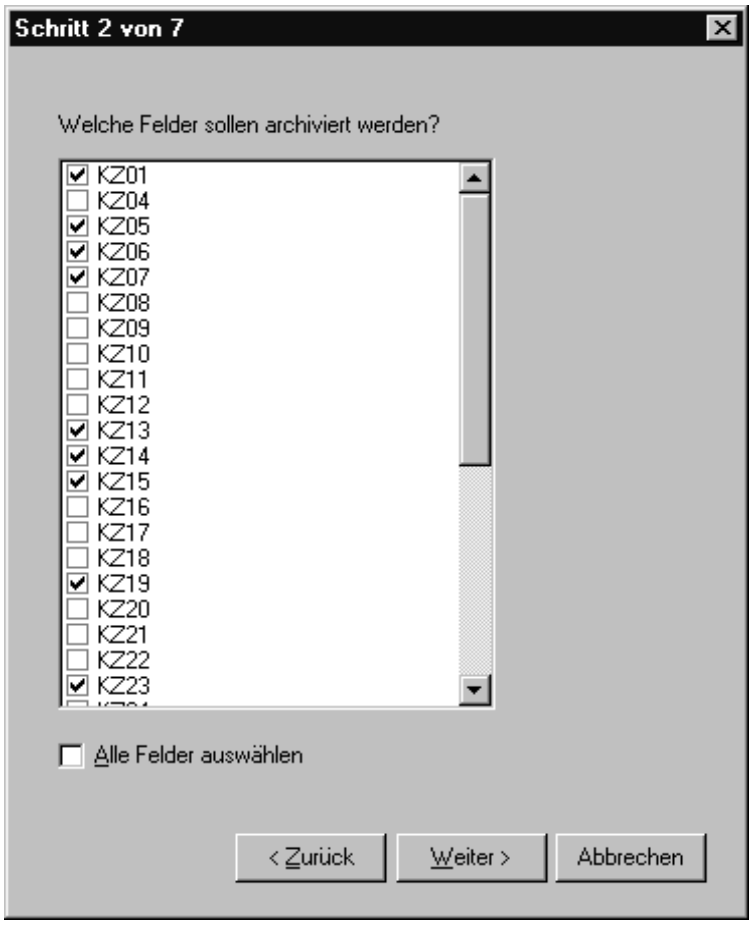


**Abbildung 4: Schritt 2 von 7**

Im Schritt 4 (vgl. Abbildung 5) werden Datenbankfelder beschrieben, die mehrere Schlüssel enthalten. In der Grundeinstellung stellt das einzige Fenster des Dialogs alle im Schritt 2 ausgewählten Datenbankfelder in einer Baumansicht dar, der die Namen der einzelnen Felder als Knoten besitzt. Einem Knoten können dann Informationen über die Länge der einzelnen Schlüssel innerhalb des Feldes hinzugefügt werden (s. Dehms[11]). Mit einem weiteren Dialog, der wiederum dem Dialog aus Abbildung 4 entspricht, werden die Datenbankfelder ausgewählt, die mehrdeutige Schlüssel enthalten. Unter mehrdeutigen Schlüsseln versteht man Zahlenfolgen, die an unterschiedlichen Stellen innerhalb eines Feldes mit verschiedenen Bedeutungen verbunden sind.

Der letzte Dialog (vgl. Abbildung 6) verlangt noch einmal nach Angaben über das Format des zugrundeliegenden ASCII-Flat-Files. Hier muss u. a. in einem eigenen Editierfeld das verwendete Feldtrennzeichen angegeben werden.

Eventuell sind in dem ASCII-Flat-File Begrenzerzeichen („Delimiter“) vorhanden, die die Feldinhalte einrahmen. Diese müssen im Panel „Begrenzer“ angegeben werden, damit sie bei der Ausgabe in das XML-Dokument ausgeschnitten werden können. Im Unterschied zu den Trennzeichen muss jedoch das Aussehen der Begrenzerzeichen in den Primärdaten angegeben werden, damit sie das Migrationsprogramm von den eigentlichen Begrenzerzeichen unterscheiden kann. Oft werden beispielsweise als Begrenzer Anführungszeichen verwendet, die auch als Zeichen im eigentlichen Feldinhalt auftauchen können.

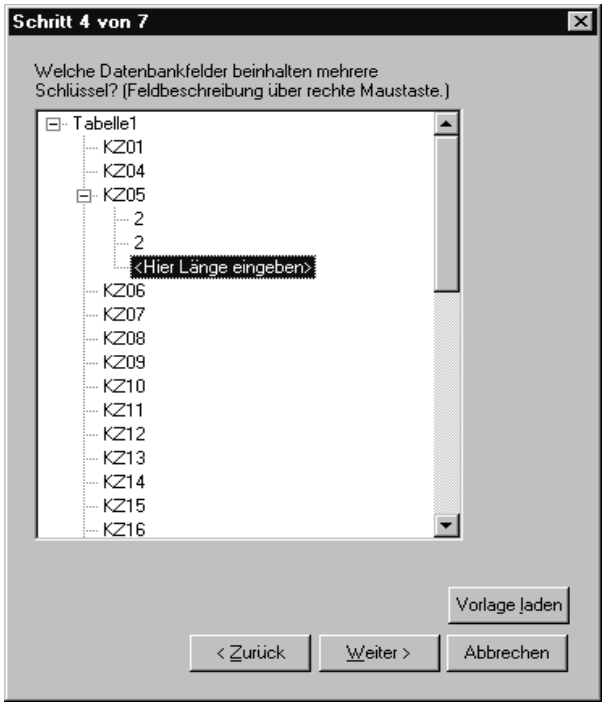


**Abbildung 5: Schritt 4 von 7**

[11] Björn Dehms (wie Anm. 4).

Die verschiedenen Zeilenumbrüche der unterschiedlichen Betriebssysteme können am linken Rand des Dialoges ausgewählt werden. Eine Projektübersicht listet zudem die am aktuellen Migrationsvorgang beteiligten Dateien auf. Der eigentliche Migrationsvorgang kann daraufhin durch Betätigen der „Migration beginnen/abbrechen"-Buttons gestartet bzw. vorzeitig beendet werden.

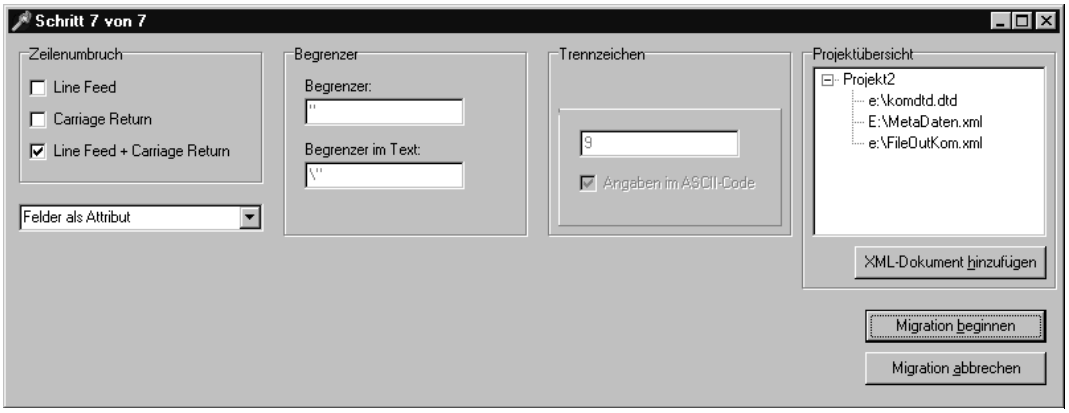


**Abbildung 6: Schritt 7 von 7**

## 4. Bereitstellung und Nutzung von Datenbank-Informationen in XML

Für die Bereitstellung und Nutzung von Datenbank-Informationen in XML sind die folgenden drei Szenarien vorstellbar. Die ersten beiden Nutzungsszenarien betreffen die Verwendung des XML-Dokuments als „Datenbank", das letzte Szenario beschreibt die Auswertung der enthaltenen Informationen über ein Statistiksystem:

*Auswahl eines oder mehrerer Datensätze ohne Einschränkung der Ausgabe*

Der Benutzer könnte sich einen oder mehrere von ihm ausgewählte Datensätze komplett anzeigen lassen, d. h. es würden die Inhalte aller Felder der ausgewählten Datensätze ausgegeben. In der relationalen Algebra entspricht dies der „Selektion" genannten Grundoperation.

*Auswahl eines oder mehrerer Datensätze mit Einschränkung der Ausgabe*

Diese Nutzungsmöglichkeit entspricht der unter Punkt 1 genannten, jedoch würden nur die vom Benutzer ausgewählten Felder der selektierten Datensätze ausgegeben. Die Einschränkung der Ausgabe auf bestimmte Tabellenfelder wird in der relationalen Algebra mit „Projektion" bezeichnet.

*Nutzung mit einem Statistikprogramm*

Da XML als Dateiformat u. a. ausgewählt wurde, um XML-Dokumente in andere Dateiformate migrieren zu können, ist eine Umwandlung des XML-Dokuments in das von einem Statistikprogramm benötigte Dateiformat möglich. Dieses könnte von einem Statistiksystem eingelesen werden und komplexe Auswertungen des Datenbestandes an einem Einzelplatzrechner ermöglichen. Diese Nutzungsmöglichkeit wäre somit nicht für die Allgemeinheit bestimmt.

Damit eine möglichst strenge Authentizität der Datenbank auch nach der Migration in ein XML-Dokument bewahrt bleibt, sollte sich die Nutzung des XML-Dokuments möglichst wenig von der Nutzung relationaler Datenbanken über ein Datenbankmanagement-System unterscheiden. Somit müssten auch bei der Verwendung von XML-Dokumenten als Daten-

basis die bei Datenbanken üblichen SQL[12]-Statements (DELETE, UPDATE, INSERT und SELECT) angeboten werden.

Da nach der eigentlichen Archivierung in aller Regel keine Änderungen an den in der archivierten Unterlage gespeicherten Informationen mehr vorgenommen werden, kann die Umsetzung der INSERT-, DELETE- und UPDATE-Statements entfallen. Es müssen lediglich SELECT-Statements angeboten werden, womit die Nutzung der Datenbank auf eine Abfrage der Informationen beschränkt ist.

Ein SELECT-Statement hat in SQL die folgende Syntax:

Select <FieldList> From <Tabelle> Where <Conditionlist>

Durch die „Conditionlist“ geschieht die Selektion der Datensätze einer Datenbanktabelle. Hier wird die Ergebnismenge (= die Menge der von einer Datenbankabfrage zurückgelieferten Datensätze) dadurch eingeschränkt, daß zu verschiedenen Feldbezeichnern Bedingungen angegeben werden, wie z. B.

Select <FieldList> From <Tabelle> Where Feld1>100 And Feld2=300

In diesem Beispiel werden somit nur solche Einträge aus der mit <Tabelle> spezifizierten Tabelle ausgewählt, bei denen „Feld1“ Werte größer 100 besitzt und der Wert von „Feld2“ genau 300 beträgt. In der Fieldlist werden die auszugebenden Felder der Datenbanktabelle angegeben. Eine mögliche Erweiterung der in obigem Beispiel angegebenen SELECT-Anweisung wäre somit

Select Feld3,Feld4 From <Tabelle> Where Feld1>100 Or Feld2=300

Hierdurch wird die Ergebnismenge auf die Felder „Feld3“ und „Feld4“ projiziert, d.h. in der Ausgabe erscheinen von jedem Datensatz nur die Werte dieser beiden Felder. Mit der Angabe „*“ für den Parameter „Fieldlist“ werden die Werte aller Felder ausgegeben. In der Fieldlist können auch Aggregationsfunktionen, z.B. Summe, Mittelwert oder Trefferanzahl der gefundenen Datenbankeinträge, verwendet werden.

**Erzeugung der Abfrage (auf dem Client)**

Um in SQL unerfahrenen Benutzern eine komfortable Auswertungsmöglichkeit des XML-Dokuments zu bieten, wird eine grafische Benutzeroberfläche in Form einer HTML-Seite im WWW bereitgestellt, mit der eine Suchanfrage gestellt werden kann. Das World Wide Web wurde als Medium gewählt, um die Daten für jedermann zugänglich zu machen. Eine Alternative wäre das Verschicken der Datenbank/des XML-Dokuments auf einem Datenträger, wobei allerdings die Nutzung der Unterlage durch den Empfänger nicht sichergestellt werden könnte, da diese u. U. spezielle Programme erfordert. Dennoch ist das Bestellen der Daten prinzipiell möglich.

Das im WWW bereitgestellte Formular beinhaltet einen mit der Skriptsprache „Javascript“ erzeugten Baum. Der Aufbau des Baumes ist sehr rechenintensiv und erfordert ein wenig Geduld. Der schließlich angezeigte Javascript-Baum ermöglicht eine Auswahl von Werten zu bestimmten Datenbankfeldern. Zu jedem Feld kann nur ein Wert ausgewählt werden, die selektierten Werte der verschiedenen Felder werden in der TextBox des HTML-Formulars angezeigt. Sie können in der TextBox nicht bearbeitet werden und werden im Hintergrund zu einem SELECT-Statement zusammengestellt, indem die ausgewählten Werte in der Fieldlist mit „And“ miteinander verknüpft werden. Wird aus dem JavaScript-Baum als Staats-

[12] SQL = Structured Query Language. Mit SQL können die Datensätze einer Datenbank abgefragt und manipuliert werden.

angehörigkeit beispielsweise „DDR“ und als Wochentag „Mittwoch“ gewählt, so besitzt das generierte SELECT-Statement den folgenden Aufbau:

SELECT * FROM Table WHERE KZ17 = 31 And KZ08 = 3


**Abbildung 7: Die Anzeige eines Datensatzes**

Als Werte der Felder erscheinen hier die Schlüssel und nicht die Klartexte des Schlüsselverzeichnisses.

Nach Betätigen des „Abschicken“-Buttons wird das im Hintergrund generierte SELECT-Statement zum Server gesandt. Die Verarbeitung des SELECT-Statements ist im folgenden Kapitel beschrieben, in Abbildung 7 ist beispielhaft das Ergebnis der Verarbeitung eines SELECT-Statements zu sehen. Das grafische Front-End bietet also die Ausgabe von einzelnen Datensätzen an; bei mehreren zur Ausgabe anstehenden Datensätzen kann zwischen diesen geblättert werden. Jeder Datensatz wird komplett angezeigt, d.h. es wird bei der Anzeige keine Einschränkung auf bestimmte Datenbankfelder ermöglicht. Dies entspricht der ersten der zu Beginn dieses Kapitels beschriebenen Nutzungsmöglichkeiten.

**Erzeugung der Ausgabe (auf dem Server)**

Eine der für die Ausgabe der Informationen zuständigen Komponenten ist der SQL-Parser, der in Form einer ActiveX-DLL implementiert ist. Die Form einer Dynamic Link Library (DLL) wurde gewählt, um den SQL-Parser als eigenständige Komponente auch in anderen Projekten verwenden zu können, denn die Arbeit des SQL-Parsers hängt nicht vom zugrundeliegenden XML-Dokument ab. Im Anschluß an die Beschreibung des SQL-Parsers befindet sich die Beschreibung der zur Erzeugung der Ausgabe verwendeten Active Server Pages (ASP).

*SQL-Parser*

Der SQL-Parser dient zur Analyse eines SQL-Statements. Er kann die oben beschriebenen SELECT-Statements verarbeiten und wurde unter Zuhilfenahme der frei erhältlichen Tools „LEX“ (zur lexikalischen Analyse) und „YACC“ (als Parsergenerator) erstellt. Diese beiden Tools erhalten als Eingabe eine Grammatik, die den SQL-Befehl beschreibt, und erzeugen als Ausgabe jeweils eine Delphi-Datei, die eine Klasse mit dem Code zur lexikalischen Analyse bzw. dem Code zum Parsen des SQL-Statements enthält. Normalerweise ist es die Hauptaufgabe eines Parsers, für ein eingegebenes Wort zu überprüfen, ob dieses aus der zugrundeliegenden Grammatik erzeugt werden kann. Es wurde jedoch vielmehr von der Möglichkeit zur Abarbeitung semantischer Aktionen Gebrauch gemacht, die bei der Erkennung bestimmter Merkmale des SQL-Statements ausgeführt werden. Zur Erzeugung der eigentlichen Ausgabe werden Active Server Pages (ASP) verwendet.

*Active Server Pages (ASP)*

Microsoft® Active Server Pages (ASP) sind eine serverseitige Skripttechnologie zur Erzeugung von dynamischen, interaktiven Webseiten. Eine ASP-Seite ist im Prinzip eine HTML-Seite, die zusätzlich zu den HTML-Konstrukten auf dem Server auszuführenden Code (= Skripte) enthält. Diese Skripte werden ausgeführt, wenn der Benutzer mit Hilfe eines Browsers auf eine solche HTML-Seite (deren Dateiendung allerdings „.asp“ ist) zugreift. Der Server arbeitet die angefragte Datei von oben nach unten ab und führt alle eingebetteten Skripte aus. Die Skripte können mit JScript oder Visual Basic Script (VBScript) verfasst sein und erzeugen meist eine HTML-Seite, die an den Client geschickt und dort vom Browser angezeigt wird.

Die für die Zwecke der Examensarbeit entwickelte Active Server Page ist speziell an das im Rahmen der Diplomarbeit migrierte XML-Dokument angepaßt und kann nicht für andere migrierte Datenbanken verwendet werden. Dies wird durch den besonderen Aufbau der zugrundeliegenden Datenbank verhindert, der keine für alle „XML-Datenbanken“ gültigen Auswertungsskripte zulässt.

Die ASP erzeugt eine HTML-Seite, die einen der vom Benutzer ausgewählten Datensätze anzeigt (vgl. Abbildung 7). XSL Style Sheets konnten zur Anzeige nicht verwendet werden, da sie noch nicht in einer „official recommendation“ festgeschrieben sind und deshalb z. Z. noch von keinem Browser unterstützt werden.

Zu Beginn der Skriptverarbeitung wird dem SQL-Parser das vom Benutzer eingegebene SELECT-Statement übergeben, der dieses auf korrekte Syntax überprüft. Dann wird das XML-Dokument in den Speicher geladen.

Es werden der Reihe nach alle im XML-Dokument enthaltenen Datensätze darauf getestet, ob sie die in dem SELECT-Statement angegebenen Bedingungen erfüllen. Der erste Datensatz, der bei dieser Überprüfung ein positives Ergebnis liefert, wird ausgegeben. Wird über den Client vorwärts geblättert, so wird wieder der Reihe nach der nächste passende Datensatz im

XML-Dokument gesucht. Beim Zurückblättern wird in der gleichen Weise rückwärts nach dem nächsten passenden Datensatz gesucht.

Die Verarbeitung von XML-Dokumenten unterliegt einigen Einschränkungen, die sich aus der Performanz des XML-Parsers ergeben.

**Berechnungen zur Performanz des XML-Parsers**

Sämtliche Angaben zur Performanz des XML-Parsers sind Lovett[13] entnommen. In diesem Artikel werden folgende Variablen angegeben, die die Performanz des XML-Parsers beeinflussen:

- die Art der XML-Daten
- das Verhältnis von Tags zu Text
- das Verhältnis von Attributen zu Elementen
- die Menge der Formatierungszeichen („The amount of discarded white space.“ Lovett.)

Leider ist in der Quelle nicht genauer angegeben, in welcher Weise diese Größen die Performanz des XML-Parsers beeinflussen.

Es werden an dieser Stelle nur Berechnungen bzgl. des Hauptspeicherbedarfs angeführt, da sich dieser im Verlaufe der Arbeit als der begrenzende Faktor herausstellte. Diese Berechnungen werden am Ende des Kapitels auf die „Datei Grenzzwischenfälle“ übertragen. Die zugrundeliegenden Abschätzungen sind sehr grob, geben aber einen guten Eindruck über den Ressourcenbedarf. Bei Lovett heißt es hierzu sinngemäß:

Der Speicherbedarf beim Laden eines XML-Dokumentes in den Hauptspeicher ist etwa bis zu vier mal größer als beim Ablegen des XML-Dokumentes auf der Festplatte. Der Hauptspeicherbedarf eines XML-Dokumentes kann mit der folgenden Formel grob abgeschätzt werden:

$$ws = 32(n + t) + 12t + 50u + 2w$$

Die folgende Tabelle gibt einen Überblick über die Variablen und deren Bedeutung:

| Variable | Beschreibung |
|---|---|
| ws | Größe des Hauptspeicherbedarfs in Bytes |
| n | Summe der Elemente und Attribute |
| t | Anzahl der Textknoten |
| u | Anzahl der verschiedenen Element- und Attributnamen |
| w | Anzahl der Unicode-Zeichen in den Textknoten (einschließlich der Attributinhalte). Wird der ASCII-Zeichensatz verwendet, so muß die Anzahl der Zeichen mit 2 multipliziert werden. |

Die Variablen besitzen bei dem nach Dehms[14] erzeugten XML-Dokument in Abhängigkeit von der Anzahl der abgespeicherten Datensätze die hier aufgelisteten Werte:

n = 37 x Anzahl der Datensätze
t = 73 x Anzahl der Datensätze
u = 12
w = (2 x 40) x 37 x Anzahl der Datensätze

---

[13] Chris Lovett, Inside MSXML Performance, Redmont, Microsoft Corporation, 21.02.2000, *http://msdn.microsoft.com/xml/articles/xml02212000.asp.*
[14] Björn Dehms (wie Anm. 4).

Für die Variable w wurde die mittlere Länge des Inhalts der Textknoten sehr grob mit 40 Bytes geschätzt. Da der ASCII-Zeichensatz zugrundeliegt, muß dieser Wert wie in der Tabelle beschrieben mit 2 multipliziert werden.

Werden die angegebenen Werte in obige Formel eingesetzt, so ergibt sich nach deren Umstellung für die Anzahl der möglichen Datensätze folgende Formel:

$$\#Datensätze = \frac{ws - 600}{10316}$$

Um die maximale Anzahl der zu verarbeitenden Datensätze zu erhalten, muss für die Variable „ws“ die Größe des virtuellen Speichers in Byte eingesetzt werden.

Die Gleichung wird nun auf die tatsächlich zur Verfügung stehende Hardware umgesetzt: Der auf dem verwendeten Rechner zur Verfügung stehende virtuelle Speicher besaß etwa 250 MB. Von diesen sind etwa 60 MB ständig durch das Betriebssystem und im Hintergrund laufende Programme belegt, so dass noch 190 MB ( = 199229440 Byte) virtueller Speicher für den XML-Parser zur Verfügung stehen. Dieser Wert wird in die obige Formel eingesetzt, so dass sich für die maximale Anzahl mit dem Rechner zu verarbeitender Datensätze folgender Wert ergibt:

$$\#Datensätze = \frac{199229440 - 600}{10316} \approx 19300$$

Dieser Wert stimmt ungefähr mit den Erfahrungswerten überein, nach denen bei XML-Dokumenten mit etwa 20.000 Datensätzen eine Fehlermeldung über die komplette Auslastung des virtuellen Speichers erscheint. In der Praxis werden jedoch nur etwa 10.000 Datensätze auf dem Server hinterlegt, da die Antwortzeiten aufgrund des häufigen Zugriffs auf den Swap-Speicher bei größeren XML-Dokumenten nicht mehr vertretbar sind. Nach Möglichkeit sollte das gesamte XML-Dokument in dem physisch vorhandenen Hauptspeicher Platz finden.

Unter diesen Bedingungen eignet sich das XML-Dateiformat nicht als Grundlage für die Präsentation der in der Datenbank vorhandenen Informationen im Internet, denn bei insgesamt 650.000 vorhandenen Datensätzen werden auch in absehbarer Zukunft keine Rechnersysteme zur Verfügung stehen, die eine derartige Datenmenge verarbeiten können.

## 5. Erfahrungen mit XML als Archivierungs- und Nutzungsformat

Einfache, nur aus einer Tabelle bestehende Datenbanken lassen sich mit XML beschreiben. Bei Datenbanken ohne Schlüsselverzeichnis ist eine Migration ohne Informationsverlust problemlos durchführbar. Erheblichen Arbeitsaufwand bereitet bei verschlüsselten Datenbanken hingegen die verlustfreie Umsetzung des Schlüsselverzeichnisses in die Schlüssel-DTD. Aufgabe des Archivars ist es, diesen Informationsverlust möglichst klein zu halten.

Mit XML archivierte Datenbanken gängiger Größenordnung können auf heutigen Rechnersystemen unter Verwendung eines Standardbrowsers nicht sinnvoll präsentiert verwendet werden, da der Hauptspeicherbedarf des XML-Parsers bei der Verarbeitung von XML-Dokumenten zu hoch ist. Zur Nutzung von XML-archivierten Datenbanken werden demzufolge spezielle Programme zur Aufbereitung der Präsentation notwendig, was den Vorteil von XML als softwareunabhängigem Archivierungsformat erheblich relativiert.

Zusammenfassend kann festgehalten werden, dass sich XML zwar zur Aufbewahrung und Sicherung von einfachen, relationalen Datenbanken anbietet, zur Präsentation der Informationen ohne Zusatzprogramme unter realistischen Einsatzbedingungen aber untauglich ist. Vor diesem Hintergrund ist die Verwendung des XML-Dateiformats zur Langzeitarchivierung von Datenbanken nur bedingt zu empfehlen. Vor allem bei verschlüsselten Datenbanken

sollte überlegt werden, ob sich die zeitaufwendige Umsetzung des Schlüsselverzeichnisses in eine Schlüssel-DTD lohnt oder ob eine Magazinierung der Originaldatenbankdatei als ASCII-Flat-file gegenüber einer Migration nach XML nicht erheblich wirtschaftlicher ist.

Als Vorteil einer XML-Migration kann allerdings angesehen werden, dass auch mit der heute vorhandenen Technologie eine Auswahl von Datensätzen im XML-Format durchaus direkt unter Zugriff auf ein XML-Dokument präsentiert werden kann. Diese Präsentationsmöglichkeit wurde im Rahmen des Projekts zum Zweck der Aufbereitung der Datenbank als Online-Abfragemöglichkeit für interessierte Benutzer realisiert. Als Server zur Abfrage der Datenbankinformationen wurde eine Windows NT Workstation und der Personal Web Server (PWS) verwendet. Die Antwort-HTML-Seite wurde von einer Active Server Page (ASP) erzeugt. Die Online-Abfrage ist unter *http://miles.uni-koblenz.de/Query/open.html* abrufbar.

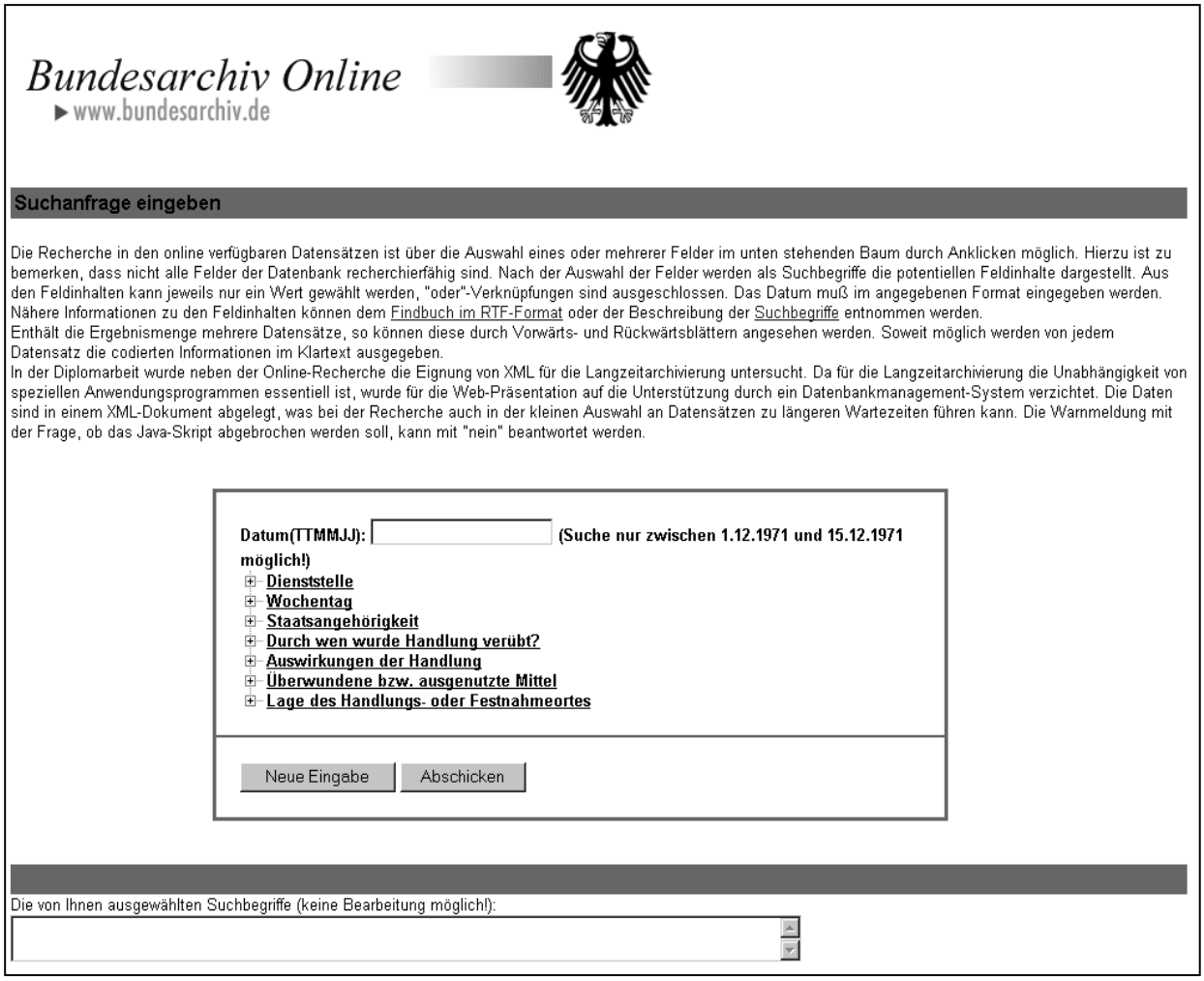


**Abbildung 8: Die Startseite der Online-Abfrage zur Datei der Grenzzwischenfälle**