Integration von Dateien in FDBS: Probleme und Lösungsvorschläge *

Michael Höding

Otto-von-Guericke-Universität Magdeburg
Institut für Technische Informationssysteme
Postfach 4120, D–39016 Magdeburg
E-mail: hoeding@iti.cs.uni-magdeburg.de
Tel.: ++49-391-67-12220 Fax: ++49-391-67-12020

1 Einleitung

Betrachtet man die EDV-Infrastruktur von Unternehmen und Organisationen, so wird man mit einer Reihe unterschiedlicher Informationssysteme konfrontiert. Diese Informationssysteme unterstützen unternehmenstypische Funktionen und Abläufe in Verwaltung, Produktionsvorbereitung und Produktion. Einige Systeme bieten einen sehr großen Funktionsumfang und integrieren so mehrere Abteilungen eines Unternehmens. Allerdings ist zu verzeichnen, daß die funktionalen Lücken, die derartige große Systeme im allgemeinen lassen, sehr schnell von stark spezialisierten und damit sehr leistungsfähigen Applikationen gefüllt werden. Da diese Spezialapplikationen jedoch oft unter funktionalen Gesichtspunkten entwickelt wurden und meist in verhältnismäßig preiswerten PC-Umgebungen laufen, kommen zur persistenten Datenhaltung oft Dateien zum Einsatz [Höd96b]. Dabei ist eine große Variantenvielfalt der genutzten Datenhaltungskonzepte zu beobachten. Sie reicht von der Abspeicherung der Daten in Form einer Hauptspeicherkopie, über spezielle applikationstypische Dateiformate bis hin zu Dateiformaten, die Quasi-Standards entsprechen und an etablierte Datenmodelle angelehnt sind.

Trotz dieser sehr heterogenen EDV-Infrastruktur, die sowohl Datenbanksysteme als auch dateibasierte Spezialsysteme umfaßt, ist eine Integration oft wünschenswert oder notwendig, da nur auf diese Weise eine globale Sicht auf alle Daten der verschiedenen heterogenen Informationssysteme realisiert oder ein global konsistenter Datenbestand gewährleistet werden kann[EL90].

Für derartige Integrationsprobleme werden seit einigen Jahren föderierte Datenbanksysteme (FDBS) diskutiert [HM85, SL90]. Diese Systeme integrieren ihre heterogenen Komponentensysteme bei weitgehender Wahrung der lokalen Autonomie und bieten eine homogene und transparente Sicht auf die verteilten Daten der Föderation. Neben der Integration von "echten" Datenbanksystemen ist auch die Integration von Dateienverbünden eine wichtige, zu untersuchende Problematik. *Dabei verstehen wir unter einem Dateienverbund eine Menge von Dateien, die semantisch als zusammengehörig betrachtet werden müssen.* Dies ist notwendig, da einerseits in vielen Fällen der Gesamtdatenbestand einer Applikation oder eines Applikationssystems auf mehrere Dateien verteilt ist und da andererseits nicht nur die Dateien selbst Daten enthalten, sondern auch Datei- und Pfadnamen Daten repräsentieren können. Das Hauptproblem für die Integration solcher Dateienverbünde in FDBS ist das Fehlen eines konzeptionellen Schemas. Wir werden uns deshalb in diesem Beitrag auf Überlegungen zur Ableitung eines Schemas, das die Struktur des Dateienverbundes adäquat beschreibt, konzentrieren.

Dazu werden wir zunächst kurz Grundlagen föderierter Datenbanksysteme in Zusammenhang mit Dateienverbünden betrachten. Danach stellen wir eine Klassifikation für Dateienverbünde vor und diskutieren die für die Schemaableitung zur Verfügung stehenden Ausgangsinformationen. Aufbauend darauf werden einige Schritte für die systematische Ableitung einer Strukturbeschreibung bzw. eines konzeptionellen Schemas für Dateienverbünde skizziert.

*Diese Arbeit wurde gefördert vom Land Sachsen-Anhalt unter FKZ: 1987A/0025 "Föderierung heterogener Datenbanksysteme und lokaler Datenhaltungskomponenten zur systemübergreifenden Integritätssicherung" – "**SIGMA**$_{FDB}$ (**S***chema* **I***ntegration and* **G***lobal integrity* **M***aintenance* **A***pproach* $_{for}$ **F**$_{ederated}$ **D**$_{ata}$ **B**$_{ases}$)"

2 Föderierte Datenbanksysteme

In diesem Abschnitt gehen wir auf einige wesentliche Grundlagen föderierter Datenbanksysteme ein. Dabei diskutieren wir kurz die Integration von Dateienverbünden.

Ein *föderiertes Datenbanksystem* (FDBS) besteht aus einer Reihe heterogener lokaler Datenbanksysteme und Datenhaltungssysteme, die mit einem *föderierten Datenbankmanagementsystem* (FDBMS) kooperieren [HM85, SL90, HTJ+95]. Das FDBMS bietet eine globale Schnittstelle, die Anwendungen auf globaler Ebene eine homogene Sicht auf die föderierten Daten offeriert. Den globalen Nutzern bleibt somit die Heterogenität der *Komponentensysteme* (CDB) verborgen. Da das FDBMS auf die lokalen Systeme über deren normale Zugriffsschnittstelle als normaler oder speziell autorisierter Nutzer zugreift, braucht das lokale System nicht verändert zu werden. Damit wird die lokale Autonomie der Komponentensysteme erhalten, wodurch die teilweise große Zahl spezifischer lokaler Applikationen weitgehend unbeeinflußt weiter genutzt werden kann. Zusätzliche Daten, z.B. Metadaten des FDBMS, aber auch echte, nur in der globalen Schicht bekannte Daten, werden in der globalen Datenbank des FDBMS verwaltet.

Ein wesentliches Ziel jeder Integration ist die Überwindung der Heterogenität verschiedenster Komponentensysteme. Wir setzen voraus, daß das Problem der technischen Heterogenität gelöst ist und hinreichende Mechanismen zur Kommunikation zur Verfügung stehen.

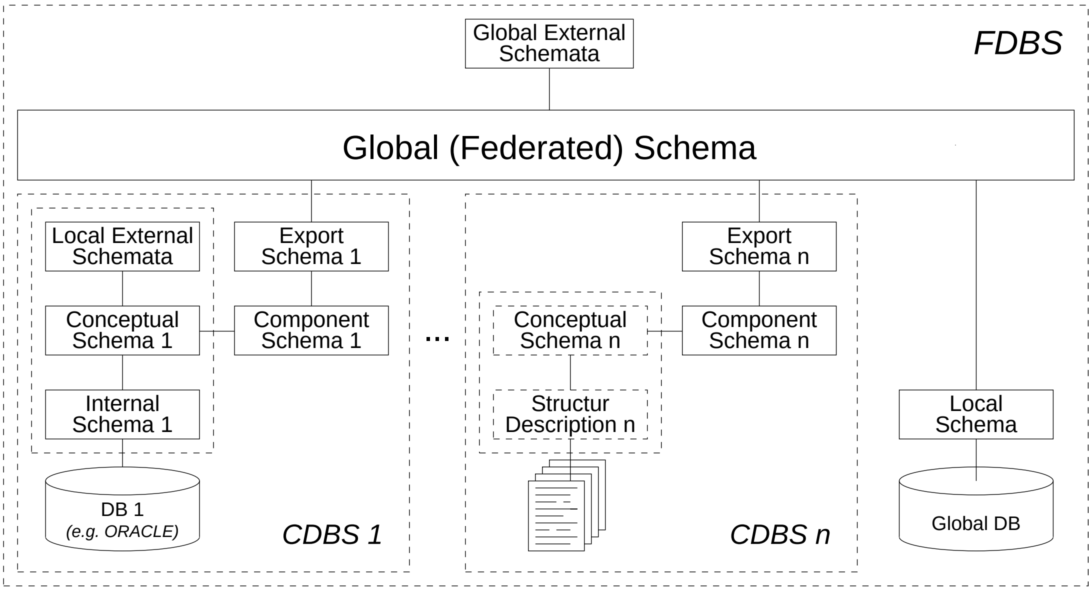

Abbildung 1: 5-Schemaebenen-Architektur nach [SL90]

Im weiteren unterscheiden wir Heterogenität auf Datenmodell-, auf Schema- und auf Datenebene. Diese Arten der Heterogenität spiegeln sich auch in der von [SL90] vorgestellten und allgemein anerkannten 5-Ebenen-Architektur wieder (vgl. Abbildung 1). Diese Referenzarchitektur erweitert die 3-Ebenen-Architektur nach ANSI/SPARC, bei der internes Schema, konzeptionelles Schema und mehrere externe Schemata unterschieden werden. In der 5-Ebenen-Architektur werden die lokalen konzeptionellen Schemata, die auf dem jeweiligen lokalen Datenmodell basieren, als erste Schemaebene betrachtet. Durch Transformation der *lokalen Schemata* in äquivalente Schemata auf Basis eines gemeinsamen Datenmodells wird die Datenmodellheterogenität überwunden. Die erzeugten Schemata werden als *Komponentenschemata* bezeichnet. Als nächstes werden durch Filterung die sogenannten *Exportschemata* erzeugt. Im nächsten Schritt erfolgt eine Integration aller Exportschemata zu einem *föderierten Schema*, das alle zu integrierenden lokalen Datenbanken adäquat abbildet. Dieses FDBS-Schema kann auch als Schema der globalen Schnittstelle dienen. Weiterhin ist auf Basis dieses Schemas die Ableitung weiterer *externer Schemata*, die nutzerspezifische Sichten realisieren, möglich. Dabei ist i.a. sowohl die Erzeugung von Datenmodell- als auch von Schemaheterogenität erwünscht.

Wir können zusammenfassend für den FDBS-Entwurfsprozeß zwei wesentliche Aufgaben unterscheiden. Dies sind die Schematransformation zur Überwindung der Datenmodellheterogenität und die Schemaintegration zur Überwindung der Schemaheterogenität. Zur Lösung dieser Aufgaben gibt es eine Reihe von Methodiken. Einige nutzen dabei als Integrationsdatenmodell semantisch reiche Datenmodelle, z.B. objektorientierte Datenmodelle [RBB+95, TS93]. Andere Integrationsmethoden basieren auf semantisch ärmeren Datenmodellen. Ein Beispiel hierfür ist das *Generic Integration Model* GIM [SS96a].

Entsprechend der hier vorgestellten Architektur für FDBS wird offensichtlich, daß das wesentliche Problem bei der Integration von Dateienverbünden das Fehlen eines konzeptionellen Schemas ist, das im Sinne der 5-Ebenen-Architektur als lokales Schema fungiert [Höd96a]. Das Schema ist im allgemeinen in einem oder mehreren Anwendungsprogrammen verborgen. Teilweise findet man auch in den Dateien selbst eine Vermischung von Daten und Metadaten vor. Es ist also notwendig, ein Schema abzuleiten bzw. zu modellieren, das die Struktur, d.h. alle durch die Applikationsprogramme möglichen Ausprägungen des Dateienverbundes, adäquat beschreibt. Da man bei der Wahl des Datenmodells, das zur Modellierung eingesetzt wird, weitgehende Freiheiten hat, ist es möglich, das gemeinsame Datenmodell des FDBS zu diesem Zweck zu nutzen. Dadurch wird eine erneute Datenmodelltransformation in das gemeinsame Datenmodell überflüssig. Das abgeleitete Schema kann sofort als Komponentenschema der Föderation dienen. Der weitere Entwurfsprozeß kann dann entsprechend einer der bekannten Methodiken zur Schemaintegration bzw. zum Entwurf föderierter Datenbanksysteme durchgeführt werden [RBB+95, TS93, SS96b].

Allerdings sind auch nach der Ableitung eines konzeptionellen Schemas einige Besonderheiten für Dateienverbünde in FDBS zu beachten. Zu bemerken ist, daß der Ableitungsprozeß meist sehr aufwendig ist und daß das abgeleitete Schema deshalb im allgemeinen nicht mit vollständiger Sicherheit alle verborgenen Metainformationen enthält. Ein weiteres Problem ist die geringe Granularität der Zugriffsoperationen von Spezialapplikationen auf Dateienverbünde. Oft werden ganze Dateien geladen, manipuliert und wieder gespeichert. Sich daraus ergebende Einschränkungen sind sowohl bei der Integration von Dateienverbünden in FDBS als auch bei der Bewertung von Ergebnissen von Anfragen auf der globalen Schnittstelle des FDBS zu beachten.

3 Dateienverbünde

In diesem Abschnitt skizzieren wir einen systematischen Weg zur Ableitung eines konzeptionellen Schemas für Dateienverbünde. Dieser Ansatz basiert auf der Annahme, daß sich die große Zahl verschiedener Dateiformate zu Mengen ähnlicher Formate, denen die gleichen Grundkonzepte zugrunde liegen, zusammenfassen läßt, die deshalb auch durch spezifische Methoden untersucht werden sollten. Wir stellen eine solche empirische Klassifikation vor. Weiterhin diskutieren wir die zur Verfügung stehen Ausgangsinformationen für die Schemaableitung.

Bei Untersuchungen an Systemen zur Fabrikplanung, die im Rahmen des Projektes $\mathbf{SIGMA}_{FDB}$ durchgeführt wurden [HTJ+95, Köl96], haben wir erkannt, daß die Integration der von diesen PC-basierten Systemen genutzten Dateienverbünde eine wesentliche Forderung der Anwender ist. Die im folgenden kurz vorgestellte Klassifikation ist Ergebnis dieser Untersuchungen. Unserer Erfahrung nach bietet diese Klassifikation eine gute Überdeckung der zu integrierenden Dateiformate. Wir unterscheiden die folgenden Gruppen von Dateienverbünden:

- *Satzorientierte Dateienverbünde*
  Viele Spezialapplikationen, die in den konventionellen Programmiersprachen, die Record-Typen unterstützen, geschrieben wurden, nutzen für zur Laufzeit in Feldern oder Listen gehaltene Daten satzorientierte Dateien zur Datenspeicherung. Bemerkenswert ist, daß diese Dateien im allgemeinen keine Metainformationen enthalten, wodurch die Ableitung eines Schemas erschwert wird. Teilweise nutzen diese Systeme random-access lese/schreib-Operationen. Diese hohe Granularität der Zugriffsoperationen macht derartige Dateienverbünde für die Integration in FDBS besonders interessant.

- *xBase-ähnliche Dateienverbünde*
  Das xBase-Dateiformat wird von vielen PC-Applikationen zur Datenspeicherung genutzt. Es ist hinreichend bekannt und wohldefiniert [Bor95]. Ein auf dem xBase-Dateiformat basierender Dateienverbund besteht aus einer Menge von Dateien, die Daten ähnlich einer Relation des relationalen Modells abspeichern. Außerdem enthält jede Datei als Kopf Metainformationen, die die Struktur der Relation beschreiben. Teilweise sind auch Indexdateien oder spezielle Dateien, die einen Katalog darstellen, vorhanden. Die Bekanntheit des Formats und die Verfügbarkeit von Metainformationen ermöglicht eine weitgehende Ableitung eines Schemas für xBase-ähnliche Dateienverbünde.

- *Vollständig strukturierter Text mit Metainformationen*
  Diese Klasse von Dateienverbünden wird oft für (durch den Nutzer lesbare) Datenaustauschformate

genutzt[1]. Dabei sind Hierarchien von Metadaten, denen Daten zugeordnet sind, in einer Datei zu finden. Auf den ersten Blick erscheint die Ableitung eines Schemas recht einfach, da die Metadaten die Daten in einer oft sehr intuitiven Weise assoziieren. Komplizierter wird es allerdings, wenn keine vollständige Beschreibung der genutzten Metadaten vorliegt oder eine große Menge von Beispieldateien für die Ableitung des Schemas untersucht werden muß. Hierfür ist die Unterstützung durch spezifische Analysewerkzeuge notwendig.

- *Teilweise strukturierter Text mit Metainformationen*
  Auch die Integration von Textdateien, die strukturierte Komponenten enthalten, ist ein sinnvolles Unterfangen. Allerdings kann nur auf die strukturierten Komponenten zugegriffen werden. So kann z.B. das Wort `Telefonnummer` in einer E-Mail assoziieren, daß die nachfolgende Nummer die Telefonnummer des durch das Wort `From` und den nachfolgenden Namen bestimmten Absenders ist. Diese Daten könnten nach einer Integration eines solchen Mailverwaltungssystems in eine Kundendatenbank übernommen werden. Allerdings sind derartige Informationen recht unsicher, weshalb vor allem schreibende Zugriffe auf diese Dateienverbünde mit beträchtlichen Risiken verbunden sind.

- *Hypertext*
  Neben den auf WWW-Sites zur Verfügung stehenden Dateienverbünden aus HTML-Dateien, werden Hypertexte auch in vielen anderen Bereichen zur strukturierten Abspeicherung von Informationen, z.B. von hierarchisch strukturierten Produktkatalogen oder von Hilfesystemen, genutzt. Dazu dient vor allem das Konstrukt des Verweises auf andere Dateien des Dateienverbundes.

Zu bemerken ist, daß es Dateienverbünde gibt, die zu mehreren der aufgeführten Klassen gehören. So kann eine HTML-Datei, die ohne Zweifel zu einem als Hypertext zu klassifizierenden Dateienverbund gehört, auch teilweise formalisierbare Strukturinformationen, z.B. in Tabellenform, enthalten, so daß man sie auch als teilweise strukturierten Text charakterisieren kann.

Nachdem wir eine erste empirsche Klassifikation für Dateienverbünde vorgestellt haben, möchten wir im folgenden die für die Schemaableitung zur Verfügung stehenden Ausgangsinformationen diskutieren. Wir betrachten die folgenden Arten von Ausgangsinformationen:

- *Hintergrundwissen über das Anwendungsszenario und über das lokale Informationssystem*
  Dieses informale Wissen kann durch die Analyse von Dokumentationen, Beispielsitzungen und durch Interviews mit Anwendern gewonnen werden. Für eine automatische Weiterverarbeitung ist die Formalisierung dieses Wissens eine unabdingbare Forderung.

- *Beispieldateien*
  Diese wesentliche Informationsquelle repräsentiert einen oder mehrere mögliche Zustände des Dateienverbundes. Durch die Analyse der Beispieldateien können wesentliche notwendige Bestandteile des Schemas erkannt werden. Zu beachten ist jedoch, daß es anhand von Beispieldateien im allgemeinen nicht möglich ist, das Schema vollständig abzuleiten, da nicht immer alle durch das Applikationssystem möglichen Konstrukte auch wirklich im aktuellen Zustand des Dateienverbundes genutzt wurden.

- *Informationen über Zugriffsoperationen*
  Diese Informationen können Auskunft über die Struktur und die Granularität der gespeicherten Daten geben. Derartige Informationen kann man durch spezielle Dateiserver gewinnen, die die lokalen Dateizugriffe global überwachen[2]. Außerdem ist es denkbar, diesen Mechanismus zur Kommunikation mit dem FDBMS und zur Konsistenzsicherung im Sinne der in [TC96] diskutierten aktiven Mechanismen zu nutzen.

- *Quelltexte von Anwendungsprogrammen*
  Durch Quelltexte von Anwendungsprogrammen ist die vollständige Ableitung des Schemas möglich. Allerdings sind derartige Analyseprobleme im allgemeinen nicht entscheidbar. Deshalb müssen derartige Quelltexte durch den erfahrenen Programmierer hinsichtlich des abzuleitenden Schemas analysiert werden.

[1] Ein bekannter Vertreter ist das *DXF-Format* des Zeichenprogramms *Autocad*
[2] Wir haben dazu einige Untersuchungen an einem modifizierten SAMBA-Dateiserver durchgeführt.

4 Ein Vorschlag für die Schemaableitung

Aufbauend auf den in den vorherigen Abschnitten vorgestellten Überlegungen möchten wir jetzt kurz die Schritte unseres Vorschlages zur Schemaableitung diskutieren. Zuerst stellen wir die vorgeschlagenen Schritte überblicksartig vor. Danach beschreiben wir die einzelnen Schritte genauer. Zu bemerken ist, daß der Schemaableitungsprozeß im wesentlichen immer vom entwerfenden Menschen dominiert sein wird, da es für eine Reihe von Problemen keine eindeutige Lösung gibt. Jedoch sollten zu entwickelnde Methoden und Werkzeuge den Entwerfer mit verdichteten Informationen und Lösungsalternativen unterstützen, um so den Ableitungsprozeß weniger aufwendig zu gestalten und die Qualität der abgeleiteten Schemata zu erhöhen.

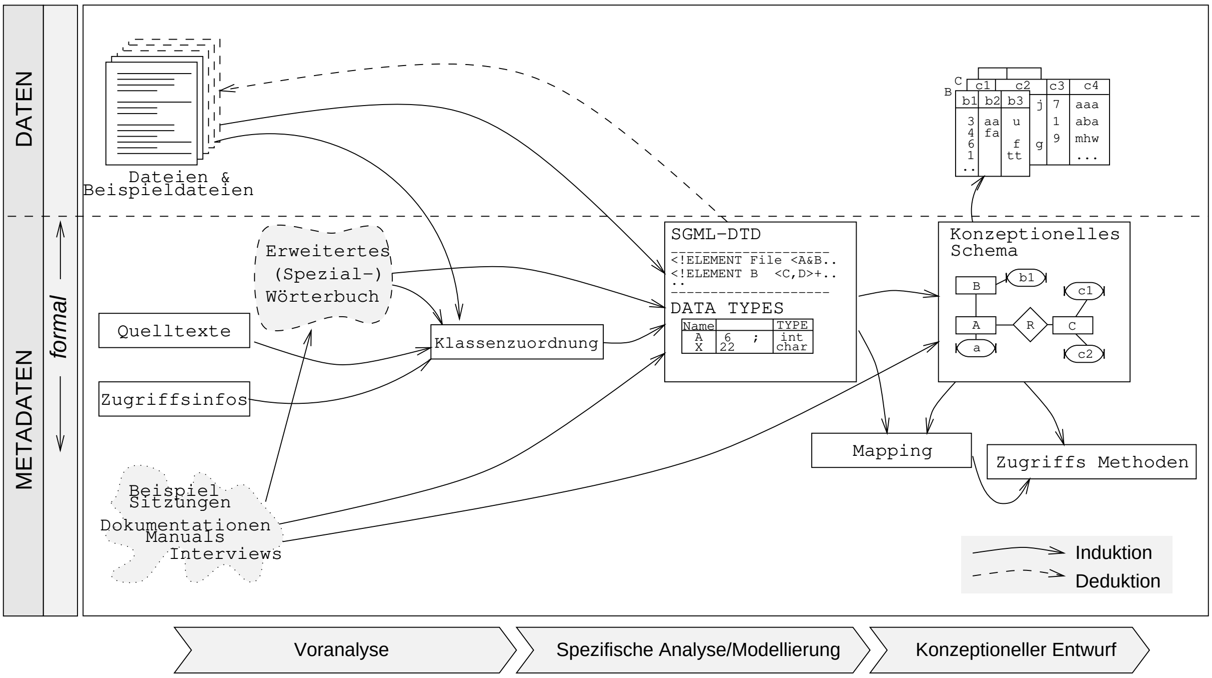

Abbildung 2: Schritte zur Schemaableitung

Das Vorgehen zur Ableitung eines konzeptionellen Schemas für einen Dateienverbund ist in Abbildung 2 skizziert. Der erste Schritt besteht in der Anreicherung eines allgemeinen Wörterbuches um spezifische Beispieldaten und vermutete Metadaten, die bei der Analyse des Hintergrundwissens gewonnen wurden. Unter Zuhilfenahme dieses erweiterten Wörterbuches wird auf Basis einer Analyse der Beispieldateien und Zugriffsinformationen eine Zuordnung des Dateienverbundes zu einer Klasse vorgenommen. Danach kommen klassenspezifische Analysemethoden und -werkzeuge zum Einsatz, die den Designer beim Entwurf einer Strukturbeschreibung und, im weiteren, eines konzeptionellen Schemas unterstützen.

4.1 Voranalyse

Der erste Schritt der Voranalyse ist die Formalisierung des informalen Hintergrundwissens. Wir schlagen dazu die Nutzung eines Wörterbuches vor (vgl. Abb. 3). Dieses Wörterbuch enthält Daten und Metadaten, die Metadaten referenzieren. Außerdem ist ein allgemeiner und ein spezifischer Teil zu unterscheiden. Im allgemeinen Teil sind Muster für Daten enthalten, wie z.B. verschiedene Muster für `date`. Weiterhin sind Daten enthalten, die evtl. Metadaten assoziieren, z.B. der Zahlenwert `1996`, der möglicherweise auf ein Attribut namens `year` verweist. Die Formalisierung des Hintergrundwissens wird durch die Erweiterung des Wörterbuches um anwendungsspezifische Daten und Metadaten und um Referenzen vorgenommen. Wenn sich z.B. aus der Analyse des Hintergrundwissens die Vermutung ergibt, daß ein Lieferdatum in dem Dateienverbund abgespeichert ist, so wird eine Referenz vom allgemeinen Metadatum `date` zu einem neuen Metadatum `delivery_date` aufgebaut. Weiterhin wird das Wörterbuch mit für das Anwendungssystem spezifischen Beispieldaten, z.B. typischen Produktnamen, angereichert. Wir können also zusammenfassend die folgenden Arten von Informationen im erweiterten Wörterbuch unterscheiden:

- verschiedene Muster, z.B. für Datumswerte (allgemeine Informationen)
- mögliche spezifische Objekt- und Attributnamen (zusätzliche Metainformationen)
- deren Synonyme (allgemeine Metainformationen)
- spezifische Beispieldaten (zusätzliche Informationen)

Das erweiterte Wörterbuch kann helfen, Hinweise zur Modellierung eines Schemas zu geben. Allerdings ist das hier skizzierte Wörterbuch nur ein erster, noch unvollständiger Vorschlag, da z.B. Probleme wie Homonyme und Synonyme in Verbindung mit einer großen Wissensbasis schnell zu unübersichtlichen Strukturen führen.

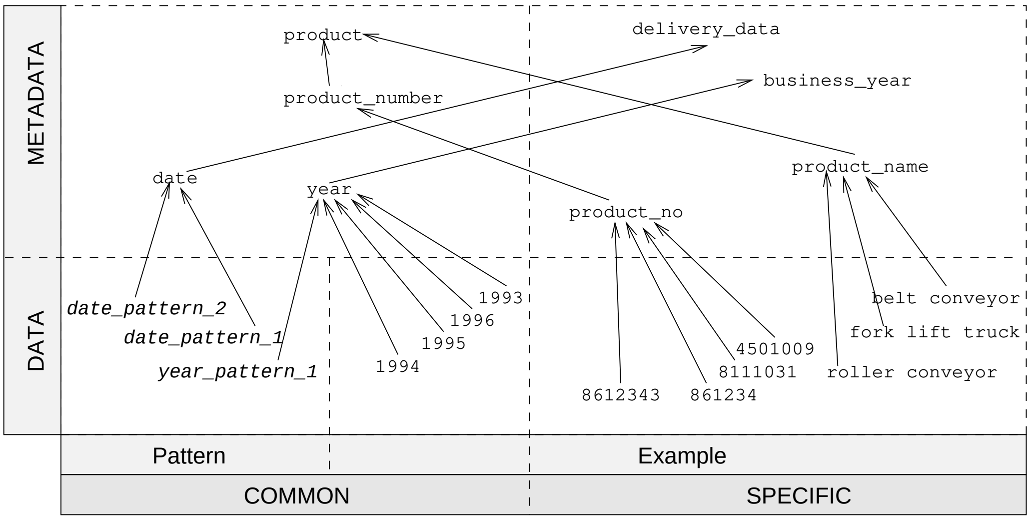

Abbildung 3: Daten und Metadaten im erweiterten Wörterbuch

In nächsten Schritt der Voranalyse wird der untersuchte Dateienverbund einer Klasse oder mehreren Klassen einer vorhandenen Klassifikation zugeordnet. Dabei werden die Beispieldateien unter Anwendung des erweiterten Wörterbuches und der Zugriffsinformationen analysiert, womit folgende Fragen beantwortet werden können:

- Sind in den Beispieldateien vermutete Daten enthalten ?
- Sind in den Beispieldateien vermutete Metadaten enthalten ?
- Also: Enthält der Dateienverbund nur Daten oder auch Metadaten ?
- Gibt es eine feste Satzgröße ?
- Mit welcher Granularität erfolgen lese/schreib-Zugriffe ?

Auf Grund der erhaltenen Antworten kann eine Klassenzuordnung erfolgen. Dafür müssen im Rahmen zukünftiger Arbeiten quantifizierbare Merkmale bestimmt und für jede Klasse entsprechende Gültigkeitsbereiche definiert werden.

4.2 Spezifische Analyse

Nach der Zuordnung des Dateienverbundes zu einer Klasse erfolgt die Analyse der Beispieldateien durch klassenspezifische Analysemethoden. Dabei ist eine Werkzeugunterstützung vor allem bei einer sehr großen Anzahl von Beispieldateien oder sehr komplexen Dateienverbünden notwendig. Als Ergebnis dieses Schritts soll ein Schema erzeugt werden, das die physische Struktur des Dateienverbundes beschreibt.

Für die Formulierung dieser Strukturbeschreibung schlagen wir die Nutzung der Sprache SGML vor. SGML zeichnet sich gegenüber der Nutzung einer Grammatik dadurch aus, daß es nicht nur formal ist, sondern auch eine anschauliche und verständliche Darstellung ermöglicht. Die Darstellung eines Schemas in SGML erfolgt in einer sogenannte *Document Type Definition* (SGML-DTD). Eine sehr populäre DTD ist z.B. das HTML-Format, in der die möglichen Strukturen für HTML-Dateien definiert sind. Ein Nachteil von SGML hinsichtlich der Nutzung als Strukturbeschreibungssprache für die Integration von Dateienverbünden in FDBS ist das Fehlen eines Typkonzeptes. Wir erweitern deshalb die Strukturbeschreibung für einen Dateienverbund um eine Tabelle, in der zu den terminalen Elementen der SGML-DTD Datentypen definiert werden. Um in späteren Phasen der Schemaintegration eine erneute Datentyptransformation zu vermeiden, sollten für die Strukturbeschreibung die Datentypen des gemeinsamen Datenmodells des FDBS genutzt werden.

Die spezifische Analyse und Ableitung der Strukturbeschreibung kann für die unterschiedlichen Klassen von Dateienverbünden verschieden gut durch Werkzeuge unterstützt werden. Bei sehr gut dokumentierten Dateiformaten, bei denen neben den Strukturierungsprinzipien auch die Semantik verfügbarer Konstrukte hinreichend erklärt ist, z.B. bei xBase-ähnlichen Dateiformaten, kann man den Schemaableitungsprozeß mitunter vollständig automatisieren.

Bei Dateien, die Metainformationen und Informationen vermischen, kann ein Tool anhand der Beispieldateien zuerst Kandidaten für Trennzeichen ermitteln. Aus dieser Menge kann der Entwerfer Trennzeichen auswählen, mit deren Hilfe bei einer erneuten Analyse der Beispieldateien, auch unter Anwendung des Wörterbuches, häufige Worte, die möglicherweise Schlüsselworte (also Metadaten) sind, zu finden. Aus dieser Menge von Schlüsselwortkandidaten müssen manuell die Schlüsselworte ausgewählt werden. Anhand der bestimmten Schlüsselworte und der Reihenfolge ihres Auftretens in den Beispieldateien kann ein Vorschlag für die Strukturbeschreibung generiert werden.

Die Analyse von satzorientierten Dateien durch den Menschen wird dadurch verkompliziert, daß im allgemeinen die Grenzen (oder die Länge) eines Datensatzes nicht klar erkennbar sind und dadurch, daß zur Abspeicherung numerischer Werte binäre Formate zum Einsatz kommen. Ein Analysetool muß also zum einen die Datensatz- und darauf basierend die Attributgrenzen erkennen und zum anderen die Umwandlung binär abgespeicherter numerischer Daten in eine anschauliche Form vornehmen. Zu beachten ist, daß die Art der Binärdarstellung von der für die Implementierung der Applikation genutzten Programmiersprache bzw. des Compilers oder der Systemarchitektur abhängt. Durch ein Tool sollten die am häufigsten genutzten Formate unterstützt werden. Da die Dateien nur Daten und keine Metadaten enthalten, muß mit Hilfe der im Wörterbuch enthaltenen Daten und ihrem Auftreten an einer bestimmten Dateiposition auf die Metadaten geschlossen werden. Mit anderen Worten, wenn an jeder 42sten Stelle einer Datei eine 4-Byte große Zahl steht, die als 32-Bit-Integer dekodiert in 30 Prozent der Fälle einer der im erweiterten Wörterbuch gespeicherten Produktnummern entspricht, so können wir vermuten, daß die Satzgröße 42 ist und daß an dieser Position eines Satzes immer eine Produktnummer gespeichert wird.

Wir können also zusammenfassen, daß die teilweise automatisch unterstützte Ableitung einer Strukturbeschreibung ein für unterschiedliche Klassen sehr spezifischer Prozeß ist. Eine Reihe von Werkzeugen ist zu entwerfen und ihr Nutzen für die Dateimodellierung zu überprüfen.

4.3 Konzeptioneller Verbesserungsentwurf

Mit der Herleitung einer Strukturbeschreibung für einen Dateienverbund ist die wesentliche Voraussetzung für die Integration des Dateienverbundes in ein FDBS geschaffen. Wir schlagen trotzdem einen weiteren Schritt zur Verbesserung der Strukturbeschreibung in Form eines konzeptionellen Schemas vor. Dieses konzeptionelle Schema sollte auf dem gemeinsamen Datenmodell des FDBS basieren.

Wir können davon ausgehen, daß in vielen Fällen für zu integrierende Dateienverbünde kein Datenbankentwurf durchgeführt wurde. Deshalb ist mit einer Reihe von Modellierungsfehlern zu rechnen, die aber teilweise durch die Analyse der vorhandenen Beispieldateien erkannt werden können. Ein Beispiel dafür ist z.B. die Verletzung der ersten Normalform.

Ein anderer Grund für einen Verbesserungsschritt ist die Möglichkeit, bei Nutzung eines objektorientierten gemeinsamen Datenmodells das konzeptionelle Schema semantisch anzureichern und so ein Schema zu erhalten, das über die Beschreibung der Struktur hinausgeht und in Anwendungen versteckte oder dem Anwender bekannte Semantik enthält.

5 Zusammenfassung und Ausblick

Im vorliegenden Beitrag haben wir die Probleme der Integration von Dateienverbünden in FDBS und einige Lösungsansätze diskutiert. Zusammenfassend sei bemerkt, daß die vorgestellten Ansätze und insbesondere die vorgeschlagene Klassifikation einer weiteren Untersuchung bedürfen. Die entworfenen Methoden und Algorithmen werden teilweise im Rahmen der FDBS-Entwurfsumgebung $\mathbf{SIGMA}_{BENCH}$ prototypisch realisiert und validiert werden.

Literatur

[Bor95] G. Born. *Fileformat Handbook.* International Thomson Publishing, London, 1995.

[EL90] J. Encarnação und P. C. Lockemann. *Engineering Databases - Connecting Islands of Automation Trough Databases.* Springer-Verlag, 1990.

[HM85] D. Heimbigner und D. McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3):253–278, Juli 1985.

[Höd96a] M. Höding. An Approach to the Integration of File Based Systems into Database Federations. In *Proc. of the Int. Workshop on Heterogeneous Information Management*, Prague, Czech Republic, November 1996. *To appear.*

[Höd96b] M. Höding. Föderierung von Dateien: Finden, Verwalten und Nutzen von Metainformationen. In S. Conrad, G. Saake, I. Schmitt und C. Türker, Herausgeber, *Kurzfassungen – 8. Workshop "Grundlagen von Datenbanken", Friedrichsbrunn/Harz (28.05.–31.05.96)*, S. 46–50. Institut für Technische Informationssysteme, Universität Magdeburg, Mai 1996.

[HTJ+95] M. Höding, C. Türker, S. Janssen, K.-U. Sattler, S. Conrad, G. Saake und I. Schmitt. Föderierte Datenbanksysteme — Grundlagen und Ziele des Projektes $\mathbf{SIGMA}_{FDB}$. Preprint Nr. 12, Fakultät für Informatik, Universität Magdeburg, Dezember 1995.

[Köl96] A. Köller. Dateien in föderierten Datenbanksystemen - Untersuchungen an Systemen zur Fabrikplanung. Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, September 1996.

[RBB+95] E. Radeke, R. Böttger, B. Burkert, Y. Engel, G. Kachel, S. Kolmschlag und D. Nolte. Efendi: Federated Database System of Cadlab. In M. J. Carey und D. A. Schneider, Herausgeber, *Proc. of the 1995 ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA, SIGMOD RECORD 24(2)*, S. 481. ACM Press, Juni 1995.

[SL90] A. P. Sheth und J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SS96a] I. Schmitt und G. Saake. Integration of Inheritance Trees as Part of View Generation for Database Federations. In *Proc. of the 15th Int. Conf. on Conceptual Modelling (ER'96).* LNCS, Springer-Verlag, Berlin, Oktober 1996. *To appear.*

[SS96b] I. Schmitt und G. Saake. Schema Integration and View Generation by Resolving Intensional and Extensional Overlappings. In *Proc. of the 9th Int. Conf. on Parallel and Distributed Computing Systems (PDCS'96)*, September 1996.

[TC96] C. Türker und S. Conrad. Using Active Mechanisms for Global Integrity Maintenance in Federated Database Systems. In S. Conrad, M. Höding, S. Janssen und G. Saake, Herausgeber, *Kurzfassungen des Workshops "Föderierte Datenbanken", Magdeburg, 22.04–23.04.96*, S. 51–65. Bericht 96–01, Institut für Technische Informationssysteme, Universität Magdeburg, April 1996.

[TS93] M. Tresch und M. Scholl. Schema Transformation Processors for Federated Objectbases. In S. C. Moon und H. Ikeda, Herausgeber, *Proc. of the 3rd Int. Conf. on Database Systems for Advanced Applications (DASFAA'93), Daejon, Korea*, S. 37–46. World Scientific Press, April 1993.