Aspektorientierte Datenhaltung – ein Modellierungsparadigma

Matthias Liebisch
Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena
m.liebisch@uni-jena.de

ZUSAMMENFASSUNG

Relationalen Datenbanksystemen kommt seit Jahrzehnten eine zentrale Rolle bezüglich der Speicherung von Anwendungsdaten zu. Im Fokus der nicht nur initial zu bewältigenden Aufgabe einer Datenbankmodellierung steht vor allem die strukturierte und modularisierte Abbildung von fachlichen Anforderungen aus den zugehörigen Anwendungen. In vielen Fällen stören jedoch sogenannte **funktionale Aspekte**, wie beispielsweise die Internationalisierung oder Versionierung von Nutzdaten, diesen Prozess, da sie als querschnittliche Belange weite Teile des Datenmodells betreffen und sich mit herkömmlichen Methoden nicht als eine logische Einheit modellieren lassen. Durch den damit verbundenen Verlust der Modularisierung entstehen weitere Nachteile wie eine schlechtere Wiederverwendbarkeit, Nachvollziehbarkeit und Wartung.

Diese Arbeit präsentiert das Paradigma der **Aspektorientierten Datenhaltung** als Lösungsansatz für die beschriebene Problematik. In Analogie zum Konzept der Aspektorientierten Programmiersprachen (AOP) zur Bewältigung der *Cross-Cutting Concerns* wird dabei das Ziel einer weitestgehenden Modularisierung sowohl der fachlichen als auch funktionalen Aspekte während der Datenbankmodellierung verfolgt. Hierfür werden Anforderungen und Eigenschaften formuliert, die später als Spezifikationsgrundlage für mögliche Realisierungsstrategien dienen. Abschließend beschreibt die Arbeit einen Implementierungsansatz am geeigneten Beispiel sowie dessen Bewertung.

Kategorien und Themenbeschreibung

H.2.8 [**Database Management**]: Database Applications; H.2.1 [**Database Management**]: Logical Design—*Schema and subschema, Data models*

Allgemeine Bestimmungen

Design

Copyright is held by the author/owner(s).
GvDB Workshop'10, 25.-28.05.2010, Bad Helmstedt, Germany.

1. EINLEITUNG UND MOTIVATION

Relationale Datenbankmanagementsysteme (RDBMS) spielen eine zentrale Rolle für die Persistierung beliebiger Daten von Anwendungssystemen. Die beim Entwurf von Software zugrunde gelegten Voraussetzungen und Annahmen können sich im Laufe der Zeit bedingt durch äußere Einflüsse verändern, beispielsweise durch neue wirtschaftliche, juristische oder funktionale Anforderungen[4]. Müssen Erweiterungen in einer Anwendung vorgenommen werden, dann wirken sich diese Anpassungen häufig auch auf das zugehörige Datenbankmodell aus (Schema-Evolution).

Die klassische relationale Daten(bank)modellierung folgt einem Phasenmodell, welches als wichtigste Schritte die Anforderungsanalyse, den konzeptuellen Entwurf über ein ER-Modell, den logischen Entwurf im relationalen Modell sowie abschließend den physischen Entwurf in einem konkreten Datenbanksystem beinhaltet[8]. Diese einzelnen Phasen unterliegen dabei nicht nur den fachlichen und inhaltlichen Vorgaben und Zielsetzungen, sondern sind auch mehr oder weniger stark geprägt von zusätzlichen, sogenannten funktionalen Aspekten, welche wie folgt definiert sind.

►*Definition 1.* Ein **funktionaler Aspekt** bezeichnet im Kontext dieser Arbeit eine spezifische Eigenschaft eines Datenmodells bzw. Gesamtsystems, welche sich nicht einem konkreten Objekt (*business object*) zuordnen lässt, sondern eine Vielzahl der modellierten Objekte betrifft.

Modul				
TeilNr	Bezeichnung	Preis	Währung	Flags
...	...	...	...	...

Modulstruktur		
Oberteil	Unterteil	Menge
...	...	...

Oberteil sowie Unterteil sind Fremdschlüssel bezogen auf Modul.TeilNr

Abbildung 1: Modellierung von Stücklisten

Vertreter solcher funktionalen Aspekte sind insbesondere die Versionierung, Internationalisierung und Tenantfähigkeit von Datenmodellen, wobei letztgenannter Aspekt gerade auch im Kontext von *Database as a Service* eine wichtige Rolle spielt[1]. Zur Verdeutlichung der Herausforderungen im Zusammenhang mit funktionalen Aspekten sei das in

Abbildung 1 dargestellte Beispiel mit der(unvollständigen) relationalen Modellierung einer Stücklistenstruktur betrachtet. Neben den fachlichen Anforderungen, welche sich beispielsweise im Schema der Tabellen MODULSTRUKTUR und MODUL widerspiegeln, ist in diesem Beispiel die Integration des funktionalen Aspektes der Internationalisierung denkbar – von mehrsprachigen Texten (MODUL.BEZEICHNUNG) über verschiedene Preise und Währungseinheiten (MODUL.PREIS, MODUL.WÄHRUNG) bis hin zu marktspezifischen Stücklistenvarianten[5]. Abhängig von der Realisierung des funktionalen Aspektes kommt es dadurch zu einer mehr oder weniger starken Transformation des ursprünglich fachlich orientierten Datenmodells, wobei auch Attribute von jeglichen Aspekten unbeeinflusst bleiben (MODUL.TEILNR, MODUL.FLAGS).

Eine derartige Vermischung von fachlichen und funktionalen Anforderungen bei der Modellierung bringt einige Probleme mit sich. Eine direkte Folge ist die fehlende Abgrenzung von Zuständigkeiten, da eine Kapselung der funktionalen Aspekte nicht ohne weiteres möglich ist, wie beispielsweise bei der Modellierung verschiedener Business-Objekte durch die jeweiligen Experten bzw. Fachabteilungen. Stattdessen kann sich der Wirkungsbereich eines funktionalen Aspektes über das gesamte Datenmodell erstrecken und zur Anpassung einer Vielzahl von Relationen führen. Dadurch ist neben der eigentlichen Arbeit zur Realisierung des fachlich orientierten Datenbankmodells zusätzlicher Abstimmungsaufwand notwendig, um die Anforderungen bezüglich der funktionalen Aspekte adäquat zu berücksichtigen. Aufgrund einer solch engen Verzahnung ist weiterhin sowohl mit einem geringen Wiederverwendungsgrad als auch mit erhöhtem Wartungsaufwand der einzelnen Modellierungsobjekte zu rechnen, d.h. es ist beispielsweise nicht einfach möglich ein implementiertes Modul „Mehrsprachkeit“ aus einem Datenmodell zu isolieren und in einem anderen Anwendungskontext zu verwenden.

Nachfolgend wird in Abschnitt 2 ein Konzept zur Lösung der einleitend beschriebenen Probleme bezüglich der Integration funktionaler Aspekte vorgestellt, welches die Entwicklung strukturierterer Datenmodelle ermöglicht. Hierfür notwendige Strategien und Methoden zur Realisierung werden anschließend in Abschnitt 3 vorgestellt und in Abschnitt 4 bewertet, bevor Abschnitt 5 die vorliegende Arbeit zusammenfasst.

2. PRINZIP DER ASPEKTORIENTIERTEN DATENHALTUNG

Dieser Abschnitt stellt das Konzept der Aspektorientierten Datenhaltung vor, welches bezugnehmend auf die Beschreibung funktionaler Aspekte (siehe Definition 1) wie folgt formuliert werden kann.

►*Definition 2.* Die **Aspektorientierte Datenhaltung** ist ein Paradigma, welches eine Datenbankmodellierung unterstützt, in der funktionale Aspekte unabhängig von der fachlichen Modellierung integriert werden können. Vordergründig wird dabei das Ziel einer weitestgehenden Modularisierung sowohl der fachlichen als auch funktionalen Anforderungen verfolgt, sodass es zu einer möglichst minimalen Überschneidung dieser beiden Modellierungsebenen kommt.

Dabei grenzt sich das hier vorgestellte Konzept gegenüber den von RASHID postulierten **Aspect-Oriented Database Systems** (AODB) durch die grundsätzliche Zielstellung ab. Auf der einen Seite werden die AODB als Weiterentwicklung objektrelationaler bzw. -orientierter Datenbanksysteme dargestellt, wobei Techniken der AOP sowohl zur System-Implementierung selbst als auch für die Bereitstellung notwendiger Hilfsmittel zur Speicherung und Abfrage von Aspekten aus einer entsprechenden Programmiersprache heraus (z.B. AspectJ[2]) eingesetzt werden[13]. Auf der anderen Seite soll das Paradigma der Aspektorientierten Datenhaltung Möglichkeit aufzeigen, für funktionale Aspekte eine geeignete Abbildung basierend auf dem relationalen Datenbankmodell als praxisrelevanteste DBMS-Technologie zu definieren, um sie im Rahmen einer klassischen (nicht notwendigerweise aspektorientierten) Anwendungsentwicklung bzw. -evolution zur Verfügung zu stellen.

In einem ersten Unterabschnitt wird auf die Entstehung und Bedeutung des Begriffs der Aspektorientierten Datenhaltung eingegangen, deren Ziele und Eigenschaften anschließend Gegenstand der Betrachtung sind.

2.1 Begriffsbildung

Fundamentale Konzepte der Informationstechnologie sind die Modularisierung und Kapselung von Daten und Funktionalität in logische Einheiten. Dies äußert sich neben der Definition von Architekturen und Schichtenmodellen wie dem ANSI/SPARC-Modell[6] vor allem auch in der Entwicklung der Programmiersprachen. Beginnend mit Assemblersprachen über Prozedurale Sprachen bis hin zur Objektorientierten Programmierung (OOP) wurden Möglichkeiten geschaffen, um Programmierer bei der Entwicklung von wiederverwendbarem sowie leicht pflegbarem modularisierten Code zu unterstützen, welcher logisch zusammengehörende Funktionen kapselt.

Seit über einem Jahrzehnt hat sich als Weiterentwicklung der OOP das Paradigma der **Aspektorientierten Programmierung**[10] etabliert. Motiviert ist dieses Programmierkonzept aus dem Konflikt heraus, dass es selbst bei einer optimalen Modularisierung der Geschäftslogik nicht möglich ist, übergreifende Querschnitts-Funktionalität im Rahmen einer Anwendung sinnvoll zu kapseln. Diese auch mit *Cross-Cutting Concerns* oder *Aspekten* bezeichneten Anforderungen sind demzufolge über mehrere Module und logische Einheiten verteilt, womit sich die Pflege und Wartung sowie die Wiederverwendbarkeit des Codes deutlich aufwändiger gestaltet. Beispiele für derartige Aspekte sind Logging, Fehlerbehandlung oder Transaktionsverwaltung. Aspektorientierte Programmiersprachen wie AspectJ[2] bieten dem Entwickler mit *Join Points*, *Pointcuts* sowie *Advices* sprachliche Konstrukte die es ermöglichen, derartige Aspekte modularisiert zu realisieren. Anschließend findet im Prozess des *Weaving* eine Verzahnung des Codes statt.

Aufgrund der Analogie zwischen *Cross-Cutting Concerns* und den einleitend in Abschnitt 1 beschriebenen *funktionalen Aspekten* sowie der damit verbundenen Probleme bei der Integration in Applikationscode bzw. in ein Datenmodell wird das hier vorgestellte Modellierungsparadigma als „Aspektorientierte Datenhaltung“ bezeichnet. Dabei deutet der Begriff „Datenhaltung“ an, dass das Paradigma sowohl

Konzepte für die reine Modellierung (SQL-DDL) als auch bezüglich der Interaktion von Anwendungen mit einem solchen Datenmodell (SQL-DML) beinhaltet muss.

2.2 Eigenschaften

Entsprechend der einleitenden Definition 2 für die Aspektorientierte Datenhaltung werden im Folgenden zur Spezifizierung des Paradigmas wichtige Anforderungen und Eigenschaften formuliert.

Modularität: Die Integration funktionaler Aspekte soll in einer Form möglich sein, dass alle zu einem Aspekt gehörenden Informationen in einer logischen Modellierungseinheit (z.B. einem Relationsschema) zusammengefasst werden können und sich dadurch die Vermischung mit der fachlichen Abbildung vermeiden lässt.

Orthogonalität: Diese Eigenschaft umfasst zwei Dimensionen bei der Betrachtung von funktionalen Aspekten. Einerseits ergeben sich bei einer Menge von n verschiedenen Aspekten insgesamt $n!$ mögliche Reihenfolgen für deren Integration in ein Datenmodell. Andererseits ist zu erwarten, dass ein konkretes Attribut für beliebig viele $[0, \ldots, n]$ Aspekte eine Relevanz besitzt. Dadurch gilt für zwei Aspekte A_i und A_k bezüglich ihrer beeinflussten Attributmengen $\mathcal{M}$:

$$\begin{aligned} |\mathcal{M}(A_i) \cap \mathcal{M}(A_k)| &= c \quad (0 \leq c \leq m), \\ \text{wobei} \quad m &= max(|\mathcal{M}(A_i)|, |\mathcal{M}(A_k)|) \end{aligned}$$

Orthogonalität fordert nun für die Aspektintegration, dass sowohl alle potentiellen Reihenfolge-Varianten ein gegebenes Datenmodell in einen semantisch äquivalenten Zustand überführen als auch, dass es zu keinen Wechselwirkungen zwischen den Aspekten bezüglich ihrer beeinflussten Attributmengen kommt.

Lokalität: Die Abbildung von funktionalen Aspekten sollte derart gestaltet sein, dass bei der Integration eines Aspektes nicht das gesamte Datenmodell einer Transformation unterzogen werden muss, sondern möglichst minimale Änderungen nur an den notwendigen Stellen (Modellierungseinheiten) ausreichen.

Universalität: Das Konzept der Aspektorientierten Datenhaltung ist nicht auf einen konkreten Anwendungsbereich eingeschränkt. Um in der Praxis ein maximal breites Einsatzgebiet bezüglich realer DBMS-Produkte zu erreichen, werden zur Implementierung funktionaler Aspekte nur Konzepte der Datenbankmodellierung auf Basis des *SQL:92*-Standards[3, 9] verwendet.

Benutzbarkeit: Ein existierendes oder zu entwickelndes Datenmodell soll sich einfach um funktionale Aspekte erweitern lassen. Hierfür ist die Unterstützung des Nutzers (vorrangig des Anwendungs-Administrators) in Form eines Integrations-Assistenten notwendig (siehe Abschnitt 3.2), welcher nach Abfrage anwendungsspezifischer Informationen die eigentliche Transformation des Datenmodells vornimmt.

Basierend auf den beschriebenen Anforderungen können nun Ansätze zur Implementierung entworfen werden, der nachfolgende Abschnitt 3 vermittelt hierzu einen Überblick.

3. REALISIERUNGSSTRATEGIEN

Nachdem im bisherigen Verlauf der Arbeit das Paradigma der Aspektorientierten Datenhaltung motiviert und näher beschrieben wurde, widmet sich dieser Abschnitt der Präsentation geeigneter Ansätze für die Realisierung. Neben der eigentlichen Abbildung funktionaler Aspekte in einem Datenmodell (Abschnitt 3.1) umfasst dieses Thema entsprechend der Anforderung der *Benutzbarkeit* auch Strategien für die Unterstützung bei der Integration eines Aspekts (Abschnitt 3.2).

3.1 Aspektmodellierung

Zur Gewährleistung der in Abschnitt 2.2 formulierten Eigenschaften bietet sich das *EAV*-Konzept[11, 12] an, welches prinzipiell durch die drei Tabellen ENTITY, ATTRIBUTE und VALUE modelliert wird. Dabei werden Attribute eines Objektes (Entity) nicht auf traditionelle Weise in Form von Tabellenspalten abgebildet, sondern als Zeilen mit zugehörigem Attributwert in der Tabelle VALUE gespeichert. Damit ergibt sich die Möglichkeit, zusätzliche Informationen eines Objektes durch Hinzufügen von Daten (DML-Anweisung) zu realisieren, statt zwingend eine Schema-Evolution am bestehenden Datenmodell (DDL-Anweisung) durchzuführen.

Die grundlegende Idee des EAV-Konzeptes kann nun für die Aspektorientierte Datenhaltung genutzt werden, um die Auswirkungen eines funktionalen Aspektes auf die fachspezifischen Datenobjekte abzubilden. Zur Veranschaulichung der nachfolgenden Erläuterungen ist in Abbildung 2 das resultierende Datenmodell basierend auf dem Beispiel in Abbildung 1 dargestellt, wobei die Tabelle MODULSTRUKTUR aus Gründen der Übersichtlichkeit ausgeblendet wurde. Im Mittelpunkt des Realisierungsansatzes stehen drei Tabellen:

AspectDefinition: Die Tabelle beinhaltet die Stammdaten eines funktionalen Aspektes. Im vorliegenden Ansatz umfassen diese die Aspektbezeichnung (NAME), eine Beschreibung der Schlüsselwerte (KEYVAL) und einen Primärschlüssel (ASPDEFID). Die tatsächlichen Wertausprägungen beispielsweise für die verfügbaren Lokalen des Aspektes Internationalisierung sind in einer weiteren hier nicht dargestellten Tabelle hinterlegt.

AspectValue: Diese Tabelle dient zur Persistierung der eigentlichen Informationen fachlicher Datenobjekte bezüglich funktionaler Aspekte. Dabei wird für ein konkretes Attribut (ATTID) eines Datensatzes (ROWID) innerhalb einer bestimmten Tabelle (TABID) der neue aspektspezifische Wert (VALUE) hinterlegt und über den Primärschlüssel ASPVALID referenziert. Der Datentyp von VALUE muss hier so generisch wie möglich sein (z.B. NVARCHAR), um die verschiedenen Datentypen der referenzierten Tabellenattribute aufnehmen zu können. Der damit verbundenen untypisierten Speicherung wird mit Hilfe des Attributes TYPEID begegnet, welches die ursprünglichen Typinformationen aufnehmen kann. Wie die Attributnamen TABID, ATTID und TYPEID andeuten, verweisen sie als Fremdschlüssel auf entsprechende Verwaltungstabellen, welche jedoch aus Platzgründen nicht dargestellt sind. Zur besseren Veranschaulichung wurden deshalb statt IDs sprechende Attributwerte verwendet.

AspectAssign: Über diese Tabelle findet die tatsächliche Zuordnung aspektspezifischer Attributwerte zu funktionalen Aspekten statt. Dazu enthält ein Datensatz neben dem Primärschlüssel (AspAssID) einen Verweis sowohl auf den konkreten Aspekt (AspDefID) als auch den gepflegten Attributwert (AspValID) sowie den Wert (Value) bezüglich des Schlüsselkriteriums des referenzierten Aspektes. Beispielsweise stehen die Einträge mit AspAssID $\in \{1004, 1005\}$ für die Versionierung (AspDefID = 2) des Attributes Bezeichnung desjenigen Datensatzes in Modul mit dem Primärschlüsselwert 20080. Gleichzeitig ist mit „Y-Kabel“ eine dieser Versionen (AspAssID = 1006) auch übersetzter Attributwert für die Lokale „at“ im Kontext des Aspektes der Internationalisierung.

Voraussetzung für das beschriebene Konzept ist die Existenz einattributiger typgleicher Primärschlüssel in allen für funktionale Aspekte potentiell relevanten Tabellen, damit diese in AspectValue.RowID einheitlich referenziert werden können. Diese Forderung lässt sich jedoch immer gewährleisten, da entweder die betreffende Tabelle bereits einen solchen Primärschlüssel besitzt oder um einen zusätzlichen künstlichen Schlüssel mit der Option auto increment erweitert werden kann.

3.2 Aspektintegration

Für den praxistauglichen Einsatz des in Abschnitt 3.1 vorgestellten Realisierungskonzeptes ist wie in den Anforderungen erläutert zusätzlich eine Unterstützung des Nutzers bzw. Administrators im Prozess der Integration von funktionalen Aspekten in ein bestehendes oder neu zu entwickelndes Datenmodell notwendig. Der im Folgenden dargestellte Algorithmus umfasst die wichtigsten Schritte für einen solchen Integrations-Assistenten. Werden andere Realisierungskonzepte verwendet, können Abweichungen oder komplett neue Teilschritte notwendig sein.

1. Festlegung der Verbindungsparameter zur Datenbank.
2. Auswahl des zu integrierenden funktionalen Aspektes.
3. Festlegung von Tabellen und den jeweiligen Attributen, welche für den zu integrierenden Aspekt im Sinne der zugrundeliegenden Anwendung relevant sind. Beispielsweise ist beim Aspekt der Internationalisierung unter anderem die Entscheidung zu treffen, welche Tabellenattribute zukünftig übersetzbar (mehrsprachig) sein sollen.
4. *Optional:* Erstellung eines Backups der Datenbank.
5. Sicherstellung des einattributigen Primärschlüssels in allen gewählten Tabellen, gegebenenfalls durch Hinzufügen eines künstlichen Schlüssels wie am Ende von Abschnitt 3.1 beschrieben.
6. Erstellung der zum Realisierungskonzept gehörenden Verwaltungstabellen AspectDefinition, AspectValue und AspectAssign. Existieren diese Tabellen bereits, so ist nur ein zusätzlicher Eintrag in der Tabelle AspectDefinition für den neuen funktionalen Aspekt notwendig.
7. *Optional:* Bereitstellung einer View für jede zu Beginn ausgewählte Tabelle, welche in Verbindung mit den aspektspezifischen Verwaltungstabellen eine Gesamtsicht auf die fachlichen Objekte inklusive der zusätzlichen Daten zu den funktionalen Aspekten ermöglicht.

4. BEWERTUNG

Abschließend soll der präsentierte Realisierungsansatz anhand der für die Aspektorientierte Datenhaltung formulierten Anforderungen und Eigenschaften (Abschnitt 2.2) bewertet werden. Darüber hinaus erlaubt der Ansatz aufgrund des verwendeten EAV-Konzeptes unabhängig von funktionalen Aspekten die Abbildung beliebiger zusätzlicher Tabellenattribute auf Datensatzebene ohne Schemaänderung. Damit wird eine Semistrukturiertheit ermöglicht, wie sie vor allem bei der XML-Notation[7] bekannt ist.

Aufgrund der Kapselung aller notwendigen Informationen zur Abbildung von funktionalen Aspekte in den Tabellen AspectDefinition, AspectValue und AspectAssign ist die *Modularität* diesbezüglich sichergestellt. Eine Veränderung fachlicher Tabellen ist nur im Rahmen der Voraussetzung eines einattributigen typgleichen Primärschlüssels notwendig. Damit ist auch gleichzeitig die Eigenschaft der *Lokalität* erfüllt. Die Modellierung dieser drei beschriebenen Verwaltungstabellen basiert allein auf atomaren Attributen mit einfachen Domänen (int, varchar) sowie Fremdschlüsseln, dies genügt der Forderung nach *Universalität*.

Die Bewertung der *Orthogonalität* gliedert sich entsprechend der hier getroffenen Definition in zwei Teile. Einerseits ist für eine Menge funktionaler Aspekte die semantische Äquivalenz bezüglich ihrer möglichen Reihenfolgen gesichert, da sich die Integration aller Aspekte nur als zusätzliche Datenzeilen in den gemeinsam genutzten initial angelegten Verwaltungstabellen niederschlagen. Dabei sind keine aspektspezifische Änderungen an den fachlichen Tabellen notwendig. Andererseits können für zwei verschiedene Aspekte Wechselwirkungen in Hinblick auf die beeinflussten Attributmengen ausgeschlossen werden aufgrund der getrennten Speicherung aspektspezifischer Informationen zu fachlichen Daten (AspectValue) und der tatsächlichen Zuordnung zu Aspekten (AspectAssign) wie beispielhaft in Abschnitt 3.1 erläutert.

Schließlich ist mit dem in Abschnitt 3.2 vorgestellten Algorithmus die Grundlage für einen Integrations-Assistenten gelegt, um die *Benutzbarkeit* des gesamten Realisierungsansatz zu gewährleisten.

5. ZUSAMMENFASSUNG

Die vorliegende Arbeit präsentierte das Modellierungsparadigma der **Aspektorientierten Datenhaltung** mit dem Ziel, *funktionale Aspekte* als Vertreter von querschnittlichen Belangen in einer modularisierten Form im Datenmodell abzubilden. Dadurch lässt sich die Vermischung von fachlicher und funktionaler Modellierungsebene vermeiden, dies führt letztlich zur besseren Strukturierung, Wartbarkeit und Wiederverwendbarkeit eines Datenmodells.

Mit den Eigenschaften der *Modularität*, *Orthogonalität*, *Lokalität*, *Universalität* und *Benutzbarkeit* wurden grundsätzliche Anforderungen formuliert. Diese dienten anschließend

Modul				
TeilNr	Bezeichnung	Preis	Währung	Flags
19780	Schraube	0.75	€	00110101
20080	Kabel	0.10	€	00110011
...	...	...	...	...

AspectDefinition		
AspDefID	Name	KeyVal
1	Internationalisierung	Lokale
2	Versionierung	Revision
...	...	...

AspectValue					
AspValID	TabID	RowID	AttID	Value	TypeID
101	Modul	19780	Bezeichnung	screw	CHAR
102	Modul	19780	Preis	0.45	DEC
103	Modul	19780	Währung	$	CHAR
104	Modul	19780	Bezeichnung	bullone	CHAR
105	Modul	19780	Preis	0.78	DEC
106	Modul	19780	Währung	€	CHAR
107	Modul	20080	Bezeichnung	X-Kabel	CHAR
108	Modul	20080	Bezeichnung	Y-Kabel	CHAR
...	...	...	...	...	...

AspectAssign			
AspAssID	AspDefID	AspValID	Value
1001	1	101	us
1002	1	102	us
1003	1	103	us
1004	2	107	1
1005	2	108	2
1006	1	108	at
1007	1	104	it
1008	1	105	it
...	...	...	...

Zusätzliche Integritätsbedingungen:

- Unique-Bedingung in Tabelle AspectValue über (TabID, RowID, AttID, Value, TypeID)
- Unique-Bedingung in Tabelle AspectAssign über (AspDefID, AspValID, Value)
- AspectAssign.AspDefID ist Fremdschlüssel bezüglich AspectDefinition.AspDefID
- AspectAssign.AspValID ist Fremdschlüssel bezüglich AspectValue.AspValID

Abbildung 2: Modellierungsansatz für funktionale Aspekte

als Spezifikationsgrundlage für das am Beispiel vorgestellte Realisierungskonzept. Die abschließende Bewertung zeigte die Verwendbarkeit dieses Konzeptes für das Paradigma der Aspektorientierten Datenhaltung.

Zukünftige Arbeiten werden sich mit der Vervollständigung des skizzierten Realisierungsansatzes beschäftigen, wobei sowohl für die angedeuteten Probleme bezüglich des EAV-Konzeptes als auch für einen generischen applikationsseitigen Zugriff auf funktionale Aspekte Lösungsstrategien bewertet werden müssen. Parallel dazu besteht die Aufgabe einer prototypischen Implementierung des hier vorgestellten Konzeptes. Dies betrifft sowohl die reale Modellierung (relationale Abbildung) in einem existierenden DBMS-Produkt als auch die Entwicklung des Assistenten zur Integration von funktionalen Aspekten.

6. LITERATUR

[1] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-tenant databases for software as a service: schema-mapping techniques. In *SIGMOD Conference*, pages 1195–1206, 2008.

[2] O. Böhm. *Aspektorientierte Programmierung mit AspectJ 5: Einsteigen in AspectJ und AOP.* dpunkt.verlag, 2005.

[3] D. C and H. Darwen. *SQL - Der Standard. SQL/92 mit den Erweiterungen CLI und PSM.* Addison-Wesley, 1998.

[4] C. Ebert. *Systematisches Requirements Engineering und Management.* dpunkt.verlag, 2008.

[5] A. Göbel. Internationalisierung in Produktkonfiguratoren - Anforderungen und Konzepte für die Datenhaltung. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2009.

[6] T. Härder and E. Rahm. *Datenbanksysteme: Konzepte und Techniken der Implementierung.* Springer, 2001.

[7] E. R. Harold and W. S. Means. *XML in a Nutshell.* O'Reilly, 2005.

[8] A. Heuer, G. Saake, and K.-U. Sattler. *Datenbanken. Konzepte und Sprachen.* Mitp-Verlag, 2007.

[9] ISO/IEC 9075:1992, Information Technology - Database Language SQL, 1992. URL: `http://www.andrew.cmu.edu/user/shadow/sql/sql1992.txt` [Stand: 30.03.2010].

[10] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J. Irwin, and J.-M. Loingtier. Aspect-Oriented Programming. In *ECOOP*, pages 220–242, 1997.

[11] P. Nadkarni. An Introduction to EAV Design for Generic Clinical Study Data Management Systems, 2002. URL: `http://ycmi.med.yale.edu/nadkarni/Introduction%20to%20EAV%20systems.htm` [Stand: 30.03.2010].

[12] P. Nadkarni, L. Marenco, R. Chen, E. Skoufos, G. Shepherd, and P. Miller. Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. In *JAMIA*, 6, pages 478–493, 1999.

[13] A. Rashid. *Aspect-Oriented Database Systems.* Springer, 2004.