in: Tagungsband CAD'98 "Tele-CAD - Produktentwicklung in Netzen" , März 1998,
Informatik Xpress 9, pp. 167-176

Unterstützung dynamischer Typkonstruktion in Technischen Informationssystemen

Henrik Loeser, Nan Zhang, Jürgen Zimmermann
Universität Kaiserslautern, FB Informatik,
Postfach 3049, 67653 Kaiserslautern
E-Mail: {loeser | zhang | jnzimmer}@informatik.uni-kl.de

Kurzfassung:

1. Einleitung

Technische Informationssysteme (IS) erlauben die Flexibilisierung und Optimierung von Arbeitsabläufen bei gleichzeitiger Kostenreduktion. Größere Entwicklungsprozesse stellen jedoch besondere Anforderungen an ein IS. Durch neue Techniken für die Datenverwaltung, wie z. B. objekt-relationale Datenbankverwaltungssysteme (ORDBVS, [St96]), die die Speicherung aller Daten(typen) innerhalb einer Datenbank (DB) ermöglichen, und den Einsatz von Internet- bzw. WWW-basierten (World Wide Web, [Cal95, Loe97]) Verfahren, die Lösungen für die Arbeit in verteilten und heterogenen Umgebungen bieten, lassen sich nun auch Entwicklungsprozesse verbessern [HLZ97]. Zudem können durch Workflow-Management-Systeme (WfMS) die einzelnen Prozesse sowie die in ihnen verwendeten Daten koordiniert und so zeitlich und wirtschaftlich optimale Abläufe erreicht werden.

Im Rahmen des RITA-Projektes (Rechnergestütztes Informationssystem für Technische Anwendungen, [HT96]), das in Kooperation mit einem weltweit führenden Hersteller von Fahrzeugsitzen erfolgt, wird ein Technisches IS entwickelt, das die Versuchs- und Entwicklungsingenieure bei ihrer Arbeit unterstützen soll. Durch den Einsatz von Internet-basierten Techniken lassen sich Zugriffe auf das IS sowie die Zusammenarbeit der Ingenieure untereinander in der heterogenen und über ein unternehmensinternes Netzwerk (Intranet) verbundenen verteilten Umgebung realisieren. Um den Zugriff auf alle benötigten Informationen sicherzustellen, ist es notwendig, die unterschiedlichsten Datentypen („klassische" alphanumerische Daten, Zeichnungen, Bilder, Kurzfilme, etc.) geeignet zu modellieren und in einem System zur Verfügung zu stellen. Da sich jedoch ständig neue Anforderungen an das Unternehmen und damit auch an das IS ergeben, muß dieses an die neuen Bedingungen anpaßbar bzw. um neue Datentypen oder Funktionskomponenten erweiterbar sein.

In unserem Beitrag stellen wir ein Verfahren zur dynamischen Typkonstruktion vor, das es den Ingenieuren vor Ort an ihrem Arbeitsplatz ermöglicht, das IS auf einfache Weise um neue Typen zu ergänzen und zugleich Masken für die Datenein- und ausgabe zu erzeugen. Hiermit ist es z. B. einem Versuchsingenieur möglich, einen aufgrund neuer EU-Vorschriften vorgeschriebenen Prüfungstyp mit Hilfe der Vererbung als Subtyp in die Hierarchie der Prüfungstypen einzugliedern. Anschließend steht der neue Typ ihm und den anderen Ingenieuren zur Verfügung.

Im folgenden stellen wir kurz das RITA-Projekt vor. Daran anschließend diskutieren wir in Kapitel 3 den Einsatz objekt-relationaler Datenbanksysteme für technische Informationssysteme sowie ihre Möglichkeiten zur Typerweiterung. In Kapitel 4 präsentieren wir dann unser Konzept zur dynamischen Typerweiterung, wobei wir die einzelnen Schritte, angefangen von der Typverwaltung über die Typkonstruktion bis hin zum Einsatz der neu erzeugten Typen, besprechen. Unser Beitrag schließt mit einer Zusammenfassung sowie einem Ausblick auf die weitere Arbeit.

2. RITA

Ziel des RITA-Projektes ist die prototypische Erstellung eines IS zur Unterstützung der Arbeitsabläufe in der Abteilung für Musterbau und Versuch des Kooperationspartners. Wichtigste Aufgabe ist es dabei, die Daten und Erfahrungen aus früheren Entwicklungsprojekten und den dabei durchgeführten Versuchen den Ingenieuren an ihrem Arbeitsplatz geeignet zur Verfügung zu stellen. Auf diese Weise soll die Vorbereitung und Durchführung neuer Projekte und der einzelnen Versuche vereinfacht, genauer geplant und dabei optimiert werden. So sollen die Entwicklungszeiten verkürzt und die Erreichung der Projektziele sicherer gemacht werden. Das IS soll hierzu ständig um Erfahrungen neuer Projekte sowie die Daten gerade durchgeführter Versuche ergänzt werden. Für eine Anpassung an sich ständig ergebende neue Anforderungen muß das IS um neue Datentypen und Funktionsmodule erweiterbar sein.

Im Rahmen des RITA-Projektes werden die Modellierung der anfallenden Daten und Operationen, die Steuerung und Überwachung der einzelnen Schritte eines Entwurfsprojektes sowie die Integration des IS in eine verteilte, heterogene Client/Server-Umgebung untersucht.

3. ORDBVS

3.1 Anwendungsanforderungen und ORDBVS

In einem technischen IS soll durch das Datenbanksystem (DBS) eine effiziente Verwaltung und Verarbeitung von verschiedenartigen Daten zur Verfügung gestellt werden, die typischerweise folgendes Spektrum von Merkmalen aufweisen:

- große, einfach strukturierte Datenmengen, z. B. Kataloge für Bauteile oder Normen;

- stark vermaschte Datenstrukturen, etwa CAD-Objekte oder technische Zeichnungen;
- neue Datentypen wie Bild oder Video, beispielsweise zur Darstellung von Details und Resultaten aus Versuchen.

Diese datenbezogenen Anforderungen können weder von relationalen noch von rein objektorientierten DBVS erfüllt werden. Traditionelle relationale DBVS bieten erhebliche Vorteile in Bezug auf Einfachheit der Modellierung und Verarbeitung, was bei einfach strukturierten Objekten und weniger komplexen Anwendungsproblemen die Datenverwaltung und Systementwicklung erleichtert. Zusätzlich begründet die Deklarativität relationaler DB-Sprachen einen hohen Grad an Datenunabhängigkeit und die Möglichkeit des mengenorientierten DB-Zugriffs, was dem DBVS Optimierung erlaubt. Objektorientierte Datenmodelle hingegen besitzen beträchtliche Vorzüge bei der Modellierung komplexer Datentypen, der Spezifikation des Objektverhaltens sowie der Strukturierungs- und Wiederbenutzungsmöglichkeit von Objekten, erzwingen jedoch oft navigierende, d. h. satzorientierte Verarbeitung und gelten für leistungskritische Anwendungen als ungeeignet.

Daher werden objekt-relationale Datenmodelle (ORDM) und ORDBVS angestrebt, die darauf abzielen, die Vorteile von objektorientierten und relationalen Welten zu vereinen. Sie sind momentan weltweit Gegenstand intensiver Forschungsbemühungen, die bereits zu verschiedenen Modellvorschlägen und Systemimplementierungen führten [Cha96, Kim96, St96]. Für unser Projekt verkörpern sie eine ideale Basis, insbesondere durch ihre anwendungsbezogenen Erweiterungsmöglichkeiten, die Unterstützung neuer Datentypen wie Text, Bild und HTML-Seite, eine deskriptive Anfragesprache und die Bereitstellung aktiver Fähigkeiten (*Trigger*, *Alerter*) als vielseitig einsetzbares Konzept.

3.2 Typerweiterbarkeit in ORDBVS

Die Entwicklungstendenz zeichnet sich insbesondere durch die Erweiterungsfähigkeit des Typsystems aus. Diese Eigenschaft erlaubt es, durch die Definition geeigneter Datentypen und zugehöriger Operationen anwendungsspezifische Verarbeitung in das ORDBVS zu integrieren. Durch die Kombination von ADTs, Referenztypen, Typkonstruktoren sowie typischen OO-Merkmalen wie Vererbung wird eine hohe Ausdrucksmächtigkeit erzielt. Folgende Konzepte werden dabei zur Verfügung gestellt:

- Mit ADTs kann man Typ-spezifisches Verhalten gekapselt definieren.
- Mit Typkonstruktoren lassen sich mehrwertige Attribute und geschachtelte Strukturen spezifizieren, was die Modellierungsmächtigkeit und -flexibilität wesentlich erhöht.
- Referenztypen ermöglichen eine eindeutige Tupel-Identifikation und eine direkte Darstellung von Beziehungen, was eine Alternative zu wertbasierter Primär-/Fremdschlüssel-Verknüpfung bietet.

- Vererbung vereinfacht die Definition neuer Typen und bietet einen natürlichen Rahmen für Generalisierung/Spezialisierung.
- Tabellen (Relationen) können durch komplexe Wertebereiche, Objekteigenschaften und „*composite types*“ (d. h. „*row types*“ in SQL3 [ISO96]) erweitert werden. So lassen sich mit Row-Typen, ADTs und Referenzen geschachtelte Strukturen aufbauen.

Durch ein erweiterbares Typsystem bieten ORDBVS wesentlich mehr Modellierungsalternativen und größere Flexibilität, was einer genaueren und angemesseneren DB-Modellierung zugute kommt, aber zugleich den DB-Schemaentwurf deutlich komplexer gestaltet. Außerdem sollte es in RITA möglich sein, den DB-Entwurf inkrementell durchzuführen, um die dynamische Anpassung und Erweiterung zu erlauben, für den Fall, daß später neue Anforderungen hinzukommen, z. B. wegen neuer technologischer Entwicklungen oder erweiterter/geänderter Systemaufgaben. Letzteres gilt auch deshalb, weil die Anwendungsentwicklung oft inkrementell durchgeführt wird und beim IS-Entwurf noch nicht in vollem Umfang bekannt ist. Ein wichtiger Grundmechanismus in RITA ist daher die Unterstützung von dynamischen Typerweiterungen, was den Zugriff auf die Metadaten des DBVS erfordert.

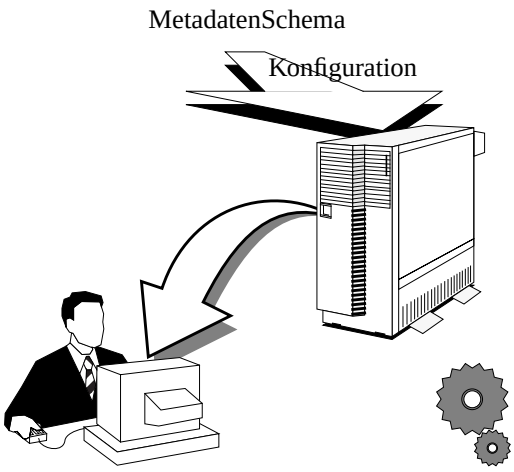
Es ist allerdings festzustellen, daß sich in den Ansätzen der einzelnen DBS-Hersteller das Angebot an vordefinierten Typen sowie die interne Realisierung der Metadaten- bzw. Typverwaltung sehr stark unterscheiden [IBM95, Inf96, Ora96, Uni96]. Dies wird besonders deutlich, wenn man die sich noch in der Anfangsphase befindenden konzeptionellen wie auch die realisierungstechnischen Entwicklungen genau betrachtet. Für uns ist „objekt-relational‘ bisher eher eine Ideensammlung und bezeichnet kein homogenes und wohldefiniertes Datenmodell [ZH97].

Aus diesen Gründen ist es wichtig, ein hinreichend allgemeines Rahmenkonzept für die Typverwaltung zu entwickeln, wodurch eine Entkoppelung von konkreten ORDBVS erreicht wird. Gleichzeitig sollen dabei alle Möglichkeiten der Typerweiterung offengehalten und systemseitig unterstützt werden.

4. Dynamische Typkonstruktion

Eine Systemkomponente zur dynamischen Definition neuer Datentypen für ein beliebiges (aber festes) ORDBVS erfordert die Lösung verschiedenartiger Probleme. Als Basis für das Gesamtsystem dient eine einheitliche Metadatenverwaltung, die durch Konfiguration an verschiedene ORDBVS angepaßt werden kann. Um ein komfortables Arbeiten mit den Metadaten zu ermöglichen, erlaubt ein graphisches Werkzeug („RITA-Types“) die interaktive Erweiterung der bestehenden Typbasis um neue Datentypen. Diese werden durch einen DDL-Generator („RITA-Gen“) in das eigentliche ORDBVS-Schema eingebracht werden.

Für die Darstellung bzw. Eingabe von Daten für selbst erzeugte Datentypen können zum einen durch „RITA-HTML“ erzeugte HTML-Schablonen benutzt werden, zum anderen ein generischer Browser „RITA-Show“.



erzeugt werden (mit RITA-HTML) bzw. kann ein allgemeiner Browser (RITA-Show)

Der gesamte Ablauf von der Phase der Konfiguration bis zur Datenein- und -ausgabe ist nochmals in Bild X dargestellt. In einem ersten Schritt wird die Metadatenverwaltung auf dem Ziel-ORDBVS installiert und für dieses konfiguriert.

// Wie sieht der Gesamtablauf durch das System aus? -> Erklaeren der einzelnen Systemkomponenten
// Welche Anforderungen haben wir an das System? (Typkonstruktion an sich) => Schema
// Wie sieht das Konfigurieren aus?
// Arbeiten mit RITA-Types
// Generieren von DB-Schemata
// Dazu noch das Generieren von HTML Seiten fuer Datenaustausch.
// Browser-Applikation RITA-Show/Browse

4.1 Metadatenverwaltung

Der interne Aufbau der Metadatenverwaltung (MDV) ist so angelegt, daß er eine möglichst breite Basis von Konzepten der unterschiedlichen ORDBMSen unterstützt. Entsprechend erlauben die Metadaten die Speicherung von viel mehr Informationen, als sie ein konkretes ORDBVS anbietet. Um die Metadatenverwaltung jedoch für ein ORDBVS konsistent zu halten, d. h. nur die Konstrukte zu erlauben, die das ORDBVS auch wirklich kennt, muß (und kann) die MDV für dieses konfiguriert werden. Diese Konfiguration erlaubt es allen anderen Applikationen, die MDV so zu benutzen, als ob sie für das konkrete ORDBVS entworfen wurde.

Eine Anforderungen an die MDV ist sicherlich der einheitliche Zugriff auf die Metadaten, unabhängig vom eingesetzten ORDBVS. Deshalb sind die Metadaten in einem relationalen Schema abgelegt, das die volle Funktionalität unter ausschließlichem Einsatz von ANSI SQL-2 (Entry Level) erlaubt, die jedes ORDBVS unterstützt.

Da die Metadatenverwaltung an sich nicht an ein bestimmtes ORDBVS gebunden sein soll, erlaubt sie eine einfache Anpassung an verschiedene System durch Konfiguration von Parametern wie vordefinierte Typen und Konstruktoren sowie Vererbungsmechanismus. Die Metadatenverwaltung legt in diesem Sinne also nicht von Anfang an eine bestimmte Menge von zur Verfügung gestellten Typen und Konstruktoren fest, sondern nur eine Vorgehensweise, wie aus diesen vorhandenen Parametern neue Typen erzeugt werden können. Die Semantik (und damit auch die Einschränkungen für ein bestimmtes ORDBVS) wird bei der Konfiguration der Metadatenverwaltung für ein konkretes System bestimmt.

4.2 Konzepte zur Typkonstruktion

Klassifikationen

Datentypen werden in den Metadaten nach der Art ihrer Unterstützung durch das ORDBVS sowie nach ihrer Erweiterbarkeit klassifiziert. Bei der Systemunterstützung unterscheiden wir *atomare* und *zusammengesetzte* Datentypen. Atomare Datentypen sind dem ORDBVS direkt bekannt (und somit für selbst nicht weiter zerlegbar). Beispiele hierfür sind SQL-2 Datentypen wie BIT, NUMBER usw. Aber auch ORDBVS-spezifische *built-in* Typen wie DataBlades oder Extenders sind atomar. Zusammengesetzte Datentypen sind solche, die durch den Benutzer über die später erklärte Vorgehensweise definiert wurden.

Orthogonal zu dieser Klassifikation über die Systemunterstützung hinsichtlich ihrer Erweiterbarkeit im Sinne der Vererbung klassifiziert werden. Hierbei existieren die beiden Alternativen erweiterbar (*extensible*) und nicht erweiterbar (*final*). Nicht erweiterbar sind beispielsweise die SQL-2 Typen, als erweiterbar könnte ein Datentyp „Image“ als DataBlade gelten.

Beispiele für atomare Datentypen, einmal nicht erweiterbar, einmal erweiterbar.

```
//      DECLARE TYPE NUMBER FINAL;
//      DECLARE TYPE Image EXTENSIBLE;
```

Zusammengesetzte Datentypen

Zusammengesetzte Datentypen werden durch Aggregation in Verbindung mit Vererbung gebildet. Über Vererbung können Eigenschaften bereits definierter erweiterbarer Datentypen

Vererbung

Neben der Aggregation ist die Vererbung ein weiteres Konzept für die Erstellung zusammengesetzter Datentypen. Hierbei können bereits definierte erweiterbare Datentypen vererbt werden, wobei deren Anzahl ein Konfigurationsparameter ist. Bei einer Mehrfachvererbung sind die üblichen Probleme wie Namens- und Struk-

turkonflikte gegeben und müssen entsprechend von dem Konstruktionswerkzeug „RITA-Types“ behandelt bzw. verhindert werden.

Die Wertigkeit der Vererbung wird als Konfigurationsparameter für das jeweilige ORDBVS festgelegt.

Den für die Aggregation zuständigen Konstruktor nennen wir „Klassenkonstruktor“. Dieser ist als einziger Konstruktor in unserer Metadatenverwaltung typbildend, d. h. nur durch ihn können neue (benannte) Datentypen erzeugt werden. Ein neuer zusammengesetzter Datentyp besteht aus Attributen sowie seiner Einordnung in eine bestehende Vererbungshierarchie.

Ein Attribut trägt einen eindeutigen Bezeichner und besitzt einen festgelegten Attributtyp und Attributkonstruktor. Als Attributtyp kann jeder bereits definierte Datentyp verwendet werden.

Ein Attributkonstruktor fungiert als Multiplikator für die Aggregation in einem Attribut. Er beschreibt, auf welche Art und Weise in einem Attribut die !!!

Der einfachste (und immer definierte) Attributkonstruktor ist der „Single“-Konstruktor, der angibt, daß das Attribut genau einen Wert aggregiert. Weitere Attributkonstruktoren, insbesondere solche zur Mengendefinition wie „Set“ und „MultiSet“, aber auch „List“ und „Array“ können bei der Konfiguration für ein ORDBVS festgelegt werden. Durch die Möglichkeit, jederzeit beliebig Attributkonstruktoren hinzuzufügen, können die bestehenden Attributkonstuktoren um neue erweitert werden, um sie danach für die Konstruktion neuer Datentypen zu verwenden.

Beispiel

Als Beispiel zeigen wir, wie ein neuer Prüfungstyp in die bestehende Vererbungshierarchie des Typs „Pruefung“ eingefügt wird, der den Ursprungstyp um eine Menge von „Bildern“ und eine Liste von Belastungswerten erweitert.

```
//      DEFINE TYPE Belastungspruefung EXTENDS Pruefung EXTENSIBLE
        bilder: Set Image;
        belastungsdaten: List FLOAT;
        END;
```

ER-Schema der Metadaten

Hier beschreiben in Form eine erweiterten ER-Diagramms den Aufbau der Metadaten, in denen die kompletten Informationen über die Datentypen gehalten werden.

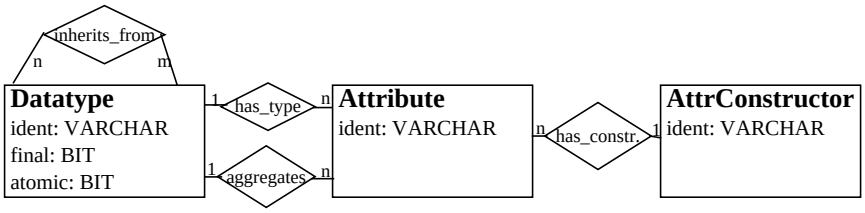


Bild 1: ER-Diagramm für Metadaten-Schema

4.3 Das Werkzeug „RITA-Types“

Das graphische Typkonstruktionswerkzeug „RITA-Types“ wurde entwickelt, um dem Ingenieur vor Ort die Möglichkeit zu geben, selbst in einer attraktiven, interaktiven Umgebung für seine Anwendung maßgeschneiderte Datentypen zu entwickeln, ohne daß er sich um Dinge wie Schemaentwurf und Datenbankprogrammierung kümmern muß. Entsprechend kann der Benutzer die vorhandenen Informationen in allen Detailstufen (mit resp. ohne ererbte Attribute, auf mehreren Ebenen) durchforsten und bekommt somit einen Überblick über bereits vorhandene Typen.

Von diesen Informationen ausgehend kann er sich entschließen, einen neuen Datentypen zu erzeugen und in die vorhandene Vererbungshierarchie einzugliedern. Für die Konstruktion unterstützt RITA-Types den Benutzer dahingehend, daß es für die Aggregation alle bekannten Aggregationskonstruktoren sowie die verwendbaren Attributtypen auflistet. Ebenso wird für Einbindung in die Vererbungshierarchie eine Liste von erweiterbaren Datentypen angezeigt, aus denen man eine Auswahl treffen kann.

Der Ingenieur kann sich aber auch dazu entschließen, einen bereits definierten Datentypen zu bearbeiten hinsichtlich seiner Einordnung in eine Vererbungshierarchie oder seiner Attribute. Dies wird durch RITA-Types allerdings nur für den Fall erlaubt, daß der zugehörige Datentyp noch nicht in das Schema des ORDBVS übernommen wurde, da ansonsten die Auswirkungen dieser einfachen Änderungsmöglichkeit auf vorhandene Daten (Schemaevolution, Anpassung abhängiger Applikationen) nicht kontrollierbar wären.

Da RITA-Types als Applet realisiert ist, kann es in jedem Java-fähigen Browser ausgeführt werden. Aber gerade dies bedeutet für den Rechner, auf dem das ORDBVS installiert ist, eine Zusatzbelastung, da — bedingt durch das Sicherheitskonzept von Java für Applets, die nur mit ihrem Ursprungsrechner in Kontakt treten dürfen — auf diesem zusätzlich ein Web-Server installiert werden muß für die Bereitstellung des Applets.

4.4 Generierung von Schema & Schablonen

In diesem Abschnitt besprechen wir, in welcher Hinsicht die in den Metadaten gehaltenen Informationen für Aktionen verwendet werden, nachdem neue Datentypen erzeugt wurden.

Schemagenerierung

An erster Stelle dieser weiteren Aktionen steht sicherlich die Schemagenerierung für das ORDBVS durch das Werkzeug „RITA-Gen“. Bei diesem Schritt handelt es sich im wesentlichen um eine Transformation der Informationen über Datentypen aus der Metadatenverwaltung in für das ORDBVS gültige Schemadefinitionen, damit die neu definierten Datentypen anschließend als vollständig in das ORDBVS integrierte Datentypen verwendet werden können. Hierbei ergibt sich die Notwendigkeit, daß nach der Definition eines Datentyps im ORDBVS dieser in der Metadatenverwaltung gesperrt wird, da eine Modifikation ansonsten eine Schemaevolution bewirken würde. Weiterhin müssen wir darauf achten, daß ein Datentyp lediglich einmal in dem ORDBVS definiert wird, besonders, da wir eine iterative und evolutionäre Vorgehensweise unterstützen, in der immer neue Datentypen erzeugt und in das DB-Schema aufgenommen werden können.

Generierung von HTML-Schablonen

Für die HTML-basierte Arbeit mit den neu erzeugten Datentypen ist die Erzeugung vier unterschiedlicher Dateien notwendig. Dies sind HTML-Masken für die Suche und Dateneingabe sowie spezielle, von einem CGI-Programm interpretierbare um SQL-Anweisungen angereicherte HTML-Dokumente. Während sich einfache, alphanumerische Daten- bzw. Attributtypen auf Texteingabefelder bzw. auf die textuelle Wertdarstellung abbilden lassen, müssen Mengen- und BLOB-Typen anders behandelt werden. Für stark geschachtelte Typen ist eine Umsetzung generell nicht sinnvoll.

Die Dateneingabe von BLOBs kann einfach über den neuen HTML-*Input*-Typ *File* erfolgen, der die Übertragung einer (dialoggestützt) spezifizierten Datei vom Client-Rechner zum WWW-Server erlaubt. Auf diese Weise können z. B. vom Ingenieur visualisierte Meßdaten in das ORDBVS eingebracht werden. Für die Anzeige werden die BLOBs in die vom CGI-Programm erzeugten HTML-Dokumente eingebunden, der WWW-Browser ist für die Handhabung verantwortlich, z. B. die Darstellung von Bildern im Dokument oder den Aufruf von externen *Viewern* für andere Typen (z. B. Postscript-Dateien mit *ghostview*). Mengenattribute erfordern ein noch abgestimmteres Vorgehen: Zwar lassen sich numerische Mengen durch Komma-separierte Aneinanderreihung der Werte bei der Eingabe unterstützen, für andere Typen ist die HTML-basierte Eingabe jedoch nicht sinnvoll. Bei der Ausgabe können für alphanumerische Typen Listen verwendet werden.

5. Zusammenfassung und Ausblick

In diesem Beitrag haben wir ein Verfahren zur dynamischen Typkonstruktion vorgestellt. Es ermöglicht dem Ingenieur, vor Ort an seinem über das Intranet mit dem zentralen DB-Server verbundenen Arbeitsplatz, neue Datentypen zu erzeugen, in die bestehende Typhierarchie des IS einzubringen und zudem Masken für die Datenein- und -ausgabe zur Verfügung zu stellen. Auf diese Weise kann das IS von seinen Benutzern an sich ergebende neue Anforderungen angepaßt werden. Durch den Einsatz WWW-basierter Techniken ist eine Nutzung des IS von jedem mit einem WWW-Browser ausgestatteten Arbeitsplatz aus möglich, wodurch die sonst in einer verteilten, heterogenen Client/Server-Umgebung auftretenden Probleme vermieden und Kosten gespart werden können [HLZ97]. Das vorgestellte Verfahren und seine Werkzeuge sind nicht an ein System aus dem sich entwickelnden und ständig ändernden Markt der ORDBVS gebunden, sondern durch einfache Anpassung der Abbildungstabellen bzw. der Grunddaten auf anderen Systemen einsetzbar.

Das besprochene Werkzeug RITA-Types fehlt z. Zt. noch weitere wünschenswerte, die Typerweiterung erleichternde Funktionalität. So wird die Erzeugung von Referenzen noch nicht unterstützt. Hier ist die Einführung eines „Referenzkonstruktors“ zur expliziten Benennung des referenzierten Typs bzw. Attributs notwendig. Ausgehend hiervon können dann zur einfacheren und fehlerfreien Dateneingabe für Referenzen HTML-Auswahllisten mit gültigen Werten erzeugt werden.

Das vorgestellte Verfahren eignet sich besonders für die Erstellung einfache, d. h. nicht geschachtelte und nicht vermaschte, Datentypen. Für andere Typen wird nach wie vor eine, zumindest teilweise, manuelle Erstellung oder Nachbereitung benötigt. Das dem Verfahren der Nutzung von HTML zugrundeliegende Verarbeitungsmodell der (sehr) kurzen Transaktionen ist nicht für alle Applikationen einsetzbar, zudem ist HTML bei der Unterstützung nicht-einfacher Datentypen schnell an seiner Leistungsgrenze. Hier, insbesondere wenn die hochfrequente Navigation durch komplexe, vermaschte Daten benötigt wird, eignet sich das Checkout/Checkin-Modell mit Client-seitiger Datenpufferung. Dann wird auch nicht mehr das HTTP, sondern z. B. CORBA oder andere Kommunikations- und Übertragungsprotokolle eingesetzt. Für den anfangs motivierten und an Beispielen gezeigten Einsatzbereich ist das vorgestellte Verfahren jedoch approbat.

6. Literatur

Cha96 Chamberlin, D.: *Using the New DB2: IBM's Object-Relational Database System*, Morgan Kaufmann, 1996.

Cal95 Cailliau, R.: *A Little History of the World Wide Web*, WWW-Konsortium, http://www.w3.org/History.html, 1995.

GJS96 Gosling, J., Joy, B., Steele, G.: *The Java Language Specification*, Addison-Wesley, 1996.

HLZ97 Härder, T., Loeser, H., Zhang, N.: *Supporting Adaptable Technical Information Systems in Heterogeneous Environments — Using WWW and*

ORDBMS, Proc. of the 8th Int. Workshop on Database and Expert Systems Applications (DEXA'97), Toulouse, Sept. 1997, S. 295-303.

HT96 Härder, T., Thomas, J.: *RITA — ein rechnergestütztes Informationssystem für technische Anwendungen*, ITG-Fachbericht 137 (STAK'96), München, März 1996, S. 111-126.

IBM95 *IBM DB2 SQL Reference — for common servers (Version 2)*, IBM Corp., 1995.

Inf96 *Informix and Illustra Merge to Create Universal Server*, Whitepaper, Informix Software, Inc., 1996.

ISO96 ISO/IEC CD 9075 Committee Draft, *Database Language SQL*, Jim Melton (eds.), July 1996.

Kim96 Kim, W.: *Object-Relational — The Unification of Object and Relational Database Technology*, Whitepaper, UniSQL Inc., Austin, 1996.

Loe97 Loeser, H.: *Datenbankanbindung an das WWW - Techniken, Tools und Trends*, in: Tagungsband der GI-Fachtagung „Datenbanksysteme in Büro, Technik und Wissenschaft" (BTW'97), K. R. Dittrich, A. Geppert (Hrsg.), Informatik aktuell, Ulm, März 1997, Springer-Verlag, S. 83-99.

Ora96 *Oracle7 Sever SQL Reference (Release 7.3)*, Oracle Corp., Feb. 1996.

St96 Stonebraker, M.: *Object-Relational DBMSs — The Next Great Wave*, Morgen Kaufmann Publ., Inc., 1996.

Uni96 UniSQL Server User's Guide, UniSQL, Inc., 1996.

ZH97 Zhang, N., Härder, T: *On Modeling Power of Object-Relational Data Models in Technical Applications*, Proc. of the 1st East-European Symposium on Advances in Databases and Information Systems (ADBIS'97), St. Petersburg, Sept. 1997, S. 318-325.

SQL2 ANSI X3.135-1992, „Database Language SQL"

7. Offene Probleme

- Zugriffschutz für die Metadatenverwaltung (Manipulation bestehender Datentypen)
- Versionierung der Datentypen sowie Konfigurierung einer Generierung auf die Datenbank

Ein erstes Ziel von RITA ist es, diese Informationen den Ingenieuren vor Ort an ihrem mit Computern ausgestattetem Arbeitsplatz zur Verfügung zu stellen und die direkte Erfassung neuer Ergebnisse zu ermöglichen. Hierzu eignen sich Internet- bzw. WWW-basierte (*World Wide Web*) Verfahren, wie der Einsatz von HTML-Dokumenten (*HyperText Markup Language*) oder Java-Programmen [GJS96], besonders gut [HLZ97]. Zum einen sind sie in dem typischerweise heterogenen Umfeld überall einsetzbar, zum anderen ermöglichen sie aber wegen der zentralen Verwaltung von Dokumenten und Programmen und unter Nutzung eines unternehmensinternen Netzwerkes (*Intranet*) eine kostengünstige Bereitstellung und Wartung. HTML-Seiten können dabei zur Datenein- und -ausgabe bei einfachen und nicht kritischen Verarbeitungsschritten dienen, beispielsweise der Erfassung von Meßdaten nach oder während eines längeren Tests. Java-Programme hingegen eignen sich bei — bezogen auf Darstellung, Transaktionsanforderungen oder interne Koordination/Kooperation — anspruchsvolleren Vorgängen, da sie eine vollständige Verarbeitungslogik auf der Client-Seite zur Verfügung stellen können.

Die Verwaltung der Informationen geschieht zentral in einer Datenbank, allgemeine Beschreibungen können aber auch als HTML-Seiten im Dateisystem abgelegt werden. Aufgrund der unterschiedlichen Datentypen der zu verwaltenden Informationen, wie z. B. lange Texte, Bilder, kurze Filmsequenzen und zudem „normale“ Zahlen und Zeichenketten, sowie der benötigten Erweiterbarkeit und erforderlichen Zugriffsmöglichkeiten bietet sich der Einsatz objekt-relationaler Datenbanksysteme (ORDBMS, [St96]) an [ZH97]. ORDBMSs befinden sich noch in der Entwicklungsphase, und die Ansätze der einzelnen DBS-Hersteller unterscheiden sich stark. Zwar steht ein um objektorientierte Konzepte erweitertes SQL für mengenorientierte Anfragen in ähnlicher Form für jedes ORDBMS zur Verfügung, doch das Angebot an Typen sowie die interne Realisierung der Metadaten- bzw. Typverwaltung und der Zugriff auf sie unterscheiden sich sehr stark.

In RITA wird jedoch für die Realisierung von dynamischen Typerweiterungen, wie sie z. B. für die Aufnahme eines neuen Prüfungstyps in eine Prüfhierarchie erforderlich sind, der Zugriff auf die Metadaten des DBMS benötigt. Möchte man sich nicht an ein bestimmtes ORDBMS binden, um RITA für den Einsatz mit beliebigen ORDBMS offen zu halten, so ist eine eigene Typ- bzw. Metadatenverwaltung erforderlich.ORDBMS

Die von uns gezeigte Typkonstruktion, verbunden mit einer eigenen Metadatenverwaltung, steht am Anfang einer Handlungskette und repräsentiert eine der Grund-

voraussetzungen für ein Informationssystem für technische Anwendungen. Mit Hilfe der vorgestellten Werkzeuge haben auch „Laien“ die Möglichkeit, auf einfache Art und Weise die für ihre Arbeit benötigten Datentypen zu definieren und effektiv zu nutzen.

Der fortschreitende Kosten- und Konkurrenzdruck in vielen Bereichen der Industrie macht es notwendig, in allen Bereichen durch Maschinen- und Computer-Einsatz zu rationalisieren sowie nach einer Optimierung von Arbeitsabläufen zu suchen. Ziel dieser Bemühungen ist die Kostenreduzierung der einzelnen Fertigungs- und Etwicklungsprozesse. Während bei der Produktion durch vermehrten Maschineneinsatz und neue Produktionsmethoden eine Kostensenkung erreicht wird, sind bei den Entwicklungsprozessen andere Lösungen gefragt. Da diese Prozesse meist Computer-unterstützt (Computer Aided, CAx) erfolgen, scheinen hier eher Lösungen aus dem Bereich der Informatik geeignet.

Datentypen können vollkommen neu entwickelt oder von vorhandenen Typen abgeleitet und anschließend als Spaltendomänen, als Attributtyp anderer abstrakter Datentypen (ADT) oder als Typ für Tabellenzeilen benutzt werden. Daher wird eine mächtige Ausdrucksmöglichkeit erzielt zusammen durch ADT, Row-Typen, Referenztypen, Typkonstruktoren sowie typischen OO-Merkmalen wie Vererbung:

Es ist daher vielmehr von Bedeutung, eine Eigenerweiterung zu finden, die eine zugeschnittene, aber gleichzeitig ausbaufähige Basis für die Typ- bzw. Metadatenverwaltung bietet, was RITA für den Einsatz mit beliebigen ORDBVS auch offenhalten kann.

Atomare Datentypen

Atomare Datentypen werden durch ein konkretes ORDBVS direkt unterstützt und sind zum einen in ANSI SQL-2 definierte Datentypen, zum anderen ORDBVS-spezifische *built-in* Typen. Des weiteren können — je nach Art des ORDBVS — noch benutzerspezifische Erweiterungen (UDF) wie DataBlades oder Extenders als atomare Typen aufgenommen werden. Atomare Datentypen werden der Metadatenverwaltung lediglich namentlich bekanntgemacht, ihr interner Aufbau ist dieser jedoch unbekannt. Die ANSI SQL-2 Datentypen sind der Metadatenverwaltung

automatisch bekannt, weitere atomare Datentypen können jedoch jederzeit hinzugefügt werden.

Erweiterbare und nicht erweiterbare Datentypen

Für jeden Datentype in der Metadatenverwaltung kann festgelegt werden, ob dieser erweiterbar (*extensible*) oder nicht erweiterbar (*final*) ist. Während für die atomaren Datentypen aus ANSI-SQL-2 automatisch festgelegt ist, daß diese nicht erweiterbar sind, können ORDBVS spezifische atomare Datentypen durchaus erweiterbar sein, bspw. könnte es sinnvoll sein, für eine Anwendung den atomaren Datentyp „Image“ (als DataBlade) um Attribute für einen Crash-Test zu erweitern. Andererseits kann es auch wünschenswert sein, selbst erstellte Datentypen als nicht erweiterbar zu kennzeichnen um z. B. zu verhindern, daß von einem Datentyp Abkömmlinge in der Datenbank auftauchen.

Sie können später, bei einer Unterstützung durch „RITA-Types“, durch Zusatzanfragen, deren Ergebnis in HTML-Auswahllisten umgewandelt wird, bei der Dateneingabe berücksichtigt werden. In diesem Fall wird statt eines statischen HTML-Dokuments ein um SQL-Anweisungen ergänztes benötigt. Für eine automatische Umsetzung ist dazu neben einer Kennzeichnung von Referenzen auch die Markierung der Attribute mit „Klartext-Beschreibung“ (z. B. der Name) eines Tupels vorzusehen, um dem Benutzer nicht nur IDs zur Auswahl anbieten zu müssen.

z.T. automatisch und durch Laien, Experte erforderlich für komplexe Abb., da manuelle Nachbesserung/Konstruktion erforderlich, insgesamt jedoch deutliche Beschleunigung bei des Erzeugungs-/Integrationsprozesses

weitere Probleme für Bearbeitung gespeicherter Daten, hier evtl. Füllen von Formular möglich,
dieses Vorgehen (Typkonstruktion mit nur für bestimmte Typen sinnvoll

Die generierten HTML-Schablonen für die Datenein- und -ausgabe sind nur für einfache alphanumerische Datentypen geeignet, für komplexe Datentypen sind andere Verfahren, z. B. Upload-Mechanismus für die Eingabe und externe *Viewer* für die Ausgabe, vorzusehen. Möchte man komplexe Datentypen, wie z. B. Konstruktionsdaten, auf dem Client bearbeiten, ist das hier realisierte Verarbeitungsmodell nicht passend. In diesem Fall wäre das Checkout/Checkin-Konzept vorzuziehen, das in einem späteren Schritt auch für das IS bereitgestellt werden

soll. Bei der Typkonstruktion werden zum jetzigen Zeitpunkt auch Beziehungen der Datentypen untereinander nicht berücksichtigt, für die Zukunft ist jedoch eine Art „Referenzkonstruktor“ zur Bildung von Beziehungen vorgesehen.

Referenzen