Aktuelles Schlagwort “Semi-strukturierte Daten”

François Bry, Michael Kraus, Dan Olteanu und Sebastian Schaffert
Institut für Informatik, Universität München, Oettingenstraße 67,
80538 München, http://www.pms.informatik.uni-muenchen.de

13. Juni 2001

1 Vorwort

Wie die meisten Forschungsgebiete der Informatik wurde die Datenbankforschung zuerst vorwiegend vom Paradigma einer zentralen Verwaltung geprägt. Diese Sicht wurde zunehmend in Frage gestellt. Der Ansatz der “Semi-strukturierten Daten” seit Mitte der 90er Jahre ist ein weiterer Schritt in diese Richtung. Ausgangspunkt war die Verwaltung von Inhalten im dezentralen WWW und die Datenmodellierungssprache XML [8]. In den Ansatz “Semi-strukturierte Daten” führt das Buch [1] ein.

Ein traditionelles Datenbanksystem setzt voraus, dass die gespeicherten Daten gemäß einem im voraus festgelegten Datenschema strukturiert sind. Schemata erleichtern die Datenspeicherung und dienen der Anfrageauswertung. Im dezentral verwalteten WWW sind Schemata oft zu restriktiv. In vielen Bereichen wie in der Bioinformatik werden Daten in heterogenen Formaten zwischen Datenbanken oder sonstigen Anwendungen ausgetauscht, denen kein einheitliches Schema zu Grunde liegt, weswegen solche Daten zunächst “unstrukturiert” danach “semi-strukturiert” genannt wurden [4]. Oft haben zudem die Daten eine Struktur, die mit den flachen Tupeln des relationalen Datenmodells nur unzutreffend wiedergegeben werden kann. Auch das Objektmodell ist oft ungeeignet: Zwar kann man damit auch “tiefe” Strukturen repräsentieren, allerdings keine unregelmäßige Strukturen mit fehlenden oder wiederholten Komponenten. Fehlt das Schema, so muss zudem die Bedeutung der Struktur in den Datensätzen selbst wiedergegeben werden. Man spricht von “strukturtragenden” oder von “selbsterklärenden” Daten.

2 Ausgangspunkt XML

Die Datenmodellierungssprache XML [8] eignet sich für die oben genannten Fälle: mit XML können beliebig komplexe Datensätze spezifiziert werden; XML läßt Ausnahmen zu (in XML können alternative Strukturen spezifiziert werden und die spezifizierte Struktur darf sogar missachtet werden); XML-Datensätze, im XML-Jargon Dokumente genannt, sind strukturtragend.

Z.B. kann eine Adresskartei wie folgt in XML spezifiziert werden:

```
<Adresskartei>
  <Adresse>
      <Name>Bartl Bastscho</Name>
      <Einrichtung>Universität München</Einrichtung>
      <EMail>bartl@bastscho.net</EMail>
   </Adresse>
   <Adresse>
      <Name>Susi Schlau</Name>
      <Einrichtung>Universität München</Einrichtung>
      <Abteilung>Fakultät für Mathematik und Informatik</Abteilung>
      <Kontakt>
        <EMail>susi@bastscho.net</EMail>
        <Url>http://www.bastscho.net/freunde/susi</Url>
      </Kontakt>
   </Adresse>
</Adresskartei>
```

Der zweite Datensatz hat eine reichere Struktur als der erste; in den beiden Datensätzen befindet sich die E-Mailadresse nicht auf der selben Tiefe.

In einer Anfrage an eine solche Adresskartei sollen nicht nur die Texte, Zahlen, oder sonstige Daten ermittelt werden, die als Inhalte der innersten Elemente vorkommen, sondern auch die Elementnamen, wie “Name”, “Einrichtung” oder “Kontakt”. Schema- und Objektdaten sollen also in Anfragen nicht wie in traditionellen Anfragesprachen wie SQL unterschiedlich behandelt werden.

Wird in einer solchen Adresskartei nach einer E-Mailadresse für eine bestimmte Person gesucht, dann kann der Elementname “EMail” (oder irgendein Synonym dieses Elementnamens, welche eine sogennante “Ontologie” liefern könnte) an beliebiger Tiefe gesucht werden. Eine Anfragesprache muß also über Sprachkonstrukte verfügen, die es ermöglichen, eine unbekannte Struktur oder eine unbestimmte Tiefe auszudrücken.

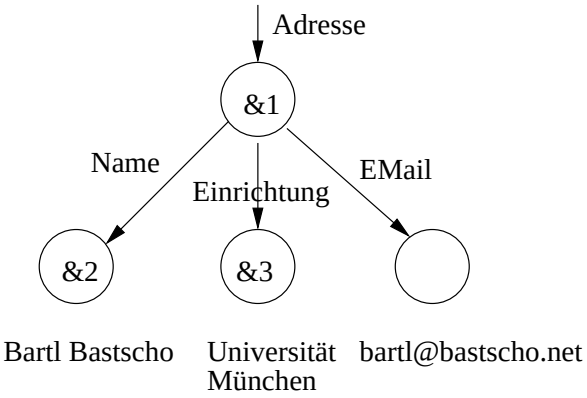
3 Datenmodelle für semi-strukturierte Daten

Für semi-strukturierte Daten wurden einige Modelle vorgeschlagen, die nur unwesentlich voneinander abweichen.

Im Object Exchange Model [1], kurz OEM, werden Datensätze Objekte genannt. Wie Objekte von Objektdatenbanken kann (muß aber nicht) ein OEM-Objekt eine sogenannte Objektidentität besitzen, so dass zwei OEM-Objekte, die den selben Wert haben, unterschiedlich sein können. Der Wert eines atomaren OEM-Objekts hat einen Typ wie etwa “integer”, “string” oder “image”. Ein zusammengesetztes OEM-Objekt besteht aus Attributen, die Werte besitzen, die wiederum OEM-Objekte sind. Die Attribute eines zusammengesetzten OEM-Objekts bilden entweder eine Menge oder eine Sequenz. OEM bietet eine weitreichende Typanpassung, um Unregelmäßigkeiten Rechnung zu tragen.

Ein OEM-Objekt kann als ein gerichteter Multigraph angesehen werden, dessen Knoten eindeutige Objektbezeichner sind und dessen Kanten die Attribute darstellen. Ein OEM-Objekt besitzt eine Wurzel, d.h. einen ausgezeichneten Knoten, von dem aus alle Knoten des OEM-Objektes erreichbar sind.

Der erste Adresskarteieintrag kann also wie folgt dargestellt werden, wobei "&n" Objektbezeichner sind:



Die Kanten stellen sowohl die Beziehung eines Elements zu einem Kindelement als auch etwaige Verweise wie (hypertext oder nicht-hypertext-) Links dar. Ein OEM-Objekt entspricht nicht notwendigerweise einem Baum, d.h. OEM läßt zyklische Objekte zu. OEM-Objekte können HTML-Links oder einfache XML-Links darstellen, jedoch nicht die erweiterten XML-Links von XLink [8].

Das Datenmodell, das der Anfragesprache UnQL [1] zu Grunde liegt, ist OEM ähnlich. Er ist aber "wertbasiert", d.h. dass es keine Objektidentität kennt. Mit diesem Datenmodell ist die Verwendung der "Bisimulation" für semistrukturierte Daten eingeführt worden [1], die sich für die Anfrageauswertung als wichtig erwiesen hat. Eine Simulation eines Multigraphen G_1 in einen Multigraphen G_2 ist eine binäre Relation S zwischen den Knoten von G_1 und G_2, die die Kanten von G_1 auf Kanten von G_2 abbildet. Eine Simulation S von G_1 in G_2 ist eine Bisimulation, falls S^{-1} eine Simulation von G_2 in G_1 ist. Für UnQL sind zwei semi-strukturierte Objekte identisch, wenn sie bisimilar sind.

Das Datenmodell YAT [1] bietet neben ähnlichen Merkmalen wie OEM und dem Datenmodell der Anfragesprache UnQL die "Modellinstanziierung". Damit können unter gewissen Umständen die Typen einer Datenmodellierung den Typen einer anderen Datenmodellierung angepaßt werden. So können unterschiedliche Modellierungen von Anwendungen verglichen und Ergebnisse von Anfragen strukturiert werden.

4 Vergleich mit den relationalen und Objektmodellen

Eine Relation oder Tabelle kann als ein semi-strukturiertes Objekt dargestellt werden, dessen Attribute die Tupel sind. Ein Tupel kann ebenfalls als ein semistrukturiertes Objekt dargestellt werden, dessen Attribute die Attributswerte des Tupels sind.

Diese Darstellung einer Relation ist ziemlich natürlich. Sie ist aber auch etwas trügerisch. Zum einen kann eine Relation auf vielen andere Weisen als semi-strukturiertes Objekt dargestellt werden. Zum anderen kann bei einer solchen Repräsentation ein Teil der Semantik verloren gehen, weil im Gegensatz

zu einer relationalen Anfragesprache wie SQL die Anfragesprachen für semi-strukturierte Daten alle Knoten gleich behandeln. So muss der Benutzer einer Anfragesprache für semi-strukturierte Daten die Bedeutung der Verknüpfungen beachten.

Ein Datenmodell für semi-strukturierte Daten kann als Vereinfachung eines Objektmodells angesehen werden, weil es in der Regel weder Klassen noch Vererbung anbietet. Der Verzicht auf Klassen und Vererbung ist jedoch fraglich. Die Modellinstanziierung von YAT [1] entspricht der Vererbung.

5 Schemaextraktion

Traditionell stützen sich Anfrageauswertung und -optimierung auf Schemata. Um Anfragen über semi-strukturierten Daten auszuwerten, bietet es sich also an, zuerst aus den vorhandenen Daten ein Schema zu extrahieren.

Viele WWW-Informationsserver liefern HTML-Seiten, die dynamisch aus einer Datenbank erzeugt werden. Es sind Methoden entwickelt worden, um aus solchen Seiten ein Schema zu erkennen, das eventuell nicht identisch mit dem Datenbankschema ist. Man spricht von “web site wrapping” und von “Schemaextraktion”. Das extrahierte Schema wird manchmal “data guide” genannt. Eine Schemaextraktion ist eine Art Klassifikation von vorhandenen Datensätzen nach ihren Strukturen. Die meisten Methoden sind halbautomatisch und interaktiv und stützen sich auf Heuristiken.

In [7] wird ein auf Deduktion und Datalog beruhender Ansatz zur Schemaextraktion dargestellt, der zuläßt, dass der selbe Datensatz mehreren Klassen zugeordnet wird und Approximationsschemata, denen nicht alle Datensätze entsprechen, liefert. Eine Metrik wird vorgeschlagen, womit die Qualität eines Approximationsschemas gemessen werden kann.

6 Anfragesprachen

Eine Anfragesprache ist wünschenswert zur Erstellung von dynamischen WWW-Seiten, die Inhalte teilen statt kopieren, um Sichten (”views”) zu erzeugen und um die Suche nach Inhalten zu erleichtern. Wünschenswerte Eigenschaften von Anfragesprachen für semi-strukturierte Daten wurden in [6] ausgearbeitet.

Viele Anfragesprachen für semi-strukturierte Daten oder “für das WWW” sind entwickelt worden – u.a. Lorel [1] und XQuery [8]. Einige sind von SQL und OQL inspiriert, andere sind im Zusammenhang mit XML entstanden. Vergleiche bieten [5, 2, 3] an.

Die meisten Anfragesprachen verwenden Variablen, um Knoten – einige sowohl für Elementinhalte als auch für Elementnamen – zu bezeichnen. Reguläre Ausdrücke werden von vielen Anfragesprachen verwendet, um die Navigation im befragten Datensatz auszudrücken.

Mit dem Sternoperator kann z.B. ein Element an beliebiger Tiefe gefunden werden, mit dem Optionsoperator können Unregelmäßigkeiten der Datensatzstruktur berücksichtigt werden. Mit einer “wildcard” kann eine Navigation entlang von unvollständig spezifizierten Pfaden definiert werden. Z.B. können

im Stil von XML-QL wie folgt E-Mailadressen an beliebiger Tiefe (ausgedrückt durch die Wildcard $ gefolgt vom Stern-Operator) gefunden werden, wobei $name und $email Variablen sind:

```
where     <$*>
             <Nachname>$name</Nachname>
             <EMail>$email</EMail>
          </>
in        "www.yellowpages.net/adresses.xml"
construct <Antwort>
             <Person>
                <Name>$name</Name>
                <Mail>$email</Mail>
             </Person>
          </Antwort>
```

Die meisten Anfragesprachen ermöglichen, die Antwort beliebig zu strukturieren.

Einige Prototypen von Systemen zur "Verwaltung von Web sites", d.h. zur Verwaltung von Inhalten, sind entwickelt worden, womit die ersten Anfragesprachen für semi-strukturierte Daten entwickelt und getestet worden sind.

Literatur

[1] *Data on the Web - From Relations to Semistructured Data and XML.* Morgan Kaufmann, 2000.

[2] A. Bonifati and S. Ceri. Comparative analysis of five xml query languages. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2000.

[3] A. Bonifati and D. Lee. Technical survey of xml schema and query languages. In *VLDB*, 2001.

[4] Peter Buneman, Susan Davidson, and Dan Suciu. Programming constructs for unstructured data. In *Proceeding of the Fifth International Workshop on Database Programming Languages*, 1995.

[5] Mary Fernandez, Jerome Simeon, and Philip Wadler. Xml query languages: Experiences and exemplars. http://cm.bell-labs.com/cm/cs/who/wadler/topics/xml.html.

[6] David Maier. Database desiderata for an xml query language. In *The Query Languages Workshop (QL'98).* http://www.w3.org/TandS/QL/QL98/pp/maier.html, 1998.

[7] S. Nestorov, Serge Abiteboul, and Rajeev Motwan. Extracting schema from semistructured data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1998.

[8] World Wide Web Consortium (W3C). *Die Spezifikation von XML und XML-bezogenen Formalismen wie XSLT und XML Schema.* http://www.w3.org/.