OttoQL

Klaus Benecke
benecke@iws.cs.uni-magdeburg.de
Martin Schnabel

IWS/FIN, Otto-von-Guericke-Universität Magdeburg
Postfach 4120
39016 Magdeburg, Germany, Sachsen-Anhalt

1 Einführung

OttoQL ist eine Anfragesprache für strukturierte Daten. Hauptaugenmerk wird auf Einsteigerfreundlichkeit der Sprache gelegt. OttoQL im aktuellen Entwicklungsstand erlaubt es beliebige XML-Dateien, die über eine Dokumenttypdefinition (DTD) verfügen zu verarbeiten. Mit wenigen aber leistungsstarken Operationen können Daten verarbeitet werden. Die Operationen arbeiten mengenorientiert, die Daten werden also in ihrer Gesamtheit betrachtet - Iteration durch einzelne Elemente ist nicht notwendig. Für Einsteiger und Programmierer funktionaler Sprachen ist diese Vorgehensweise intuitiv. Programmierer imperativer Sprachen fühlen sich mit diesem Konzept anfangs unwohl, erkennen dann aber dessen Eleganz. Der große Vorteil ist, dass man nicht beschreibt, *wie* die Daten umgewandelt werden sollen, sondern in *was* die Daten umgewandelt werden sollen. Um das *Wie* kümmert sich OttoQL.

Die drei Grundoperationen sind die Erweiterung, die Selektion und die Umstrukturierung. Durch nacheinander Anwendung dieser Operationen kommt der Benutzer Schritt für Schritt zum Ziel. Die meisten OttoQL-Programme kommen mit diesen drei Grundoperationen aus. Zusätzlich verfügt die Sprache aber auch über Variablen, Schleifen, bedingte Ausführung, Funktionen und andere gewohnte Bestandteile.

Die Einsatzmöglichkeiten von OttoQL sind vielseitig. Fast überall wo Daten ausgewertet werden sollen, kann OttoQL gute Dienste leisten. Die Daten müssen nur als XML-Dateien vorliegen. Eine Anbindung an relationale Datenbanken ist wünschenswert, wurde aber aus Ressourcenmangel noch nicht umgesetzt. Um größere Datenmengen handhaben zu können, implementiert OttoQL ein einges Datenformat (H2O), das geeignete „XML-Sätze“ nach dem TID-Konzept adressiert.

Außerdem soll es einen visuellen Editor für OttoQL-Programme geben, der die Operationen grafisch darstellen und per Drag’n’Drop erzeugen kann.

2 Die drei Grundoperationen - ganz kurz

OttoQL sieht XML-Dateien als strukturierte Daten bzw. Tabellen an. Eine flache Tabelle ist eine gewöhnliche Tabelle mit Tabellenkopf und Daten in den Zellen (Abbildung 1). Die strukturierte Tabelle kann auch Tabellen (auch wiederum strukturiert) in den Zellen enthalten (Abbildung 2). Die Beschreibung dieser verschachtelten Tabellenköpfe erfolgt durch ein Schema oder eine Dokumenttypdefinition (DTD). Die Operationen können auf

A	B	C
1	2	3
4	5	6

Abbildung 1: flache Tabelle

Artikel	Preis	M(Lager, Anzahl)	
Monitor	124.98	Lager	Anzahl
		Braunschweig	7
		Magdeburg	11
Tastatur	28.95	Lager	Anzahl
		Berlin	23
		Magdeburg	42

Abbildung 2: strukturierte Tabelle

diese strukturierten Tabellen angewendet werden, um die Tabellen zu erweitern (ext), zu filtern (mit/sans) oder umzustrukturieren (gib).

- Die **Erweiterung (ext)** fügt Spalten hinzu. Als Argumente werden ein (optionaler) Spaltenname und ein Ausdruck benötigt. Ob die äußere oder eine weiter innen liegende Tabelle erweitert wird, erkennt *ext* anhand der Variablen im Ausdruck, die sich auf Spalten des Ausgangsdokumentes beziehen.
- Die **Selektion (mit/sans)** löscht Zeilen abhängig von einer Bedingung.
- Die **Umstrukturierung (gib)** (vgl. [1] und [2]) wandelt die Struktur einer Tabelle. Sie benötigt dazu keine Regeln, sondern allein die Angabe eines neuen Schemas bzw. einer neuen DTD. Sie ist damit eine sehr mächtige Operation. Zusätzlich kann die Umstrukturierung neue Spalten erzeugen indem Daten aus vorhandenen Spalten aggregiert werden. Spalten werden eliminiert, indem man sie im Zielschema weglässt.

3 Beispiele

Das folgende OttoQL-Programm demonstriert die Erweiterung:

```
ext doc("inventar.xml")
ext Wert := Preis * Anzahl
ext Gesamtzahl := sum(L(Anzahl)) at Preis
```

Das erste *ext* holt die Daten aus Abbildung 2. Nun wird die Spalte *Wert* aus dem Produkt von *Preis* und *Anzahl* berechnet. Die Spalte wird den inneren Tabellen hinzugefügt, weil die Variable *Anzahl* im Ausdruck aus den inneren Tabellen kommt. Die Berechnung der *Gesamtzahl* im letzten Schritt erzeugt eine neue Spalte in der äußeren Tabelle. Im Ausdruck steht zwar *Anzahl* aus den inneren Tabellen, die Listenbildung **L**(..) führt aber aus ihnen heraus. Das Ergebnis ist in Abbildung 3 in Tabellenform zu sehen.

Artikel	Preis	Gesamtzahl	M(Lager, Anzahl, Wert)
Monitor	124.98	18	Lager: Braunschweig, Anzahl: 7, Wert: 874.86 Lager: Magdeburg, Anzahl: 11, Wert: 1374.78
Tastatur	28.95	65	Lager: Berlin, Anzahl: 23, Wert: 665.85 Lager: Magdeburg, Anzahl: 42, Wert: 1215.9

Abbildung 3: Ausgabe

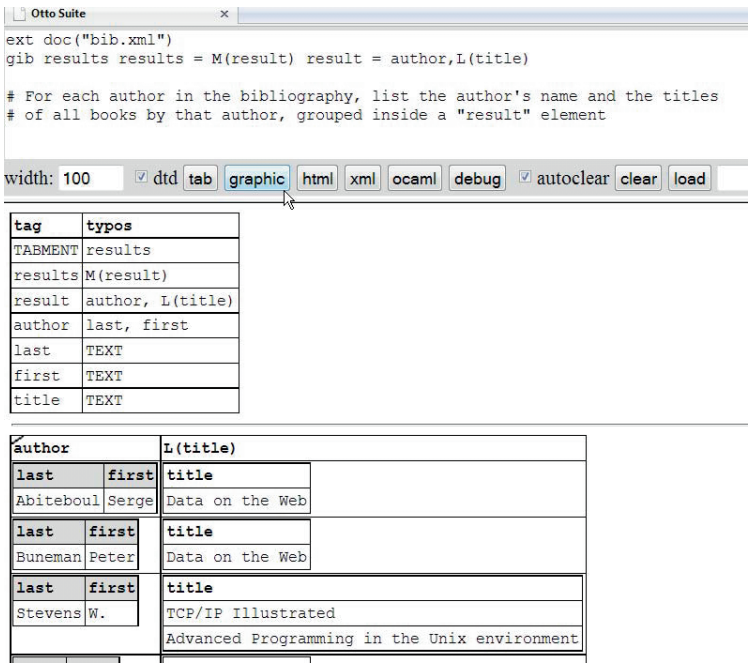

Abbildung 4: Screenshot der OttoQL-Oberfläche

Im Folgenden wird ein XML Query Use Case [3, 1.1.9.4 Q4] als OttoQL-Programm gelöst. Das XQuery-Programm erstreckt sich über 21 Zeilen und ist durch zwei verschachtelte FLWOR-Konstrukte gekennzeichnet. Explizit muss der Benutzer über einen XPath-Ausdruck angeben durch welche Struktur iteriert werden soll und das Ausgabedokument schablonenartig konstruieren.

Im Gegensatz dazu ist das OttoQL-Programm äußerst kurz und begnügt sich mit zwei Zeilen. In Abbildung 4 sieht man wie es in der OttoQL-Oberfläche ausgeführt wurde. Im ersten Schritt wird durch *ext* das Ausgangsdokument gelesen und im zweiten Schritt die Umstrukturierung *gib* ausgeführt. Allein durch die Angabe der Ziel-DTD erzeugt OttoQL die gleiche Ausgabe wie das XQuery-Programm. Für die DTD verwendet OttoQL eine der XML-DTD leicht erweiterte Beschreibung mit vereinfachter Syntax. Die Ziel-DTD $\left\{ \begin{array}{l} results \\ results = M(result) \\ result = author, L(title) \end{array} \right\}$ beschreibt die gewünschte Zielstruktur. Das Zieldokument ist vom Typ *results*, wobei *results* eine **M**enge von *result* ist. Menge bedeutet, dass die Elemente sortiert (in diesem Fall alphabetisch) und Duplikate eliminiert werden sollen. *result* ist ein Tupel aus *author* und *L(title)*. *L(title)* ist eine **L**iste und behält damit seine Ordnung und Elemente.

Die Ausgangs-DTD ist hier bewusst nicht angegeben. Genaue Kenntnis über sie ist für den Anwender nicht notwendig. Er muss nur wissen, welche Namen (also XML-Tags) die Daten haben. Das Programm würde sogar auf unterschiedlich strukturierten Ausgangsdaten funktionieren. In XQuery ließe sich so ein allgemeines Programm gar nicht oder nur sehr schwer realisieren lassen.

In Abbildung 4 sieht man unten die Ausgabe in Tabellenform inklusive der neuen DTD. Genau so kann man das Ergebnis auch als XML-Datei mit XML-DTD ausgeben und abspeichern.

4 Weiteres

Die Beispiele können nur einen kleinen Einblick in die Leistungsfähigkeit von OttoQL bieten. Sie kann als domänenspezifische Sprache angesehen werden, dessen Einsatzmöglichkeiten weit über das übliche Maß hinaus gehen. Niemand würde in XQuery weitreichende Berechnungen auf den Daten anfertigen wollen, sondern zu einem weiteren Spezialprogramm greifen. OttoQL nimmt für sich in Anspruch auch hier Lösungen zu bieten.

OttoQL wurde in Ocaml entwickelt, besitzt eine Weboberfläche und kann unter http://otto.cs.uni-magdeburg.de/otto/web/ ausprobiert werden.

Literatur

[1] K. Benecke: *A Powerful Tool for Object-Oriented Manipulation*, in Proc. IFIP TC2/WG 2.6 Working Conference on Object Oriented Database: Analysis, Design & Construction, Windermere, UK, pp. 95-122, North Holland 1991

[2] K. Benecke: *Strukturierte Tabellen - Ein neues Paradigma für Datenbanken und Programmiersprachen*, Deutscher Universitätsverlag, Wiesbaden 1998

[3] ed.: D. Chamberlain et al., *XML Query Use Cases*, W3C Working Draft 23 March 2007, http://www.w3.org/TR/xquery-use-cases/#xmp-queries-results-q4