

Sequenzcursor-basierte Verarbeitung von XML-Werten in SQL:2007-Anfrageergebnissen



seit 1558

Thomas Müller

Lehrstuhl für Datenbanken und Informationssysteme

Friedrich-Schiller-Universität Jena

Gliederung

- Relevanz der SQL-basierten Verarbeitung
- XML-Werte und die SQL-Norm
- XML-Werte und heutige RDBMS-Produkte
- Ein SQL:2007-Beispielszenario
- Sequenzcursor-basiertes Modell
- Abgrenzung gegen existierende Ansätze
- Zusammenfassung und Ausblick

Neue Anforderungen an DBSe

- zunehmende Verbreitung von XML
 - wachsendes Bedürfnis, XML-Dokumente effizient speichern, anfragen und verarbeiten zu können
 - Forderung an Datenbanksysteme, XML zu unterstützen
 - zwei Wege der Industrie:
 - native XML-Datenbanksysteme
 - Erweiterung relationaler Datenbanksysteme

Native XML-Datenbanksysteme

- vollständig auf Speicherung und Verarbeitung von XML-Daten ausgerichtet
 - Zugriff über XML-orientierte Schnittstellen
 - i. d. R. keine direkte SQL-Unterstützung
- prominentester Vertreter:
Tamino XML Server der Software AG

Bisher ausgebliebener Markterfolg der nativen XML-Datenbanksysteme

- mögliche Gründe
 - Systeme ungeeignet für gemeinsame Verwaltung bzw. integrierte Auswertbarkeit von XML-Daten einerseits und „herkömmlichen“ SQL-Daten andererseits
 - „natürliche Scheu“ der Anwender davor, Datenhaltung komplett auf völlig anderes Datenmodell (und Datenbanksystem) umzustellen
- Fazit
 - XML-Erweiterung relationaler Datenbanksysteme besitzt derzeit höhere Praxisrelevanz

SQL und XML

- Bedarf an einem SQL-basierten Datenzugriff auch in Zukunft wichtig
- Verarbeitung von XML-Werten ebenfalls wichtig
 - XML-Werte sind in SQL zu berücksichtigen
 - wurde von SQL-Normungsgremien erkannt
 - SQL:2003 unterstützt (erstmalig) XML-Werte
 - SQL:2007 baut XML-Unterstützung weiter aus

XML-Werte vor SQL:2003

- keinerlei XML-Unterstützung durch die SQL-Norm
- zur Speicherung von XML-Dokumenten in SQL-Tabellen musste stets auf eine nachteilbehaftete Speichertechnik zurückgegriffen werden
 - textbasierte Speicherung: XML-Dokument wird als einfache Zeichenkette in LOB-Spalte abgelegt
 - strukturbasierte Speicherung: aus zugrunde liegendem Schema wird DB-Schema abgeleitet, XML-Dokumente werden dann auf Tupel geeigneter Tabellen abgebildet
 - ...

XML-Werte in SQL:2003

- separater Normteil SQL/XML
- neuer SQL-Basisdatentyp „XML“
 - Tabellen mit Spalten des Typs XML
 - Nullwerte
 - „erweiterte“ XML-Dokumente
(nicht notwendigerweise genau ein Wurzelement)
 - Rückgriff auf Speicherungstechniken nicht mehr nötig (XML ist vollwertiger Datentyp)

XML-Werte in SQL:2007

- XML-Unterstützung wird weiter ausgebaut
- SQL-Basisdatentyp „XML“ basiert jetzt auf dem XQuery-Datenmodell
 - In Tabellen sind als XML-Werte beliebige XQuery-Sequenzen zulässig!
- Integration von XQuery in SQL
 - XQuery-Anfragen können in SQL-Anfragen eingebettet werden! (mittels SQL-Funktion XMLQUERY)

Validierung von XQuery-Sequenzen

- Mittels der SQL:2007-Funktion `XMLVALIDATE` können (aus keinem, einem oder sogar mehreren Einträgen bestehende) XQuery-Sequenzen validiert werden (momentan nur mit D-, E-, C- und P-Knoten auf Wurzelebene)
- Bei der Validierung wird die Sequenz mit Typ-Informationen angereichert
- Auch validierte XQuery-Sequenzen (inkl. der durch die Validierung entstandenen/geänderten Typinformationen) sind als Attributwerte zulässig!

XML-Werte und RDBMS-Produkte

- Die drei bedeutendsten relationalen DBMS-Produkte (Oracle, MS SQL Server und DB2) unterstützen heutzutage XML
 - Built-in-Datentyp zur Speicherung wohlgeformter XML-Dokumente in allen drei Produkten
 - Oracle 10g: „XMLType“
 - MS SQL Server 2005: „XML“
 - DB2 UDB 9.1: „XML“
- (aber: nur wohlgeformte XML-Dokumente, keine beliebigen XQuery-Sequenzen)

DB2 V8 vs. DB2 V9

- kein Built-in-Datentyp für XML-Werte in DB2 V8
- XML Extender als „Notbehelf“
- drei benutzerdefinierte XML-Datentypen:
XMLVARCHAR, XMLCLOB und XMLFILE
- in DB2 V9 außer Built-in-Datentyp XML auch leistungsfähige XML-relevante Funktionen wie XMLQUERY, XMLTABLE und XMLVALIDATE
- rasante Entwicklung unterstreicht Relevanz der SQL-basierten Verarbeitung von XML-Werten

Mangelnde SQL:2007-Unterstützung durch heutige RDBMS-Produkte

- wesentlicher Schritt bei Weiterentwicklung von SQL:2003 zu SQL:2007 ist Wechsel vom XML-Info-Set-basierten Datenmodell zum XQuery-Datenmodell
- (Eine) wesentliche Neuerung von SQL:2007 gegenüber SQL:2003 besteht also darin, dass als XML-Werte anstatt einfacher XML-Dokumente auch komplette XQuery-Sequenzen zulässig sein werden
- Aber: Gerade diese entscheidende Neuerung wird (bisher) von keinem RDBMS-Produkt reflektiert!

SQL:2007-Beispielszenario

- zurzeit keine SQL:2007-fähigen RDBMS-Produkte verfügbar
 - derzeit noch kein „echtes“ SQL:2007-Anwendungsszenario in der Praxis
 - nutzen eigenes realitätsnahes auf SQL:2007 abgestimmtes Beispielszenario als Grundlage für weitere Erläuterungen
- (fiktive) Anwendung aus dem praxisrelevanten Gebiet der Kundenkartenverwaltung

Beispiel-Szenario

„Kundenkartenverwaltung“

- (fiktives) Unternehmen, welches Kundenkarten ausgibt und verwaltet
- Kundenkarten u. a. einsetzbar bei Einkäufen, Autoanmietungen, Flügen, Friseurbesuchen und Hotelübernachtungen, um Rabatte bzw. Sonderkonditionen zu erhalten
- Kundenkartenverwaltung arbeitet mit verschiedenen Partnerunternehmen zusammen (Supermarktketten, Autovermietern, Hotels usw.)

KKV: Motivation

- Anreiz für die Kunden:
 - Rabatte und Bonusangebote
- Anreiz für die Partnerunternehmen:
 - „Anlocken“ und Binden von Kunden
 - gezielter Verkauf bestimmter Produkte
- Anreiz für die Kundenkartenverwaltung:
 - Sammeln von detaillierten Informationen über das Einkaufsverhalten


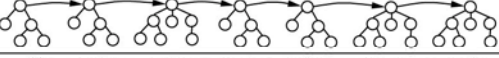
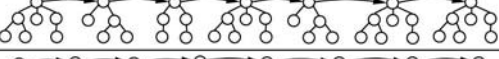

KKV: Bonusangebote

- **Bonusangebote** als „Anreiz“ für Einkäufe
- Beispiel: „Wer für mindestens €3,00 Spinatbrei kauft – und seine Kundenkarte vorlegt – bekommt beim nächsten Einkauf zwei Päckchen Waschmittel zum Preis von einem.“
- Bonusangebote sind **zeitlich befristet** und besitzen „**einlösbaren Gegenwert**“ (z. B. €0,57 bei einem Folgeeinkauf im gleichen Geschäft bzw. €0,32 bei einem Folgeeinkauf in einem anderen Geschäft)

Nutzung der Bonusangebote

- Kunde kann an Terminal im Geschäft auf seine bisher erworbenen Bonusangebote zugreifen
- Kunde kann gezielt ein oder mehrere Bonusangebote auswählen und deren aufsummierten Gegenwert als Gutschein drucken
(damit verfallen die ausgewählten Bonusangebote)
- nicht einfach „simpler Punktestand“, sondern für jedes Bonusangebot separat entscheidbar, ob es in Anspruch genommen wird oder ob stattdessen sein einlösbarer Gegenwert genutzt wird

KKV: Datenhaltung

Kukabo	KdNr	Kategorie	Bonus	letztmals_aktiv
	4711	Dienstleistung		30.08.2006
	4711	Einkauf		14.09.2006
	4711	Anmietung		05.07.2006
	4712	Dienstleistung		02.08.2006

- Verwaltung aller gesammelten Daten mit Hilfe eines SQL:2007-fähigen relationalen Datenbanksystems
- Bonusdaten der einzelnen Kunden sind getrennt nach der Kategorie in Form „echter“ XQuery-Sequenzen in Spalte des SQL:2007-Datentyps XML gespeichert

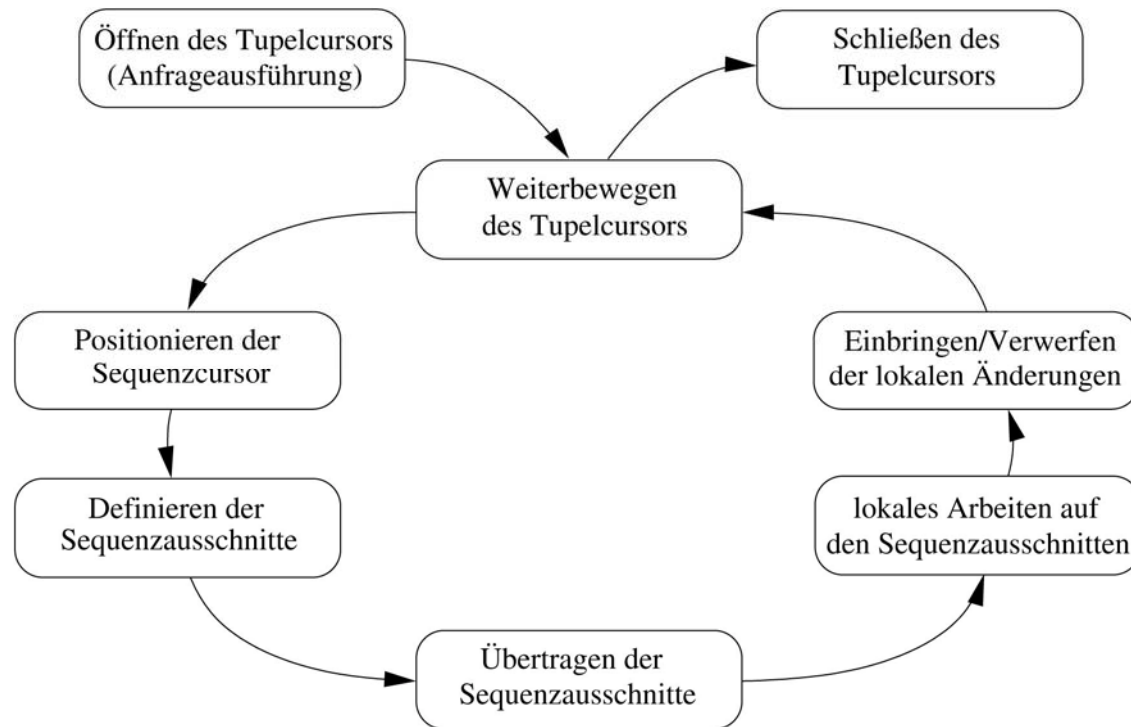
Aufwertung der Bonusangebote

- wird in Aussicht gestellt, um Kunden z. B. zur Teilnahme an telefonischer Umfrage zu motivieren
 - wer teilnimmt, erhält „Dankeschön-Code“
 - aufwertbar sind jeweils die letzten 5 Bonusangebote derjenigen Kategorien, in denen der Kunde innerhalb des letzten Monats aktiv war
 - für jedes aufwertbare Bonusangebot jeweils entscheidbar: Erhöhung des Gegenwerts um 50 Cent oder Verlängerung der Gültigkeit um eine Woche
 - wegen Wahlmöglichkeit des Kunden, keine automatische Aufwertung möglich
- entspr. Bonusangebote müssen in AP übertragen werden

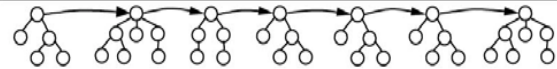
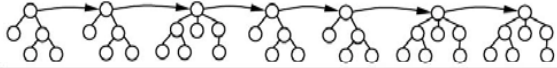
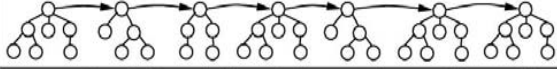
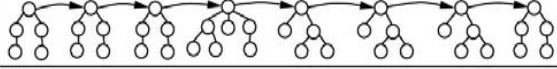
Sequenzcursor-basiertes Verarbeitungsmodell

- SQL:2007-Anfrageergebnis, welches u. a. komplette XQuery-Sequenzen als Attributwerte enthält
- Auf Anfrageergebnis wird „klassisch“ mittels SQL-Tupelcursor navigiert
- Mit **Sequenzcursor** wird in die XQuery-Sequenzen des aktuellen Ergebnistupels eingetaucht
- Bezugnehmend auf die Sequenzcursor werden **Ausschnitte** der im akt. Ergebnistupel enthaltenen Sequenzen definiert
- Sequenzausschnitte werden ins Anwendungsprogramm übertragen und dort lokal verarbeitet
- Lokal an den Sequenzausschnitten vorgenommene Änderungen können in die DB eingebracht werden

Sequenzcursor-basierter Verarbeitungsablauf



Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

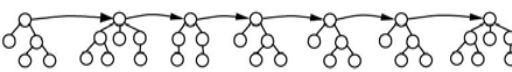
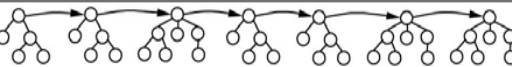
Kukabo	KdNr	Kategorie	Bonus	letztmals_aktiv
	4711	Dienstleistung		30.08.2006
	4711	Einkauf		14.09.2006
	4711	Anmietung		05.07.2006
	4712	Dienstleistung		02.08.2006

- Ausgangspunkt ist Kunden-Kategorie-Bonus-Tabelle „Kukabo“
- Kunde mit KdNr 4711 habe an Kundenbefragung teilgenommen
- Aufwertung seiner jeweils letzten 5 Bonusangebote derjenigen Kategorien, in denen er innerhalb des letzten Monats aktiv war

Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

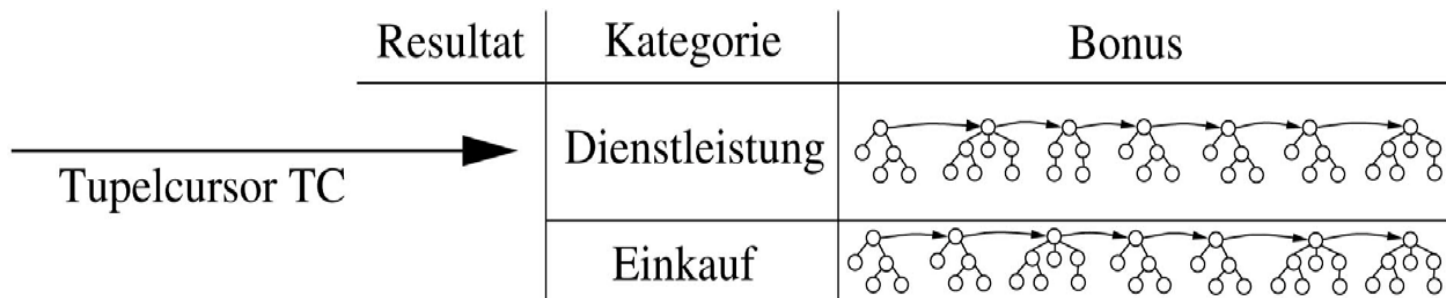
- Mittels SQL:2007-Anfrage werden diejenigen Bonusangebotesequenzen ausgewählt, die die aufwertbaren Bonusangebote enthalten

```
SELECT Kategorie, Bonus
FROM Kukabo
WHERE KdNr = 4711
      AND letzmals_aktiv >= CURRENT DATE - 1 MONTH;
```

Resultat	Kategorie	Bonus
	Dienstleistung	
	Einkauf	

Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

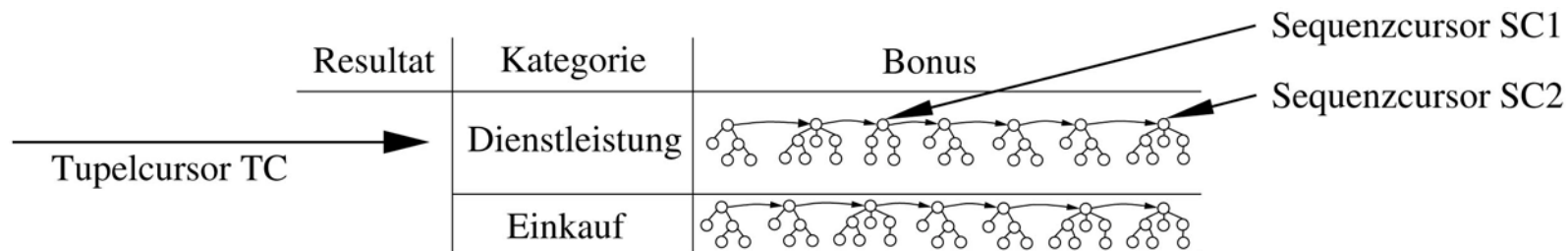
- In Embedded-SQL-Programm wird Anfrage mittels SQL-DECLARE-CURSOR-Anweisung an einen Tupelcursor „TC“ gebunden
- Anfrageausführung erfolgt beim Öffnen des Tupelcursors (OPEN TC)
- Mittels SQL-FETCH-Anweisung wird geöffneter Tupelcursor anschließend auf erstes zu verarbeitendes Ergebnistupel positioniert



Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

- Nutzen im Beispiel die zwei Sequenzcursor „SC1“ und „SC2“, um den ins Anwendungsprogramm zu übertragenden Sequenzausschnitt zu definieren
- Positionieren SC2 auf Wurzelknoten des letzten Sequenzeintrags
- Positionieren SC1 auf Wurzelknoten des fünftletzten Eintrags

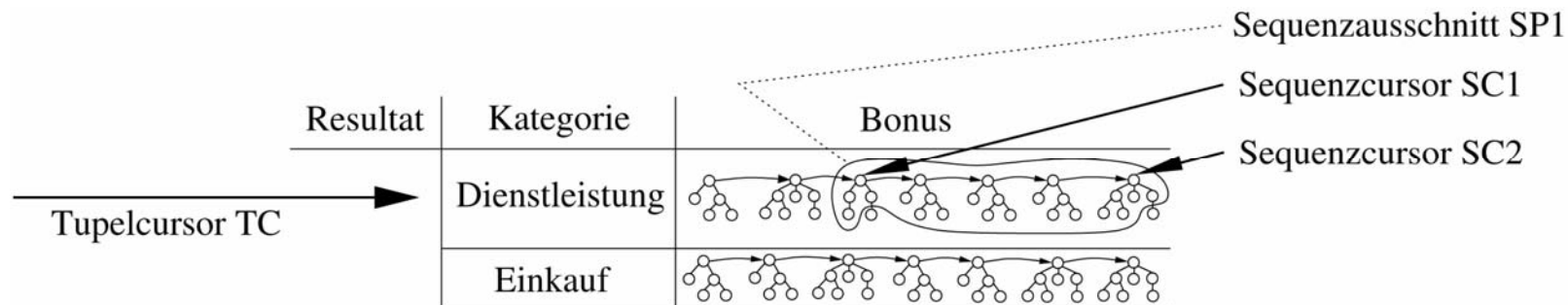
```
MOVE SC1, SC2 TO ROOT OF LAST ITEM;  
MOVE SC1 BACKWARDS 4 ITEMS;
```



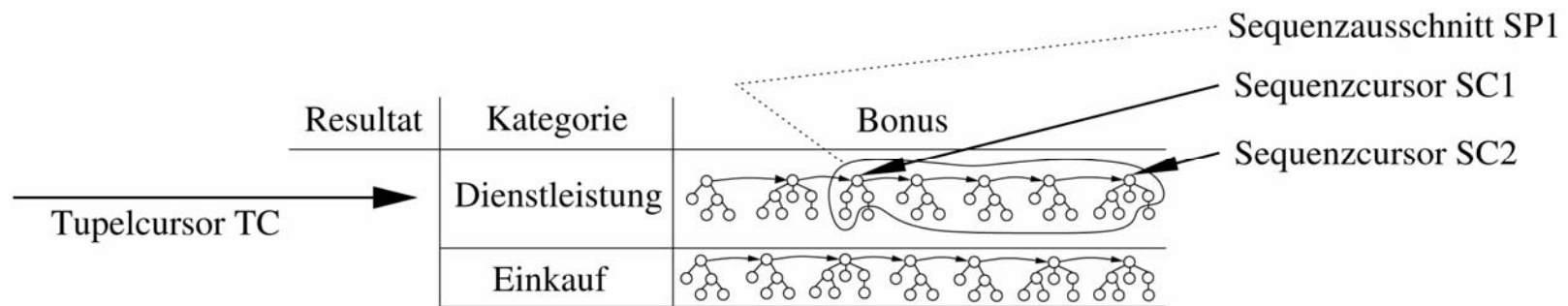
Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

- gewünschter Sequenzausschnitt lässt sich nun mit Bezugnahme auf die beiden Sequenzcursor definieren
- Kommentar- und Verarbeitungsanweisungsknoten seien für weitere Verarbeitung nicht von Interesse

```
DEFINE SEQUENCE PART SP1 AS FROM SC1 TO SC2  
EXCLUDING NODES OF KIND C, P;
```



Sequenzcursor-basierter Verarbeitungsablauf am Beispiel



- definierter Sequenzausschnitt „SP1“ umfasst die letzten fünf Einträge der Bonusangebotesequenz des ersten Ergebnistupels
- Sequenzausschnitt „SP1“ enthält genau die aufwertbaren Bonusangebote der Kategorie „Dienstleistung“

Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

- definierter Sequenzausschnitt „SP1“ wird nun ins AP übertragen
- Verwaltung im AP durch „Sequenzausschnittsverwaltung“ (SAV)
- in C++: Objekt der von uns bereitgestellten SAV-Klasse
- SAV-Objekt stellt Methoden für lesenden und ändernden Zugriff auf den Sequenzausschnitt zur Verfügung

```
TRANSFER SP1 INTO SEQUENCE PART MANAGER :*pSAV;
```

- pSAV ist Zeiger auf entsprechendes SAV-Objekt
- Bezugnahme auf SAV also vollkommen analog zu Bezugnahme auf SQLDA (SQL Descriptor Area) bei dynamischem embedded SQL

Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

- im Anwendungsprogramm lesendes und änderndes Arbeiten auf dem Sequenzausschnitt mittels spezieller DOM-ähnlicher „Sequenzausschnittsmethoden“
- Erhöhung des Gegenwerts des zweiten aufwertbaren Bonusangebots beispielsweise wie folgt möglich:

```
alt = pSAV->item(2).element("Wert").atomVal(1).getVal();  
pSAV->item(2).element("Wert").atomVal(1).setVal(alt+50);
```

- am Sequenzausschnitt lokal vorgenommene Änderungen werden von SAV protokolliert
- lokales Arbeiten auf dem Sequenzausschnitt also zunächst ohne direkte Interaktion mit dem Datenbanksystem

Sequenzcursor-basierter Verarbeitungsablauf am Beispiel

- lokal am Sequenzausschnitt vorgenommene Änderungen können in die Datenbank eingebracht werden

```
BRING IN LOCAL CHANGES OF SEQUENCE PART SP1;
```

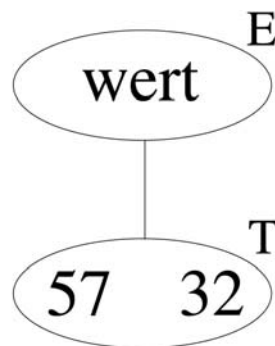
- um Bonusangebote der nächsten Kategorie zu verarbeiten, wird Tupelcursor mittels der SQL-FETCH-Anweisung um eine Position weiterbewegt
- Verarbeitung des nun aktuellen Ergebnistupels erfolgt analog...

Die typed-value-orientierte Repräsentation von XQuery-Sequenzen

- Verarbeitung basiert auf so genannter typed-value-orientierter Repräsentation von XQuery-Sequenzen und Sequenzausschnitten
- jeder im getypten Wert von Element- und Attributknoten enthaltene atomare Wert wird durch separaten Knoten repräsentiert
(→ Berliner XML-Tage 2005)
- Sequenzcursor können auf diese „neu eingeführten“ Knoten positioniert werden
- direkter Zugriff mit Hilfe der Sequenzausschnittsmethoden ebenfalls möglich

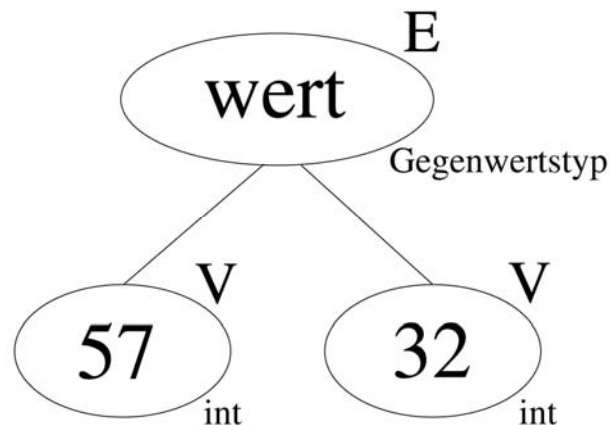
Die typed-value-orientierte Repräsentation von XQuery-Sequenzen

- Element `<wert>57 32</wert>` repräsentiert einlösbare Gegenwerte bei Folgeeinkauf im selben bzw. in einem anderen Geschäft
- Elementknoten „wert“ besitzt genau einen Sohn: den Textknoten „57 32“

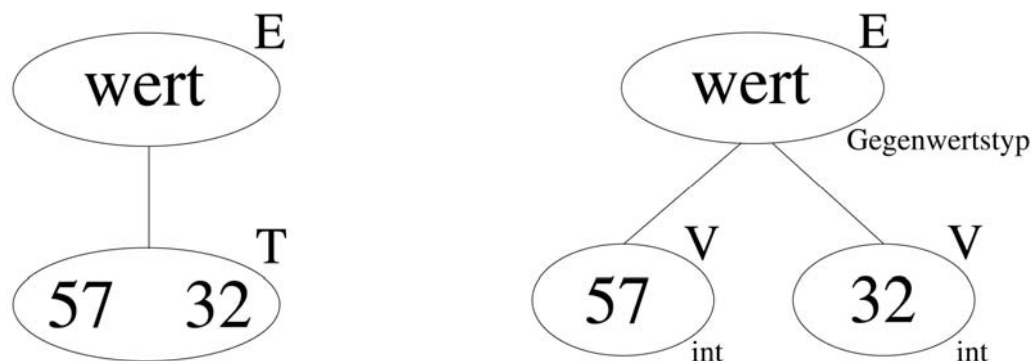


Die typed-value-orientierte Repräsentation von XQuery-Sequenzen

- Nehmen an, Bonusangebotesequenz, die diesen Elementknoten enthält, sei mittels XMLVALIDATE gegen geeignetes XML-Schema validiert worden
- Elementknoten „wert“ sei dabei der XML-Schema-Typ „Gegenwertstyp“ zugeordnet worden
- Dabei handle es sich um einen einfachen Typ (simple type), der als Elementinhalt eine Liste von Integer-Werten fordere
- Elementknoten „wert“ besitzt als getypten Wert die Integer-Liste (57, 32)



Die typed-value-orientierte Repräsentation von XQuery-Sequenzen



- Gegenwert bei Folgeeinkauf im selben Geschäft ist als linkerster Sohn des Elementknotens „wert“ ansprechbar
- Typed-value-orientierte Darstellung ist keine Erweiterung des XQuery-Datenmodells!!!
- Informationen, die laut XQuery-Datenmodell ohnehin Bestandteil einer XQuery-Sequenz sind, werden lediglich einfacher und einheitlicher zugreifbar gemacht

Abgrenzung gegen existierende Ansätze

- XQJ (XQuery API for Java) ist (noch in Entwicklung befindliche) Java-Schnittstelle zum Zugriff auf XQuery-Sequenzen
- Analogon zu JDBC: XQJ leistet für XQuery und XQuery-Systeme das, was JDBC für SQL und relationale DBSe leistet
- Obwohl nicht für relationale DBSe konzipiert, ist Nutzung im Zusammenhang mit *XML-fähigen* relationalen DBSen absehbar (wird von Sun unter Federführung von IBM und Oracle entwickelt)
- XQJ bietet Cursor zum eintragsweisen Durchlaufen einer XQuery-(Ergebnis-)Sequenz; beim gewünschten Eintrag angekommen, kann z. B. mittels DOM-Methoden in den Eintrag eingetaucht werden
- zwar ebenfalls Cursor zum Durchlaufen einer XQuery-Sequenz, jedoch vollkommen anderer Verarbeitungsansatz:
 - keine Definition von Sequenzausschnitten möglich
 - damit kein lokales Arbeiten auf Sequenzausschnitten

Abgrenzung gegen existierende Ansätze

- SQL:2007-Normentwurf bietet zwei prinzipielle Möglichkeiten für Zugriffe auf (im Anfrageergebnis enthaltene) XQuery-Sequenzen:
 - (1) Verarbeitung als „normale Zeichenketten“
 - (2) Nutzung der XMLTABLE-Funktion

Abgrenzung gegen existierende Ansätze

- (1) **In Ergebnistabelle enthaltene XQuery-Sequenzen werden als normale Zeichenketten ins AP übertragen:**
- XML-Werte werden beim Auslesen aus der Datenbank explizit oder implizit mittels der SQL:2007-Funktion `XMLSERIALIZE` in Zeichenketten serialisiert und dann als `VARCHAR`- oder `CLOB`-Werte ans AP übergeben
 - Alle Typinformationen gehen verloren; übertragene Zeichenkette muss im AP neu geparkt werden
 - Da Typinformation integraler Bestandteil einer XQuery-Sequenz, muss Verlust der Typinformation als wesentliches Manko dieses Vorgehens angesehen werden

Abgrenzung gegen existierende Ansätze

(2) Nutzung der SQL:2007-Funktion **XMLTABLE**:

- Tabellenfunktion, die in der FROM-Klausel einer SQL-SELECT-Anfrage verwendet wird
- XQuery-Sequenz wird dynamisch ins Tabellenformat überführt und wie eine Tabelle weiterverarbeitet
- Abbildungsvorschrift (Sequenz→Tabelle) nutzt XPath-Ausdrücke
 - Weiterverarbeitung der XQuery-Sequenz nicht in ihrer „natürlichen Form als Sequenz“, sondern als Tabelle
 - fraglich, ob stets eine geeignete Transformationsvorschrift gefunden werden kann
 - keinerlei Änderungen an den zugrunde liegenden Sequenzen möglich, da die dynamisch erzeugten Tabellen nicht änderbar sind
 - Erhöhung der in der Bonusangebotesequenz gespeicherten Gegenwerte wäre hier also nicht möglich

Zusammenfassung

- SQL-basierte integrierte Verarbeitung von XML-Werten und „traditionellen“ SQL-Daten ist von enormer Wichtigkeit
- SQL-Norm und relationale DBMS-Produkte stellen sich dieser Herausforderung
- XML-Unterstützung in SQL wird zur Zeit weiter ausgebaut
- SQL:2007 wird komplette XQuery-Sequenzen als Attributwerte erlauben
- Sequenzcursor-basierter Verarbeitungsablauf zur Verarbeitung von im Anfrageergebnis enthaltenen XQuery-Sequenzen
- Hauptidee: Sequenzausschnitte werden ins Anwendungsprogramm übertragen und dort lokal verarbeitet
- Beispielszenario „Kundenkartenverwaltung“ als Grundlage für Erläuterung der einzelnen Verarbeitungsschritte
- Abgrenzung gegen andere Ansätze

Ausblick

- stärkere Formalisierung und gegebenenfalls weitere Präzisierung der Sequenzcursor-anweisungen
- gleiches gilt für die zur lokalen Verarbeitung der Sequenzausschnitte genutzten Sequenzausschnittsmethoden
- Einbeziehung von Erfahrungen aus dem Einsatz des Prototyps

Letzte Folie

Fragen?

E-Mail: mueller@informatik.uni-jena.de