XML Vorlesung ETHZ, Sommersemester 2006

XML und Datenbanken

Erik Wilde
4.7.2006

http://dret.net/lectures/xml-ss06/

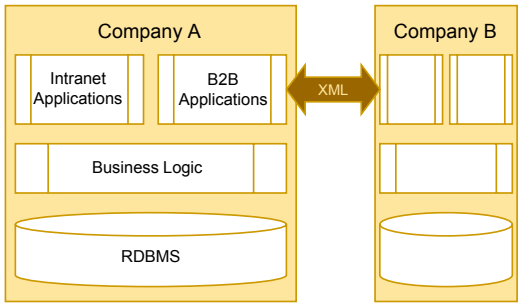
Übersicht

- Sichten auf XML
  - Transfer vs. Datenmodell, XML Support im DBMS
- XML vs. relationales Modell (ER-Modell)
- SQL/XML
  - Integration von XML in die Welt von SQL
- XQuery
  - eine eigene Abfragesprache für XML Datenbanken

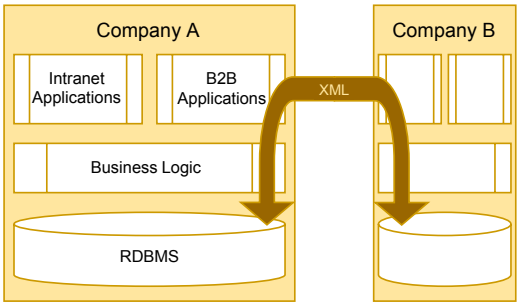
Sichten auf XML

- verschiedene Sichten sind möglich
- reine Transfersyntax
  - alles irgendwie auf XML abbilden
  - XML-basiert übertragen
  - beim Empfänger umgekehrtes Mapping
- XML als Datenmodell
  - XML Daten auf Business-Ebene
  - RDF Beschreibungen der XML-Daten
- momentan überwiegt (noch) die Transfersicht

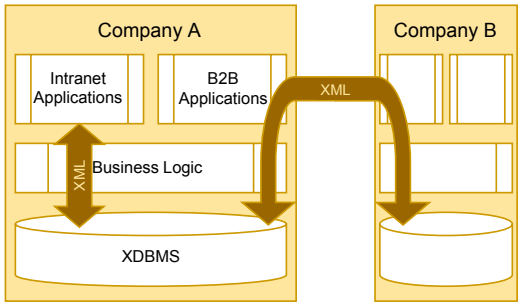
3-Tier und XML-basierter B2B



XML Support im DBMS



XML-Centric Applications



ER-Modellierung

- Entities repräsentieren Konzepte
  - reale Dinge wie Personen oder Produkte
  - abstrakte Dinge wie Projekte
- Relationships verbinden Konzepte
  - kein semantischer Gehalt im Modell
  - verschiedene Kardinalitäten unterstützt
- ER-Modelle sind beliebt und verbreitet
  - einfache Abbildung auf das relationale Modell
  - einfach zu verstehen und anzuwenden
  - für viele Probleme ausreichend gut

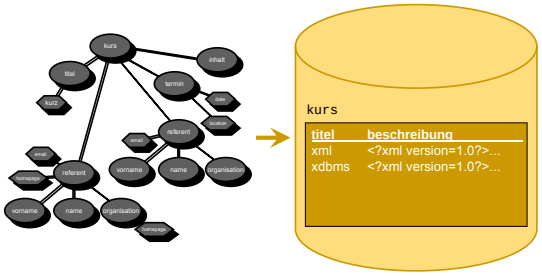
XML ist hierarchisch

- Erbe aus der Dokumentenwelt
  - Dokumente sind recht klar hierarchisch aufgebaut
  - bei modularem Aufbau schon weniger klar
    - `<!ELEMENT section (content, chapter+)`
      `<!ELEMENT chapter (content, subchapter+)`
    - `<!ELEMENT part (content, part+)`
- relational und hierarchisch gemischt
  - Entities als eigenständige Teile modellieren
  - klar hierarchische Teile als XML-Strukturen
- auch hier sind Entscheidungen nötig
  - welche Teile sollen wiederverwendbar sein?

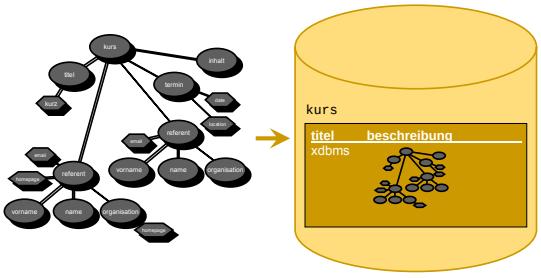
Datenbankhersteller und XML

- XML als reines Export/Import-Format
  - die Welt wird weiterhin relational modelliert
  - neue APIs für XML-Support
- XML als Sub-modell in der Datenbank
  - relationale Welt mit XML-orientierten Teilen
  - XML-spezifischer Support in der DB-Engine
- XML als "Informationsmodell" der Applikation
  - XML-basierte Datenbank
  - komplett neue DDL, DML, und Query-Sprache
  - Unterstützung der generischen Funktionen
    - Backup, Administration, Redundanz, Optimierung, ...

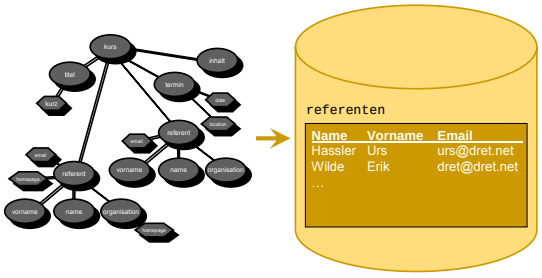
Variante 1a: XML → *LOB



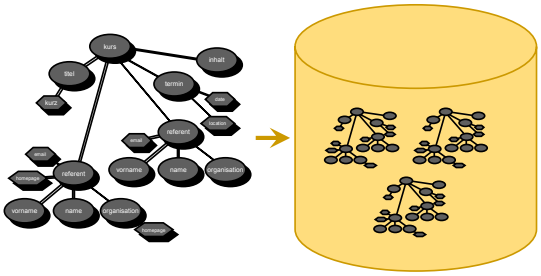
Variante 1b: XML → Spaltentyp



Variante 2: XML → Shredding



Variante 3: XML Datenbank



Relationships in XML

- XML basiert auf einer Dokumentensicht
  - es gibt eine vorgegebene hierarchische Sicht
  - Verweise in dieser Hierarchie sind möglich
- XML ist nicht immer ideal
  - auch Dokumente sind oft "modular" aufgebaut
  - Hierarchien definieren Hauptbeziehungen
  - ER-Modelle definieren gleichberechtigte Beziehungen
- XML modelliert Beziehungen auf zwei Arten
  - Inhalt (beliebig tiefe Verschachtelung)
  - Verweise (von einem Knoten auf einen anderen)

Referenzen in XML

- *well-formed* XML ist ein Baum
  - keine weitergehende Semantik
  - beliebige Semantik durch Schemasprachen möglich
- *valid* XML kennt Referenzen im Baum
  - Attribute werden als `ID` (Typ, nicht Name!) deklariert
    - Wert muss eindeutig sein im gesamten XML Dokument
  - Attribute werden als `IDREF(S)` deklariert
    - müssen auf existierende(n) `ID`s verweisen
- XML Schema verallgemeinert `ID/IDREF(S)`
  - DTD-Mechanismen oft zu primitiv und einschränkend

XML Schema Identity Constraints

- XML Schema berücksichtigt Datentypen
  - Werte mit unrelated Types sind niemals gleich
    - einer Instanz nicht unbedingt anzusehen
    - z.B. gilt `xs:integer 2` ist nicht gleich `xs:string 2`
    - aber `xs:integer 2` ist gleich `xs:integer +0002`
    - und `xs:string 2` ist nicht gleich `xs:string +0002`
  - Identity Constraints sind auf Instanzen definiert, und nicht auf ihren Typen
- d.h. am besten definiert man eigene Typen
  - z.B. Simple Type als Restriction von `xs:integer`
  - eigene Types für Key und KeyRef

Semistrukturierte Daten

- XML unterstützt semistrukturierte Daten
- Inhaltsmodelle mit 0-∞ Wiederholungen
  - mehrere Vornamen in einem Namen
  - Reihenfolge der Vornamen ist wichtig
- Text gemischt mit Elementen
  - Text vermischt mit strukturiertem Inhalt
  - `<p>Dieser <i>Text</i> benutzt... </p>`
- keine gute Abbildung auf ER-Konstrukte möglich
  - Vermeidung dieser Aspekte im Modell
  - Speicherung als XML-Struktur

SQL:2003 Datentyp XML

- neuer Basisdatentyp in SQL:2003
  - vergleichbar `BOOLEAN`, `INTEGER`, `CHAR`, `DATE`, ...
  - damit im Grundvokabular der Sprache enthalten
- erlaubte Werte des XML Datentyps
  - NULL
  - XML Content (well-formed XML Entity)
  - XML Documents (well-formed XML Dokument)
- kann in DDL Statements benutzt werden
  - `CREATE TABLE DEMO ( ID INTEGER, TEXT XML)`

Eigenschaften des XML Datentyps

- keine Ordnung
  - auch die Gleichheit ist nicht definiert
    - offene Fragen der Zeichenkodierung, Infoset vs. XML, usw.
  - lässt sich benutzerspezifisch definieren
    - `CREATE ORDERING FOR ...`
    - potentiell aufwendige Operation
- Erzeugung von XML aus anderen Werten
  - Auswertung von XQuery Anfragen
  - Konstruktoren für XML Knoten
  - Kombinieren bestehender XML Strukturen

SQL/XML Funktion XMLGEN

- basiert auf XQuery
  - noch nicht abgeschlossene Standardisierung
  - erste prototypische Implementierungen
- Argument ist ein XQuery Ausdruck
  - optional mit einer Liste von Variablen
  - Variablen sind beliebige SQL Wertausdrücke
    - Verwendung im XQuery
- Resultat ist immer vom Typ XML
  - XQuery wird später näher betrachtet

SQL/XML Funktion XMLELEMENT

- Erzeugung eines Elements
  - Name muss angegeben werden
  - optional können Attribute angegeben werden
    - markiert mit `XMLATTRIBUTES`

```
SELECT XMLELEMENT(NAME "kurs",
                  XMLATTRIBUTES(id AS Nummer),
                  titel) AS Kursliste FROM Kurse
```

Kursliste:

```
<kurs Nummer="1">XML</kurs>
<kurs Nummer="2">XSLT</kurs>
<kurs Nummer="3">XML und Datenbanken</kurs>
```


SQL/XML Funktion XMLFOREST

- Erzeugung eines Waldes von XML Elementen
  - kein umschliessendes Element
  - d.h. für sich betrachtet kein XML Dokument
  - Angabe einer Elementliste als Argumente

```
SELECT XMLFOREST(id AS "Nummer",
                 titel   )
                 AS Kursliste FROM Kurse
```

Kursliste:

```
<Nummer>1</Nummer><titel>XML</titel>
<Nummer>2</Nummer><titel>XSLT</titel>
<Nummer>3</Nummer><titel>XML und Datenbanken</titel>
```


SQL/XML Funktion XMLCONCAT

- Zusammenfügen von XML Strukturen
  - Liste von XML-Werten (keine Dokumente)
  - erzeugt einen XML-Wert pro Zeile

```
SELECT XMLCONCAT(XMLELEMENT(NAME "Nummer", id ),
                 XMLELEMENT(NAME "kurs", titel))
                 AS Kursliste FROM Kurse
```

Kursliste:

```
<Nummer>1</Nummer><kurs>XML</kurs>
<Nummer>2</Nummer><kurs>XSLT</kurs>
<Nummer>3</Nummer><kurs>XML und Datenbanken</kurs>
```


SQL/XML Funktion XMLAGG

- aggregiert mehrere XML-Werte
  - funktioniert über Zeilen hinweg
  - Zusammenführen von Results in einem XML-Wert
  - optionale Sortierung kann angegeben werden

```
SELECT XMLELEMENT(NAME "kursliste", TMP) FROM
  ( SELECT XMLAGG(XMLELEMENT(NAME "Nummer", id ),
                  XMLELEMENT(NAME "kurs", titel)) AS TMP
                  FROM Kurse ORDER BY titel)
```

```
<kursliste>
 <Nummer>1</Nummer><kurs>XML</kurs>
 <Nummer>3</Nummer><kurs>XML und Datenbanken</kurs>
 <Nummer>2</Nummer><kurs>XSLT</kurs>
</kursliste>
```

SQL/XML Funktion XMLCOLATTVAL

- von Oracle vorgeschlagen und implementiert
  - nicht Bestandteil des SQL/XML Standards
  - erzeugt Elemente aus Spalten

```
SELECT XMLELEMENT(NAME "kurs",
                  XMLCOLATTVAL(id, titel))
                  AS Kursliste FROM Kurse
```

Kursliste:

```
<kurs>
 <column name="id">1</column>
 <column name="titel">XML</column>
</kurs>
```


Weitergehende Regeln

- Abbildungen von Namensräumen
  - SQL und XML haben andere Konventionen
  - u.U. nicht transparente Umwandlung
- Abbildung von Datentypen
  - SQL und XML Schema Datentypen
  - komplexe SQL Datentypen ergeben XML Strukturen
    - z.B. werden SQL Tupel zu Elementlisten

Zusammenfassung SQL/XML

- einfache Funktionen zur Erzeugung von XML
  - XML Elemente
  - XML Attribute
  - XML Entities (Listen von Elementen)
  - XML Dokumente
- keine Unterstützung von
  - DML Konstrukten
  - Volltextsuche auf XML Dokumenten

W3C XQuery/XSLT Working Group

- Vereinigung von XQuery und XSLT
  - XQuery aufbauend auf Quilt
  - XSLT 2.0 aufbauend auf XSLT 1.1
- grosse Arbeitsgruppe mit vielen Teilnehmern
  - alle wichtigen Firmen sind vertreten
  - grosse Wahrscheinlichkeit für Akzeptanz
- aufwendiger Entwicklungsprozess
  - parallel aktualisierte Requirements Dokumente
  - Parser und Specs automatisch erzeugt aus EBNF
  - aktueller Stand sind einige Last Call WDs (03/05)

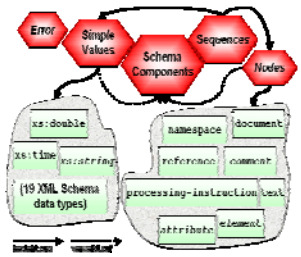
Querying XML: Language Wars?

- Produkte werden diverse Sprachen anbieten
  - SQL/XML erzeugt XML aus SQL Queries
    - evolutionärer Ansatz auf dem relationalen Datenmodell
    - erlaubt Zugriff auf andere SQL Features
    - Weiterverwendung bestehender SQL Tools/Applikationen
  - XQuery als neue Technologie
    - operiert auf XML als Datenmodell
    - keine DML in der ersten Version, kein Full-Text Support
    - geht von einer XML-zentrierten Anwendung aus
  - XPath als grundlegende Selektionssprache
    - von vielen frühen XML-Datenbanken unterstützt
    - keine echte Query-Sprache, nur einfache Prädikate

XQuery, XPath und XSLT

- XSLT als Style Sheet Sprache für XML
  - Transformationsteil als Programmiersprache
  - XPath als Expression Language von XSLT
- XSLT 1.0 und XPath 1.0 sind recht einfach
  - sehr schwaches Typsystem
  - praktisch keine Typfehler
- XSLT 2.0 und XPath 2.0 sind viel komplexer
  - XPath 2.0 als gemeinsame Basis für XSLT/XQuery
  - XPath 1.0 kann keine first-order logic
    - XPath 2.0 erweitert XPath 1.0 erheblich

XQuery Datentypen



- 5 Arten von Datentypen:
  - Knoten
  - Folgen
  - Schemakomponenten
  - einfache Werte
    - 19 XML Schema Datentypen
  - Fehler
    - ein einziger Fehlertyp
- alle Wertebereiche (bis auf Fehler) sind unendlich

XQuery unterstützt Static Typing

- alle Konstrukte sind typgestützt
  - XML Schema als Voraussetzung
- statische Typanalyse ist möglich
  - weniger Laufzeitfehler
  - XML Schema und XQuery Code reichen aus
    - XQueries können Typen deklarieren (bessere Tests möglich)
- statische Typen haben weitere Vorteile
  - bessere Optimierungen sind möglich
  - generell robustere Software
  - bessere Vorhersage der erwarteten Resultattypen

FLWOR Expressions

- vergleichbar dem `SQL SELECT FROM WHERE`
- vergleichbar mit XSLT Konstrukten
  - `FOR` → `xsl:for-each`
  - `LET` → `xsl:variable`
  - `WHERE` → `xsl:if`
  - `ORDER BY` → `xsl:sort`
- deklarative Programmiersprache
  - keine Konstrukte aus prozeduralen Sprachen
- funktional kein tatsächlicher Unterschied
  - mit XSLT und XQuery sind gleiche Resultate möglich

Zusammenfassung XQuery

- XQuery als neue Query-Sprache
  - operiert auf XML als Datenmodell
- XPath 2.0 als Grundlage von XQuery
  - ebenfalls die Grundlage für XSLT 2.0
  - wird Grundwissen für XML-Experten werden
- momentan noch in Entwicklung
  - Syntax kann sich noch ändern
  - stabiler Standard gegen Ende 2005 zu erwarten

Zusammenfassung

- XML wird immer wichtiger
  1. XML als Transferformat zwischen Anwendungen
  2. XML als Schnittstelle zwischen Anwendungen
  3. XML als Datenmodell von Anwendungen
- der Grossteil aller Daten ist in Datenbanken
  - zunehmende Unterstützung von XML
- Fusion der Datenmodelle
  - viele Daten sind angemessen mit ER modelliert
  - Teile sind u.U. besser in XML modelliert