XML Anfragesprachen
XML Query Languages

Tibor Dekany

Der Vergleich von
LOREL, XML−QL, XML−GL, XSL, XQL

Überblick

1. Einführung: 5 Anfragesprachen
2. Datenmodell: XML−GL, XSL, XQL
3. Syntax: XML−GL, XSL, XQL
4. Vergleich und Beispiele
5. Zusammenfassung

SQL ähnliche QL

- LOREL (Lightwight Object REpository Language)
  - für halbstrukturierte Datenbanksysteme
  - für XML erweitert
  - Integration heterogener Daten
- XML−QL
  - um "CONSTRUCT" erweitertes SQL
  - Anfrage und Transformation über mehrere XML Dokumente

Grafische QL

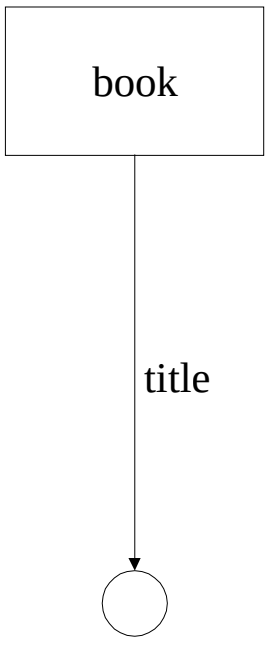
- XML−GL
  - basiert auf der grafischen Betrachtungsweise von XML Dokumenten und DTD’s als beschriftete Graphen
  - geeignet für ein visuelles WYSIWYG Interface

Nicht vollständige QL

- XSL (XML Stylesheet Language)
  - Eigentliche Rolle: Darstellungsschicht
  - Besteht aus 2 Teilen
    - Transformation (XSLT)
    - Formatierung (XSL formatted object)
  - Geeignet als Basis für neue Anfragesprachen
- XQL
  - Kann nur auswählen und filtern, unterstützt keine Transformation

Datenmodell XML−GL

- Objekte
- Beziehungen
- Eigenschaften



Datenmodell XSL (XPath)

- Wurzelknoten (root node)
- Elementknoten
- Attributknoten
- Namensraumknoten
  <article xmlns:xlink="http://www.w3.org/1999/xlink"> ... </article>
- Instruktionsknoten (PI) <?.....?>
- Kommentarknoten <!−− Kommentar −−>
- Textknoten (CDATA)

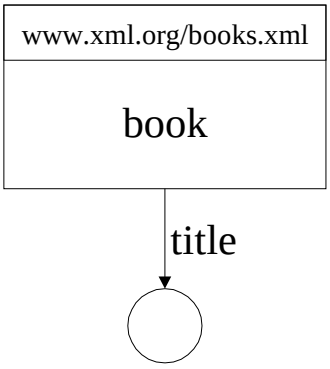
Datenmodell XQL (XML)

XQL benutzt die allgemeine Struktur von XML:

- **Eigenschaften** von Knoten
  - Typ: Dokument, Element, PI, Kommentar, CDATA, ...
  - Name: Elementname, Attributname, PI target, ...
  - Inhalt / Wert: Elementinhalt, Attributwert, ...
- **Beziehungen** zwischen Knoten
  - Hierarchie: Eltern / Kind
  - Sequenz: vorher, unmittelbar vorher
  - Position: absolut, relativ, im Bereich von

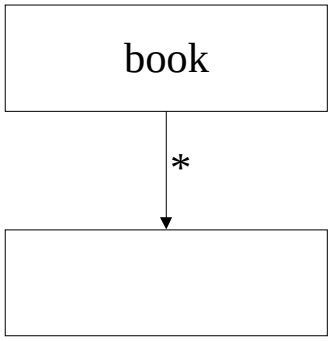
Syntax XML−GL

- extract part: Auswahl des Zieles (XML Dokumente und Bereiche).
- match part (optional): logische Konditionen (SQL: where)



Database Systems

- construct part: Stuktur des Ergebnisdokumentes
- clip part (optional): Im Ergebnisdokument zu berücksichtigenden Elemente (+ rekursive Subelemente)



Syntax XSL

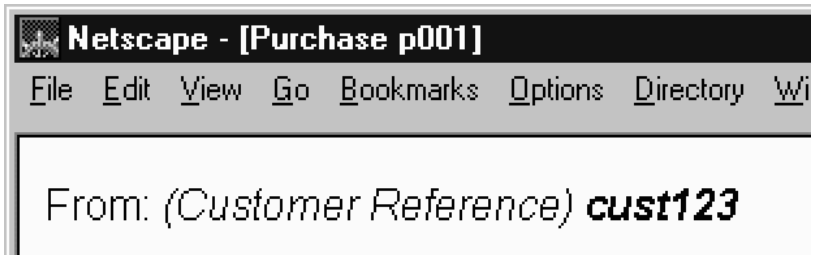
```
'<xsl:template' [ 'match=' pattern_expr ] '>'
    { xsl:directive }
    { result-elements }
'</xsl:template>'
```

```
'<xsl:apply-templates' [ 'select=' pattern_expr ] '>'
'<xsl:for-each' 'select=' pattern_expr '>'
'<xsl:value-of' [ 'select=' pattern_exprt ] '>'
'<xsl:copy-of' 'select=' pattern_expr '>'
```

Beispiel XML Datei

```
www.inter.net/document.xml
<?xml version="1.0"?>
<purchase id="p001">
    <customer db="cust123"/>
    <product db="prod345">
        <price>23.45</price>
    </product>
    <product db="prod678">
        <price>19.90</price>
    </product>
</purchase>
```

Beispiel: XSL Direktive




```
<xsl:template match="customer">
    <p><xsl:text>From: </xsl:text>
    <i><xsl:text>(Customer Reference) </xsl:text>
    <b><xsl:value-of select="@db"/>
    </b></i></p>
</xsl:template>
```

Beispiel: XSL Direktive

```
<xsl:template
        match="purchase[customer/@db='cust123']">
    <xsl:for-each select="product[price>=20]">
        <xsl:comment> This is expensive </xsl:comment>
        <xsl:copy-of select="."/>
    </xsl:for-each>
</xsl:template>
```

Syntax XQL

```
Query ::=  [ './' | '//' | '///' | './//' ]
   Element [ '[' Predicate ']' ] [ Path ]

Path ::=   [ '//' | '///' ] Element [ '['
   Predicate ']' ] [ Path ]
```

Term / Operator	Beispiel
Elementname	author
Joker	*
Attributname	@id
Gleichheit	first−name='Urs'
Eltern/Kind	author/firstname
Vor−/Nachfahre	book//first−name
Filter	author[first−name='Urs']
Neg./Konj./Disjunktion	notx ; xandy ; xory
Vereinigung/Durchschnitt	x$union$y ; x$intersect$y
Gruppierung	xand(yorz)

Beispiel XML Datei 1

www.inter.net/manufacturers.xml

```
<manufacturer>
     <mn_name>Fiat</mn_name>
     <year>1999</year>
     <model>
          <mo_name>Punto</mo_name>
          <front_rating>3.84</front_rating>
          <side_rating>2.14</side_rating>
          <rank>9</rank>
     </model>
     <model> ...</model>
</manufacturer>
<manufacturer>

...

</manufacturer>
```

Beispiel XML Datei 2

www.inter.net/vehicles.xml

```
<vehicle>
     <vendor>Urs Müller</vendor>
     <make>Fiat</make>
     <model>Punto</model>
     <year>1999</year>
     <color>metallic blau</color>
     ...
     <price>19999</price>
</vehicle>
<vehicle>
...
</vehicle>
```

1. Selektion / Ausgabe

- Ausgabe aller **<manufacturer>** Elemente, welche mindestens ein **<model>** unter den top 10 haben.

1. Lösung in XQL

```
manufacturer[model/rank<=10]
```

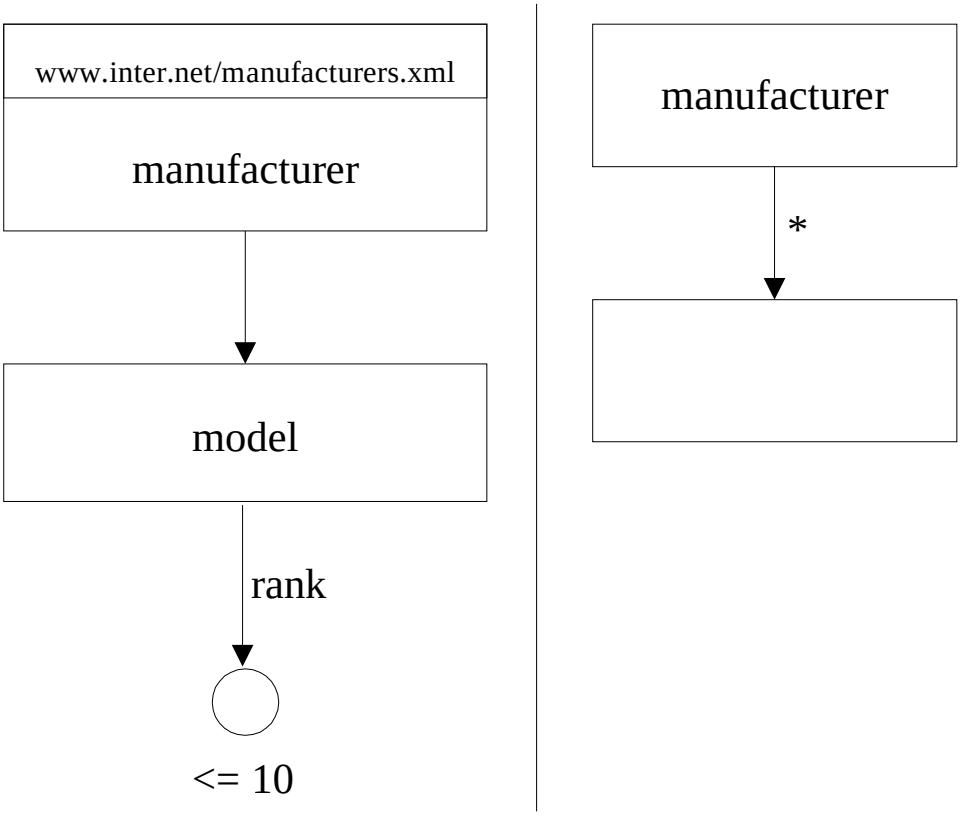
1. Lösung in XSL

```
<xsl:template match="/">
    <xsl:for-each
        select="manufacturer[model/rank<=10]">
        <xsl:value-of />
    </xsl:for-each>
</xsl:template>
```

oder

```
<xsl:template
match="/manufacturer[model/rank<=10]">
    <xsl:apply-templates />
</xsl:template>
```

1. Lösung in XML-GL



1. Lösung in LOREL

```
select M
from db.manufacturer M
where M.model.rank <= 10
```

1. Lösung in XML−QL

```
WHERE  <manufacturer>
           <model>
               <rank>$r</rank>
           </model>
       </manufacturer> ELEMENT_AS $m
   IN www.inter.net/manufacturers.xml, $r<=10
CONSTRUCT $m
```

2. Reduktion

- Aufgabe: Lasse vom manufacturers.xml Dokument alle **<model>** Subelemente weg, welche einen **<rank>** grösser als 10 haben. Ausserdem soll das Ergebnis weder **<front_rating>** noch **<side_rating>** enthalten.

2. Lösung in XQL

Keine Lösung in XQL, da XQL keine

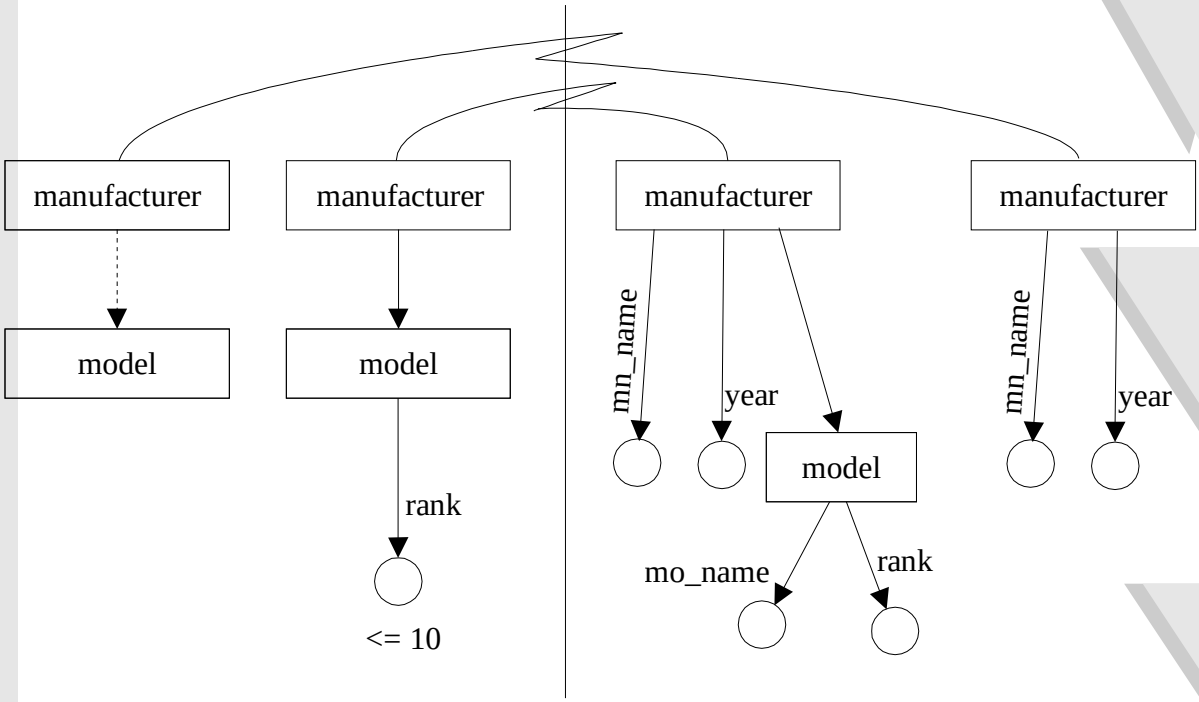
- Reduktion
- Konstruktion
- verschachtelte Anfragen

unterstützt

2. Lösung in XSL

```
<xsl:template
match="manufacturer[model/rank<=10]">
    <model>
        <xsl:value-of select="mo_name"/>
        <xsl:value-of select="rank"/>
    </model>
</xsl:template>
```

2. Lösung in XML-GL



2. Lösung in LOREL

```
select Z.mn_name, Z.year,
     (select Z.model.mo_name, Z.model.rank
     where Z.model.rank <= 10)
from db.manufacturer Z
```

2. Lösung in XML−QL

```
WHERE    <manufacturer>
             <mn_name>$mn</mn_name>
             <year>$y</year>
         </manufacturer> CONTENT_AS $m IN
             www.inter.net/manufacturers.xml
CONSTRUCT
    <manufacturer>
         <mn_name>$mn</mn_name>
         <year>$y</year>
         { WHERE
             <model>
                 <mo_name>$mon</mo_name>
                 <rank>$r</rank>
             </model> IN $m, $r<=10
         CONSTRUCT
             <model>
                 <mo_name>$mon</mo_name>
                 <rank>$r</rank>
             </model> }
</manufacturer>
```

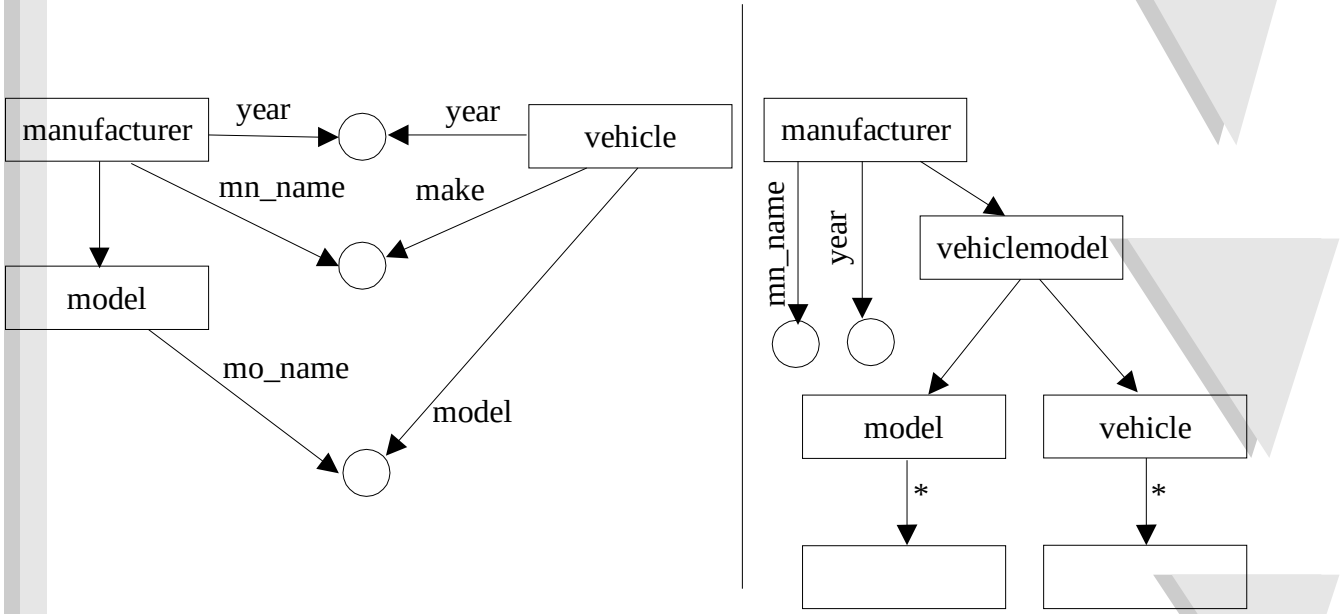
3. Joins

- Aufgabe: finde Paare von **<manufacturer>** und **<vehicle>**, unter der Bedingung, dass **<mn_name> = <make>**, **<mo_name> = <model>** und **<year> = <year>**.

2. Lösung in XQL / XSL

Keine Lösung, da beide Sprachen Joins nicht unterstützen und ausserdem nur ein Dokument abfragen können.

3. Lösung in XML−GL



3. Lösung in LOREL

```
temp:=   select (M,V) as pair
     from db.manufacturer M, db.vehicle V
     where M.mn_name = V.make
          and M.model.mo_name = V.model
          and M.year = V.year
```

3. XML–QL

```
WHERE <manufacturer>
          <mn_name>$mn</mn_name>
          <year>$y</year>
          <model>
               <mo_name>$mon</mo_name>
          </model> CONTENT_AS $mo
     </manufacturer> CONTENT_AS $m IN
          www.inter.net/manufacturers.xml
     <vehicle>
          <model>$mon</model>
          <year>$y</year>
          <make>$mn</make>
     </vehicle> CONTENT_AS $v IN
               www.inter.net/vehicles.xml
CONSTRUCT
     <manufacturer>
          <mn_name>$mn</mn_name>
          <year>$y</year>
          <vehiclemodel>
               $mo,$v
          </vehiclemodel>
     </manufacturer>
```

4. Restrukturierung

- Das Ergebnis der letzten Aufgabe soll wie folgt dargestellt werden:

```
<car>
     <make/>
     <model/>
     <vendor/>
     <rank/>
     <price/>
</car>
```

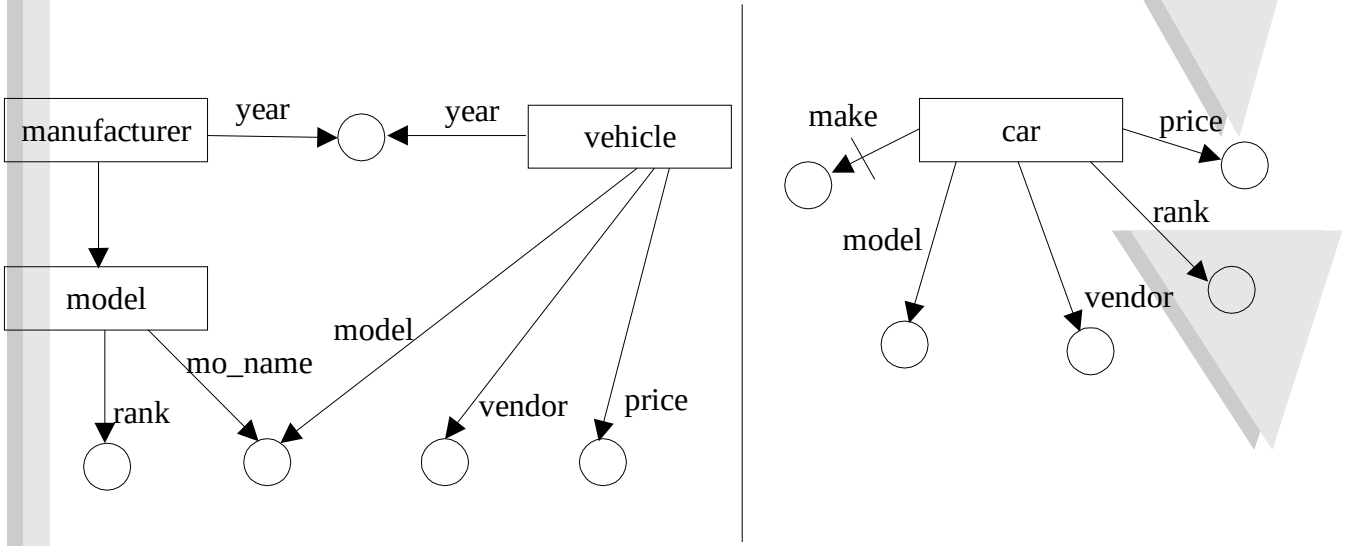
4. Lösung in XQL

Keine Lösung in XQL, da XQL Konstruktion nicht unterstützt und somit die Struktur des Ergebnisses nicht beeinflussen kann.

4. Lösung in XSL

```
<xsl:template match="manufacturer">
  <car>
     <xsl:value-of
          select="vehiclemodel/vehicle/make"/>
     <xsl:value-of
          select="vehiclemodel/model/mo_name"/>
     <xsl:value-of
          select="vehiclemodel/vehicle/vendor"/>
     <xsl:value-of
          select="vehiclemodel/model/rank"/>
     <xsl:value-of
          select="vehiclemodel/vehicle/price"/>
  </car>
</xsl:template>
```

4. Lösung in XML−GL



4. Lösung in LOREL

```
select xml(car: (select   X.vehicle.make,
                          X.vehicle.model,
                          X.vehicle.vendor,
                          X.manufacturer.rank,
                          X.vehicle.price
from temp.pair X))
```

```
WHERE <manufacturer>
        <mn_name>$mn</mn_name>
        <vehiclemodle>
            <model>
                <mo_name>$mon</mo_name>
                <rank>$r</rank>
            </model>
            <vehicle>
                <price>$p</price>
                <vendor>$mn</vendor>
            </vehicle>
        </vehiclemodel>
    </manufacturer> IN www.inter.net/queryresult3.xml
    CONSTRUCT    <car>
                    <make>$mn</make>
                    <mo_name>$mon</mo_name>
                    <vendor>$v</vendor>
                    <rank>$r</rank>
                    <price>$p</price>
                </car>
```

5. Zusammenfassung

	LOREL	*XML−QL*	*XML−GL*	*XSL*	*XQL*
Eigenes Datenmodell	X	X	X	X	−
Joins	X	X	X	−	−
Abbruch bei Zyklen	X	undef.	X	−	undef.
Existentielle Quantifizierung	X	X	X	X	X
Universelle Quantifizierung	X	−	−	−	X
Negation	X	−	X	X	X
Reduktion	−	−	−	−	−
Konstruktion neuer Elemente	X	X	X	X	−
Verschachtelte Anfragen	X	X	−	X	−
Resultat sortieren	X	X	X	X	−
Typenunterstützung	X	−	−	−	−
Unterstützung XPointer & XLink	−	−	−	−	−