Kapitel 5
Anfragesprachen: XPath

- Netzwerkmodell: keine Anfragesprache
- SQL – nur für flaches Datenmodell, aber "schöne" Sprache
  (leicht zu lernen, deskriptiv, Relationale Algebra als Basis, saubere Theorie, Optimierbarkeit)
- OQL: SQL mit Objektorientierung und Pfadausdrücken
- Lorel (OEM): OQL-Erweiterung
- F-Logic: Navigation in einem Graphen durch Pfadausdrücke mit Bedingungen
  deskriptiv, komplex.

Anforderungen an eine XML-Anfragesprache

- gleichermaßen tauglich für Datenbanken und Dokumente
- deklarativ
- navigationsbasiert (Pfade) und klauselbasiert (SQL-Stil)

ENTWICKLUNG ZU XPATH

- Um eine Anfragesprache zu definieren, benötigt man einen Formalismus, um auf einzelne Elemente/Attribute oder Mengen von Elementen/Attributen zuzugreifen.
- Auf diesem *Adressierungsmechanismus* baut dann eine klauselbasierte Sprache auf.

Anfangszeit von XML (1998)
verschiedene Navigationsformalismen dieser Art:

- XSL Patterns (innerhalb der Stylesheet-Sprache)
- XQL (XML Query Language)
- XPointer (Referenzieren von Knoten/Bereichen innerhalb eines XML-Dokuments)

verwenden alle dieselbe Grundidee mit leichten Detailunterschieden:

- Pfade in UNIX-Notation
- Bedingungen auf dem Pfad

  /mondial/country[@car_code="D"]/city[population > 100000]/name

XPATH

1999: Festlegung des Navigationsformalismus als *W3C XPath*.

- Grundlage: UNIX-Directory-Notation

  Innerhalb des UNIX Directory-Baumes: /home/dbis/Mondial/mondial.xml
  Innerhalb des XML-Baumes: /mondial/country/city/name

  Damit: logische Erweiterung der URL-Spezifikation.
  /home/dbis/Mondial/mondial.xml#mondial/country/city/name
  [XPointer; später]

- W3C: XML Path Language (XPath), Version 1.0
  `http://www.w3.org/TR/xpath`
- W3C: XPath 2.0 and XQuery 1.0
  `http://www.w3.org/TR/xquery`

XPath: Navigation, einfache Beispiele

XPath basiert auf der UNIX-Directory-Schreibweise:

- /mondial/country
  adressiert alle country-Elemente in mondial,
  Ergebnis ist eine Menge von Elementen der Form

  <country code="..."> ... </country>

- /mondial/country/city
  adressiert alle city-Elemente, die direkte Subelemente von country-Elementen sind.
- /mondial/country//city
  adressiert alle city-Elemente, die Subelemente von country-Elementen sind.
- //city
  adressiert alle city-Elemente im aktuellen Dokument.
- Wildcards für Elementnamen:
  /mondial/country/*/capital
  adressiert alle capital-Elemente, die Enkel von country-Elementen sind
  (Unterschied zu /mondial/country//capital !)

... und jetzt systematisch:

XPath: Zugriffspfade in XML-Dokumenten

- Navigationspfade

  /step/step/.../step

  setzen sich aus einzelnen Navigationsschritten zusammen,
- das Ergebnis jedes Schrittes ist eine Menge von Knoten, die als Eingabe für den nächsten Schritt verwendet wird.
- jeder Schritt besteht aus

  axis::*nodetest*[*condition*]

  - der Angabe einer Achse (optional),
  - einem Knotentyp-Test,
  - einem Prädikat (optional), das jeweils für den aktuellen Knoten ausgewertet wird.
- Pfade werden durch den Pfad-Operator "/" zusammengesetzt
- ausserdem gibt es Funktionsanwendungen.
- das Ergebnis jedes XPath-Ausdrucks ist eine *Folge (Sequence)* von Knoten oder Literalen.

XPath: Achsen

Man kann in dem XML-Baum von einem Element ausgehend in verschiedene “Richtungen” navigieren (vgl. xmllint’s “cd”-Kommando)

In jedem Navigationsschritt

path/*axis*::*nodetest*[*condition*]/*path*

kann eine *Achse (axis)* angegeben werden, entlang der navigiert werden soll. Kontextknoten für die Achse ist jeweils der durch *path* referenzierte Knoten.

- Default: child-Achse: child::country ≡ country.
- Self-Achse: self::city ≡ ./city
  ergibt das aktuelle Element, *falls* dieses vom Elementtyp city ist.
  Es kann auch mit current() angesprochen werden.
- Descendant-Achse: alle Sub-, Subsub-, ... elemente:
  country/descendant::city ≡ country//city
  selektiert alle city-Elemente, die (beliebig tief) innerhalb von country-Elementen liegen.
  Anmerkung:
  Man beachte descendant::city ≡ .//city ≢ //city

XPath: Achsen

... noch eine wichtige Achse:

- Attribut-Achse:
  attribute::car_code ≡ @car_code
  Wildcard für Attribute: attribute::* selektiert alle Attribute des aktuellen Kontextknotens.

Für die bisher genannten Achsen existieren die angegebenen Abkürzungen. Dies spielt für XSL-Patterns eine Rolle (siehe Folie 242):

XSL-(Match)-Patterns sind diejenigen XPath-Ausdrücke, die *ohne* Verwendung von “axis::” gebildet werden können (die genannten Abkürzungen sind erlaubt).

XPATH: ACHSEN

Weiterhin gibt es Achsen, für die keine Abkürzung definiert ist:

- Parent-Achse: //city[name="Berlin"]/parent::country
  ergibt das Eltern-Element des city-Elements "Berlin", *falls* dieses vom Elementtyp country ist.
  (*nur* das Eltern-Element, nicht alle Vorfahren!)
- ancestor: alle Vorgänger:
  //city[name="Berlin"]/ancestor::country selektiert das country-Element, in dem das city-Element Berlin liegt.
  Analog für descendant-or-self und ancestor-or-self.
- Geschwister: following-sibling::...
  (macht Sinn, wenn man Dokumente hat, in denen die Ordnung wichtig ist).
  Analog für preceding-sibling.

XPATH: NODETEST

- der *nodetest* legt den Knotentyp und den Namen der auszuwählenden Knoten fest
- "*" als Wildcard erlaubt

XPATH: ZUGRIFF AUF TEXTINHALT UND KNOTENNAMEN

- direkter #PCDATA-Inhalt von Elementen:
  der Knotentest text() selektiert alle Text-Subknoten des aktuellen Elements.
  Ergebnis ist eine Folge von Strings.
  /mondial/country/name/text()
- Die Funktion string() ergibt den gesamten (geschachtelten) Textinhalt des ersten Knotens der Argumentfolge:
  string(mondial/country[@car_code='D'])
- Die Funktion name() ergibt den Tag-Name des aktuellen Knotens:
  name(//city[name='Berlin'])
  //*[name='Monaco' and not (name()='country')]

XPath: Tests

In jedem Schritt

path/axis::nodetest[condition]/path

ist *condition* ein Prädikat über XPath-Ausdrücken.

- es werden nur die Knoten aus der Ergebnismenge von *path/axis::nodetest* ausgewählt, die *condition* erfüllen. *condition* enthält XPath-Ausdrücke, die relativ zu dem dort gerade aktuellen *Kontextknoten* ausgewertet werden.

  //country[@car_code="D"]

  gibt das country-Element zurück, dessen car_code-Attribut den Wert "D" hat

- Bei Vergleichen, die ein Element beinhalten, wird implizit die text()-Methode aufgerufen:

  //country[name = "Germany"] ist äquivalent zu
  //country[name/text() = "Germany"]

- Ist die rechte Seite eines Vergleichs eine Zahl, wird automatisch numerisch verglichen:

  //country[population > 1000000]

XPath: Tests (Forts.)

- XPath-Ausdrücke in *condition* haben EXISTS-Semantik:

  //country[inflation]

  gibt diejenigen Länder aus, die ein Subelement vom Typ Inflation haben.

  //country[.//city/name='Cordoba']/name

  gibt die Namen derjenigen Länder aus, in denen eine Stadt namens Cordoba liegt.

  //country[not .//city/name='Cordoba']/name

  gibt die Namen derjenigen Länder aus, in denen keine Stadt namens Cordoba liegt.

XPATH

- Man kann Bedingungen und Pfadoperatoren beliebig schachteln:

  //country[population > 10000000]/name/text()
  //country[inflation AND population > 1000000]/@car_code
  //country[population > 10000000]
  　　　　//city[@is_capital AND population > 1000000]
  　　　　　　/name/text()

  Die letzte Anfrage gibt die Namen aller Städte aus, die mehr als 1,000,000 Einwohner haben und Hauptstadt eines Landes mit mehr als 10,000,000 Einwohnern sind.

XPATH: AUSWERTUNGSSTRATEGIE

- Eingabe zu jedem Navigationsschritt: Eine Menge von Knoten (*Kontext*)
- jeder dieser Knoten wird zur Auswertung des aktuellen Schrittes einzeln betrachtet
- und liefert ein oder mehrere Knoten als (Zwischen)Ergebnis
  Dieses Zwischenergebnis dient als Kontext im nächsten Schritt
- Am Ende werden alle Teilergebnisse zusammengefaßt und zurückgegeben

XPath: Funktionen

Eingabe: ein Knoten/Wert oder eine Menge von Knoten/Werten
Ergebnis: meistens ein Wert; manchmal ein oder mehrere Knoten

- Zugriff auf Textinhalt und Knotennamen (siehe Folie 176)
- Aggregatfunktionen count(*node_set*), sum (*node_set*)

  count(/mondial/country)

  ergibt die Anzahl aller Länder.
- Kontextfunktionen (siehe Folie 182)
- Dereferenzierung (siehe Folie 185)
- Zugriff ausserhalb des aktuellen Dokuments:

  document("*url*")/*path*
- siehe W3C-Dokument *XPath/XQuery Functions and Operators*

XPath: Kontextfunktionen

- Alle Anfragen behalten die Ordnung der Elemente genau so bei, wie sie im XML-Dokument ist.
- Die position()-Funktion gibt die Position des aktuellen Knotens innerhalb der Ergebnisse einer (Teil)anfrage an.
  Als Kurzform statt [position()=x] ist [x] erlaubt; [-1] ergibt den letzten Knoten:

  //country[population > 1000000][6]

  gibt das 6. Land aus, das mehr als 1,000,000 Einwohner hat (in der Reihenfolge der Dokumentstruktur, *nicht* das sechstgrößte!)

  //country[6][population > 1000000]

  gibt das 6. Land aus, falls dieses mehr als 1,000,000 Einwohner hat.
- Die last()-Funktion gibt die Position des letzten Elementes der aktuellen Teilergebnisses – also die Mächtigkeit des Ergebnisses – an.

  //country[position()=last()]
- Beispielanfragen an hamlet.xml.

XPath: Vorwärts- und Rückwärtsachsen

- Ergebnis jeder Anfrage ist eine Folge
- Dokumentordnung (und Endergebnis): vorwärts
- Kontextfunktionen: vorwärts oder rückwärts
- alle Achsen zählen Ergebnisse *ausgehend* vom aktuellen Knoten auf.
  - Vorwärtsachsen: child, descendant, following, following-sibling
  - Rückwärtsachsen: ancestor, preceding, preceding-sibling

    //table/preceding-sibling::h4//text()

    findet alle vorhergehenden h4-Elemente (Kapitelüberschriften).

    Das Ergebnis wird -wie immer- in Dokumentordnung ausgegeben

    //table/preceding-sibling::h4[1]//text()

    findet die letzte vorhergehende Kapitelüberschrift
  - egal: self, parent, attribute (und namespace)
- im wesentlichen relevant, wenn man mit Dokumenten arbeitet.

Absolute und relative Pfade

- Pfade, die mit einem Namen beginnen, sind *relative* Pfade, die jeweils von dem Kontextknoten ausgehen (meistens innerhalb von Bedingungen) :

  //city[name = “Berlin”]
- Pfade, die mit “/” beginnen, sind absolute Pfade:

  //city[@id = /mondial/organization[name=“EU”]/@headq]

  (Semi-Join)
- ebenso Pfade, die mit “//” beginnen:

  //city[@id = //organization[name=“EU”]/@headq]
- will man das vermeiden, hat man ja noch

  //country[.//city/name=“Berlin”]

  (äquivalent: //country[descendant::city/name=“Berlin”])

  um einen relativen Pfad zu starten.

XPATH: DEREFERENZIERUNG

Zugriff über "Schlüssel'

Die Funktion id(*string*) ergibt die Elemente deren id's in *string* enthalten sind (*string* als IDREFS):

- id("D") ergibt das Element, das Deutschland repräsentiert (country/@car_code ist als ID deklariert)
- id(//country[car_code="D"]/@capital)
  ergibt den Elementknoten vom Typ city, der Berlin repräsentiert.

Auch für mehrwertige Referenzattribute (IDREFS):

- //country[car_code="D"]/@memberships
  ergibt "org-EU org-NATO ..."
- id(//country[car_code="D"]/@memberships)
  ergibt die Menge aller Elemente, die eine Organisation repräsentieren, in der D Mitglied ist.
- id(//organization[abbrev="EU"]/member/@country)
  ergibt alle Länder, die irgendeine Form von Mitgliedschaft in der EU haben.

XPATH: DEREFERENZIERUNG

Dereferenzierung durch Navigation

Syntax:

attribute::*nodetest*⇒*elementtype*

Beispiele:

- //country[car_code="D"]/@capital⇒city/name
  ergibt den Elementknoten vom Typ city, der Berlin repräsentiert.
- //country[car_code="D"]/@memberships⇒organization
  ergibt Elementknoten vom Typ organization.
- Hinweis: diese Syntax wird nicht von allem XPath-Working Drafts unterstützt:
  - XPath 1.0: nein
  - wurde ursprünglich von Quilt (2000; XQuery-Vorgänger) eingeführt
  - XPath 2.0: erste Drafts ja, später nein
  - soll später wieder kommen ...

XPath: Stringfunktionen

- concat(string, string, string*): Verkettung
- startswith(string, string)
  //city[starts-with(name,'St.')]/name
- contains(string, string)
  //city[contains(name,'bla')]/name
- substring-before(string, string, int?)
- substring-after(string, string, int?)
- substring(string, int, int): Substring i_2 Zeichen ab der i_1-ten Position

Ergänzungen in XPath 2.0

- weitere String- und Aggregatfunktionen
- komplexere Pfadkonstrukte (Alternativen, Klammerung)
  //(city|country)[name='Monaco']
  mondial/country/(city|(province/city))/name
- Erweiterter Subscript-Operator:
  //country[population > 1000000][-3]
  //country[population > 1000000][5-10]
  //country[population > 1000000][1,5-10,-3]
  (die drittletzte/fünfte bis zehnte/erste, fünfte bis zehnte und drittletzte ...)
- ANY und ALL-Semantik für *condition*:
  //country[ALL city/population > 1000000]
  //country[ANY city/population > 1000000]
  (Länder in denen alle/eine Stadt mehr als 1000000 Einwohner haben)
- letzteres geht eindeutig über übliche Navigation hinaus ...
- Abgleich der gesamten XML-Welt (XPath, XQuery) mit Datentypen (Datenmodell und XML Schema)

5.1 Einschub: Namensräume

... bekommt man erst einen Einblick, wenn man auch Anfragen stellen kann.

- Jedem Element kann ein oder mehrere Namensräume zugeordnet werden (im Datenmodell ein Knoten, ähnlich einem Attributknoten)
- damit kann man z.B. gleiche Elementnamen in unterschiedlichen Anwendungsbereichen auseinanderhalten
- momentane Situation: ein Namensraum wird mit einer URL assoziiert (keine weitergehende Semantik)
- als Attribut zum Wurzelelement (z.B. in XHTML):

```
<!ELEMENT html (head, body)>
<!ATTLIST html
  xmlns  %URI;  #FIXED 'http://www.w3.org/1999/xhtml'
>
```

  Damit ist jetzt für das HTML-Element ein (namenloser) Namensraum deklariert, der an die URL 'http://www.w3.org/1999/xhtml' gebunden ist.

... was macht man jetzt damit?

Namensräume im XML-Dokument

Gegeben:

```
<!ELEMENT html (head, body)>
<!ATTLIST html
  xmlns  %URI;  #FIXED 'http://www.w3.org/1999/xhtml'
>
```

- Eine Instanz zu dieser DTD könnte jetzt explizit so aussehen:

```
<h:html xmlns:h='http://www.w3.org/1999/xhtml'>
  <h:body> ... </h:body>
</h:html>
```

- Man braucht es aber nicht so explizit zu schreiben:
  (vgl. XHTML-Beispiel-File)

```
<html><body> ... </body></html>
```

  der Namespace gilt per Default auch für alle Subelemente
  (Intention: ist derselbe Anwendungsbereich)

Deklarationsbereich von Namensräumen

... man muss es aber wissen, wenn man Anfragen stellt:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:ht="http://www.w3.org/1999/xhtml"
                version="1.0">
```

sagt, dass der an 'http://www.w3.org/1999/xhtml' gebundene Namespace in dieser Anwendung mit "ht:" bezeichnet wird.

- Anfrage: z.B. //ht:title
- Defaultnamespaces werden an alle Subelemente vererbt,
- aber nicht an Attribute.

5.2 XPath: Fazit

Was kann XPath?

Vergleich mit den relationalen Operatoren

- Selektion: ja (Werte- und Strukturselektion)
- Projektion/Reduktion: nein. Man kann nur ganze Knoten nehmen
- Join/Verbund/Kombination: nein. Nur Semi-Joins innerhalb der Filter

Sonstige Funktionalität:

- korrelierte Subqueries: innerhalb der Filter als Semijoins
- Restrukturierung des Ergebnisses: Nein
- Nur Verfolgung eines Hauptpfades zur Navigation zu den gesuchten Elementen (Semijoins)

⇒ Nur ein Fragment einer Anfragesprache zur Adressierung von Knoten.
  - XQL (u.a. Software AG, 1998/1999) verfolgte zeitweise (als einer der Vorgänger von XPath) einen Ansatz, Joinvariablen sowie Konstrukte zur Projektion und Restrukturierung/Gruppierung in die Sprache aufzunehmen.

BEDEUTUNG VON XPATH IN DER XML-WELT

- Adressierungsmechanismus für Knoten in XML-Dokumenten
- Navigation in der Baumstruktur
- dient als Basis für verschiedene Konzepte:
  - XQuery
  - XSL/XSLT: Stylesheets, Transformationssprache
  - andere Anfragesprachen
  - XML Schema
  - XPointer/XLink