Vortrag

Hybride Speicherung von XML-Daten in RDBMS

Autor: Michael Fiedler
Betreuer: Andreas Thor

Oberseminar der Abteilung DB am 01.07.08

Inhaltsübersicht

1. **Ausgangsproblematik**
2. **Besonderheiten von PubMed**
3. **Implementierungsaspekte in DB2**
4. **Umsetzung in der Diplomarbeit**

1. Ausgangsproblematik

- **große XML-Datenmenge zu Publikationen vorhanden**
- **Einsatzfälle wie Navigation auf Daten, Data Cleaning, Zitierungsanalyse**
- **Nutzung von Abfragen in SQL wäre hierzu vorteilhaft!**
- **Überführung der XML-Daten in relationale DB**

Zwei Welten

XML (Datenformat)	SQL (Abfragesprache)
• semi-strukturiert	• strukturiert
• geordnete Folge	• mengenorientiert
• Hierarchie	
• Datenaustausch	• Backend DB für Applikationen
• daten-dokument-orientiert	• nur datenorientiert

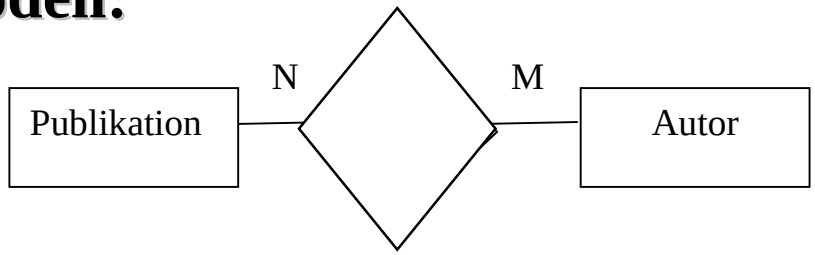
Ziel: Automatische Überführung der gegebenen XML-Daten in relationale DB

Problem: Abbildung der XML-Daten auf relational gespeicherte Daten könnte nicht verlustfrei sein.

Beispiel zu PubMed:

```
<?xml version="1.0" encoding="UTF-8"?>
<MedlineCitationSet>
    <MedlineCitation Owner="NLM" Status="MEDLINE">
        <PMID>16403481</PMID>
        <Article>
            <ArticleTitle>Are sonochemically prepared alpha-amylase protein
                          microspheres biologically active?</ArticleTitle>
            <AuthorList CompleteYN="Y">
                <Author>
                    <LastName>Avivi Levi</LastName>
                    <ForeName>S</ForeName>
                </Author>
                <Author>
                    <LastName>Gedanken</LastName>
                    <ForeName>A</ForeName>
                </Author>
            </AuthorList>
            <Language>eng</Language>
        </Article>
    </MedlineCitation>
</MedlineCitationSet>
```

ER-Modell:



SQL-Schema mit 3 Tabellen

Publikation:

PMID	ATitle	Language

Geschrieben_von:

PMID	AID	Position

Autor:

AID	LName	FName

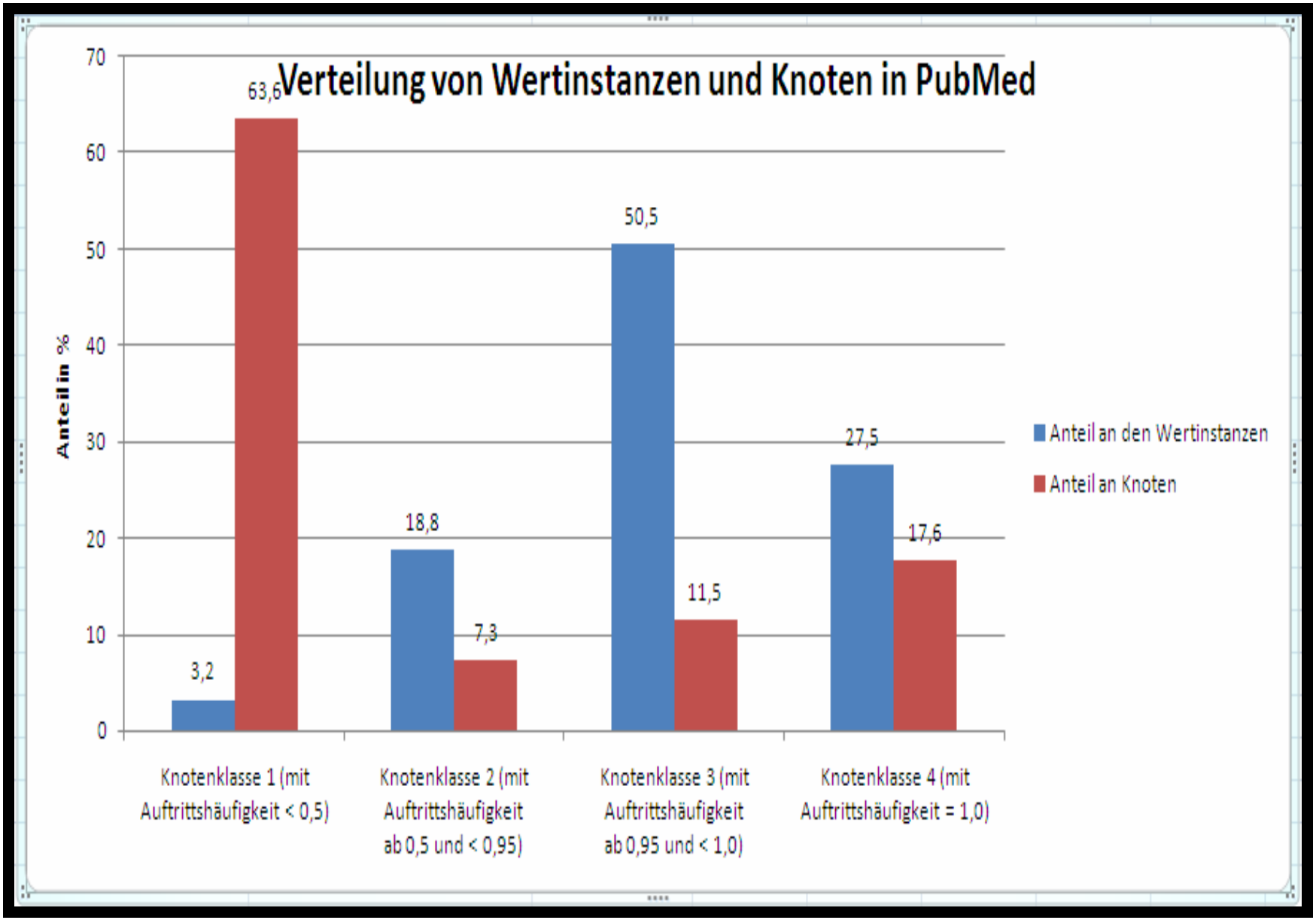
- **mehrwertiges XML-Element *Author***
- **d.h. mehrfaches hintereinander Auftreten des XML-Elements auf gleicher Hierarchieebene**
- **semantisch mit Haupt- oder Erstautor belegt**
- **bloße Reihenfolge in XML mit inhaltlicher Bedeutung durch separates Attribut *Position* in relationaler Welt abzubilden**

Fazit: Kleines XML-Dokument führt zu großem SQL-Schema mit künstlichem Primärschlüssel und zusätzlichem Attribut *Position* für die XML-Elementreihenfolge!

Grund: Informationen im Aufbau von XML-Dokumenten sind in der rationalen Welt durch separate Attribute nur repräsentierbar.

2. Besonderheiten von PubMed

- **Publikationsdaten aus den Bereichen der Medizin**
- **über langen Zeitraum erfasst**
- **Erfassungsdaten sehr umfangreich**
- **Erfassungsdaten variieren**
- **konkret:** **- Speichervolumen von 59 GB**
  - in 538 XML-Dateien
  - mit über 16 Mill. Publikationsdatensätzen
  - aus 148 XML-Elementen und 17 XML-Attributen
  - durchschnittlich 84 unterschiedliche XML-Knoten je DS
  - insgesamt 1,4*10^9 XML-Knoten in Summe


Verteilung von Wertinstanzen und Knoten in PubMed
Anteil in %
70
60
50
40
30
20
10
0
63,6
3,2
18,8
7,3
50,5
11,5
27,5
17,6
Knotenklasse 1 (mit Auftrittshäufigkeit < 0,5)
Knotenklasse 2 (mit Auftrittshäufigkeit ab 0,5 und < 0,95)
Knotenklasse 3 (mit Auftrittshäufigkeit ab 0,95 und < 1,0)
Knotenklasse 4 (mit Auftrittshäufigkeit = 1,0)
Anteil an den Wertinstanzen
Anteil an Knoten

2.1. XML-Schemaevolution

- **Schemaerweiterung mit neuen XML-Elementen und Attributen in den Rohdaten**
- **Unterscheidung zwischen Basis- und Zusatzdaten sinnvoll**
- **Basisdaten: von Anfang an erfasst und / oder mit sehr großem relativem Anteil präsent je Datensatz**
- **Zusatzdaten: liegen nur für einen Teil an Datensätzen vor, weil sie entweder erst seit kurzer Zeit oder nur optional erfasst wurden**
- **neue XML-Elemente und Attribute sehr wahrscheinlich**

2.2. Speicherungsmöglichkeiten

Zwei grundsätzliche Möglichkeiten: Anwendungsspezifische Speicherung (AWS) oder Generische Speicherung (GS)

AWS

- **jedes XML-Element ohne Unterelemente als relationales Attribut**
- **jedes XML-Element mit Unterelementen als relationale Tabelle**
- **jedes XML-Attribut als relationales Attribut in Tabelle von zugehörigen XML-Element**
- **mehrwertige XML-Elemente als 1:N bzw. N:M Beziehung von Tabellen mit künstlichem Schlüsseln abbilden**

GS

- XML-Element bzw. XML-Attribut wird nicht 1:1 auf relationales Attribut abgebildet
- Stattdessen werden eine Vielzahl von verschiedenen XML-Elementen bzw. XML-Attributen oder XML-Sequenzen auf eine Spalte in relationaler Tabelle abgebildet
- Beispiel für GS in relationaler Tabelle:

PMID	AttributID	Name	Wert	Vater

Vergleich zwischen AWS und GS als Einzellösung

Kriterium	AWS	GS
Schemagröße	sehr umfangreich und mit Tendenz unübersichtlich	klein, mit zwei Tabellen (Attributs- und Beziehungstabelle)
Schemaevolution	vollständig in relationaler Welt	keine
NULL-Werte	alle gespeichert	keinen gespeichert
Anfrageerstellung	domänenspezifisch über FROM-Klausel, inhaltlich verständlich	sehr umfangreiche WHERE-Klausel, inhaltlich kompliziert und schwer verständlich
Anfrageausführung	z.T. Joins über mehrere Tabellen è schneller	z.T. sehr viele Joins nötig è langsamer

Fazit: AWS- oder GS-Lösung

- **benutzerunfreundlich**
- **Einarbeitungsaufwand bzw. Bearbeitungs-aufwand sehr hoch**

→ **Realisierung AWS oder GS genügen nicht.**

Hybrider Ansatz mit AWS und GS zur Reduktion der Schemagröße und Anfragekomplexität erforderlich.

2.3. Entscheidung über AWS oder GS für XML-Daten

- **keine 100% sondern hybride Lösung**
- **zentrale Fragestellung:**

> **Welche Daten sind generisch und welche anwendungsspezifisch zu speichern?**

- **Idee: Häufigkeit der XML-Knoten und deren Beziehungen nutzen**
- **Entscheidung mittels statistischer Analyse**

Statistische Analyse

Statistische Werte und deren Interpretation

Erfassung aller XML-Elemente sowie deren Attribute mit:

- Pfad: Angabe der Vater-Sohn-Beziehung sowie eines eindeutigen Namens über allen XML-Elementen und XML-Attributen
- durchschnittliches Vorkommen eines Knotens je DS: Schwellwert 1
- Max je DS: Erfassen von Mehrwertigkeit
- Min je DS: Erfassen von NULL-Werten
- absolute Anzahl der Knoteninstanzen im Datensatz: zur Berechnung von Schwellwert 2
- Absolute Anzahl der Datensätze mit einem Knoten: zur Berechnung von Schwellwert 1
- durchschnittliches Vorkommen der Knoteninstanzen im Datensatz: Schwellwert 2

Vier Beispiele aus PubMed

	Title	**Abstract**	**Author**	**Language**
Min	1	0	0	1
SW1/SW2	1.0/1.0	0.515/1.0	0.978/3.162	1.0/1.002
Max	1	1	744	4
Speicherung	AWS	GS	AWS	AWS & GS
Mehrwertigkeit	keine	keine	ja	ja
Beziehung	---	---	N:M	Language1 AWS in Basistabelle und andere(n) GS

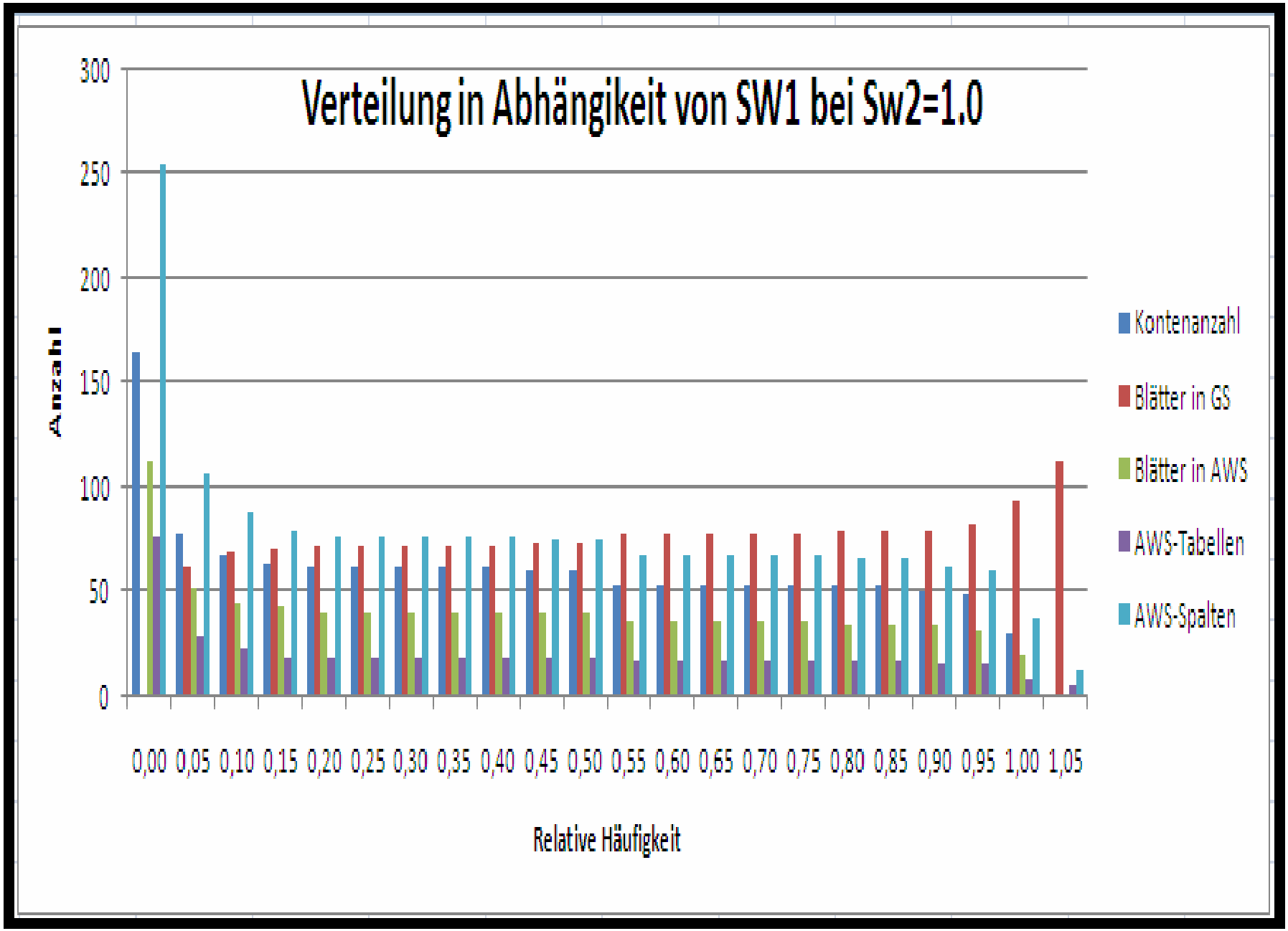
2.4. Umgang mit PubMed-Daten

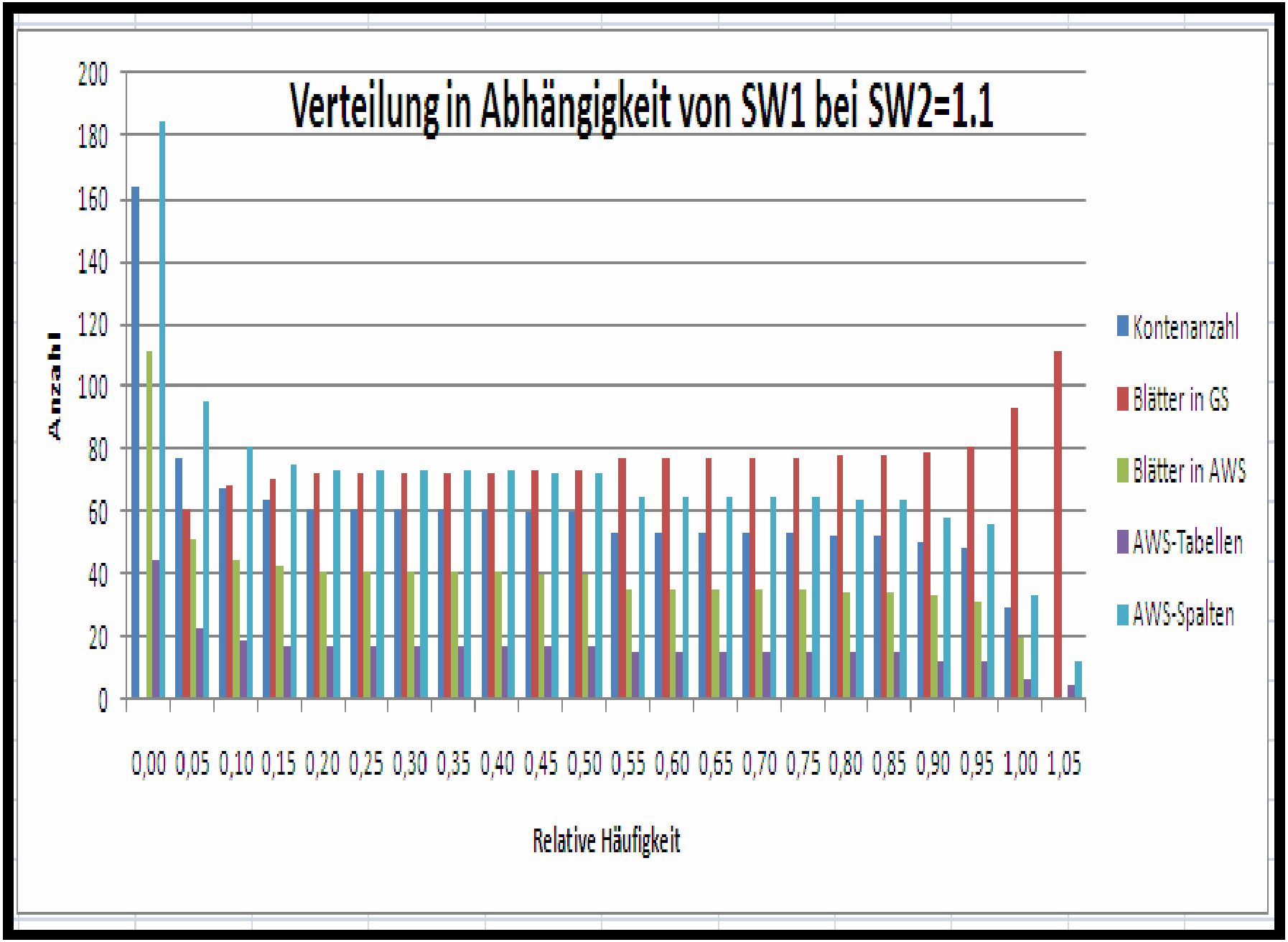
- **XML-Daten ohne Schema als Input**
- **relationale Speicherung auf Nutzungsaspekt wegen Datenvolumen abstimmen**
- **statistische Auswertung vorab nötig**
- **SQL-Schemagenerierung mit hybrider Speicherung**

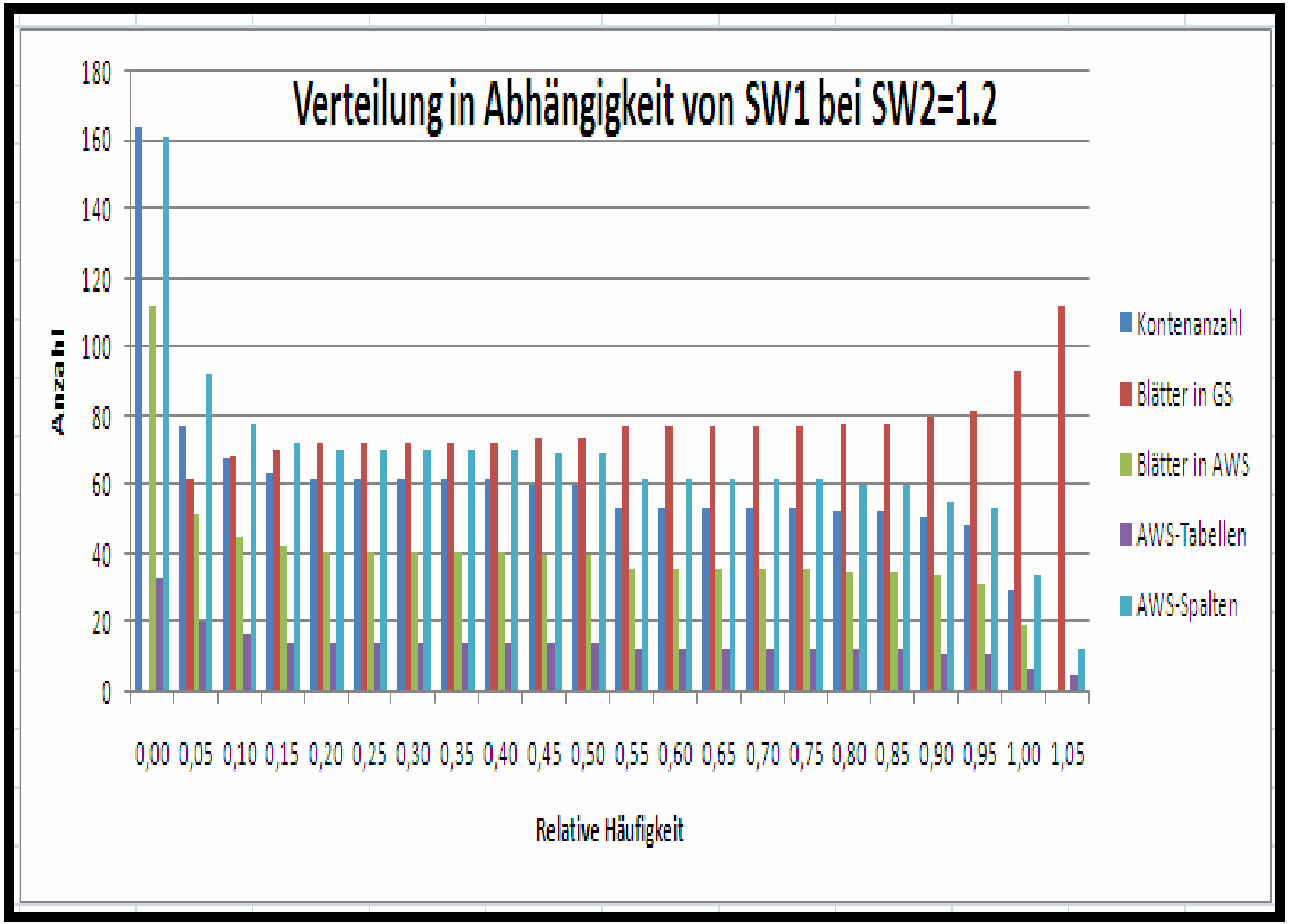
Lösungsidee:

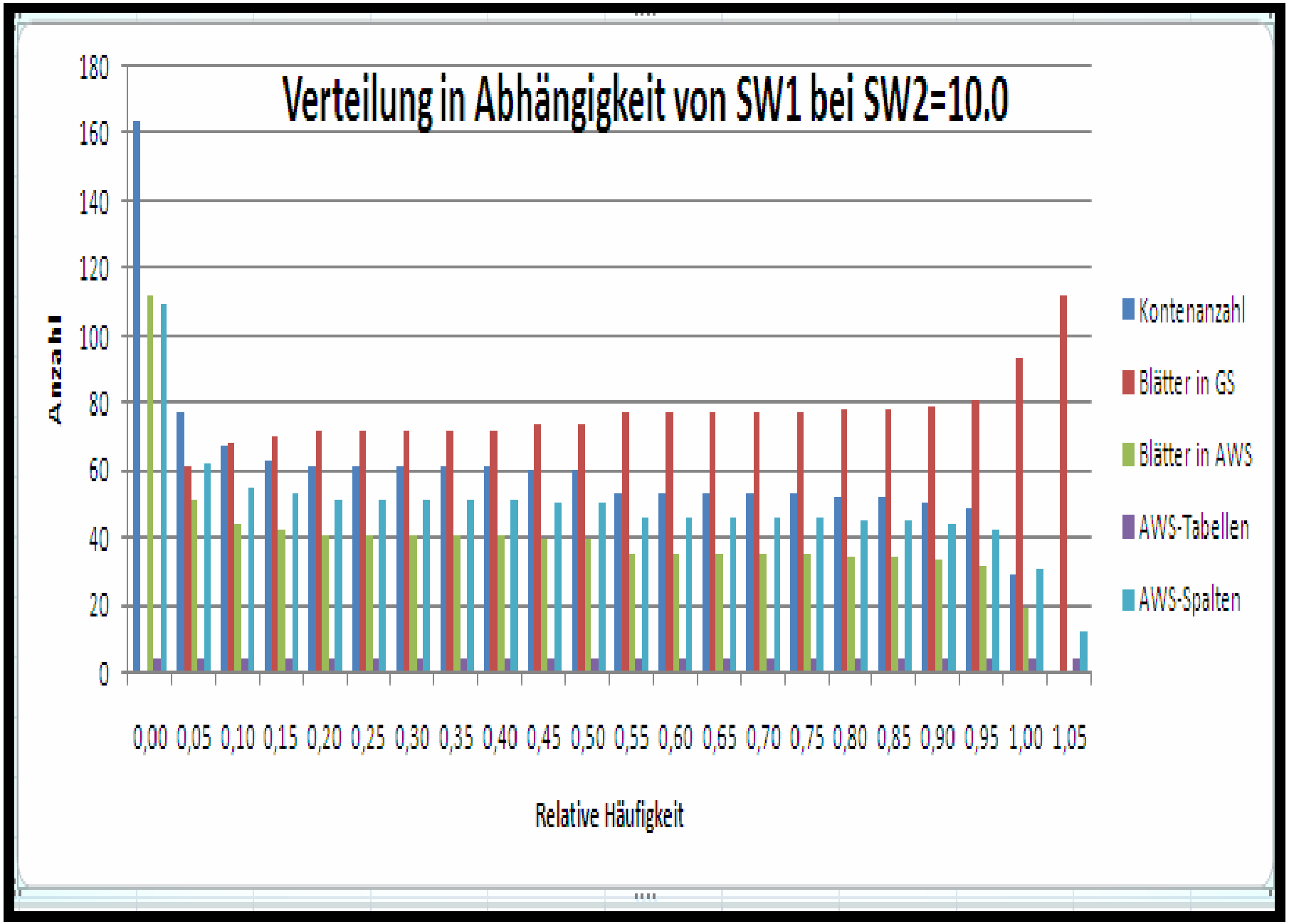
- statistische Analyse
- benutzerdefinierte Schwellwerte und Namen für AWS und GS
- bei Default für Attribut mit 0.95 und für Beziehung 1.2

- automatischer Import und Schemagenerierung
- generische Speicherung für alle XML-Elemente und Attribute unterhalb eines Schwellwerts bzw. für die durch den Benutzer festgelegten

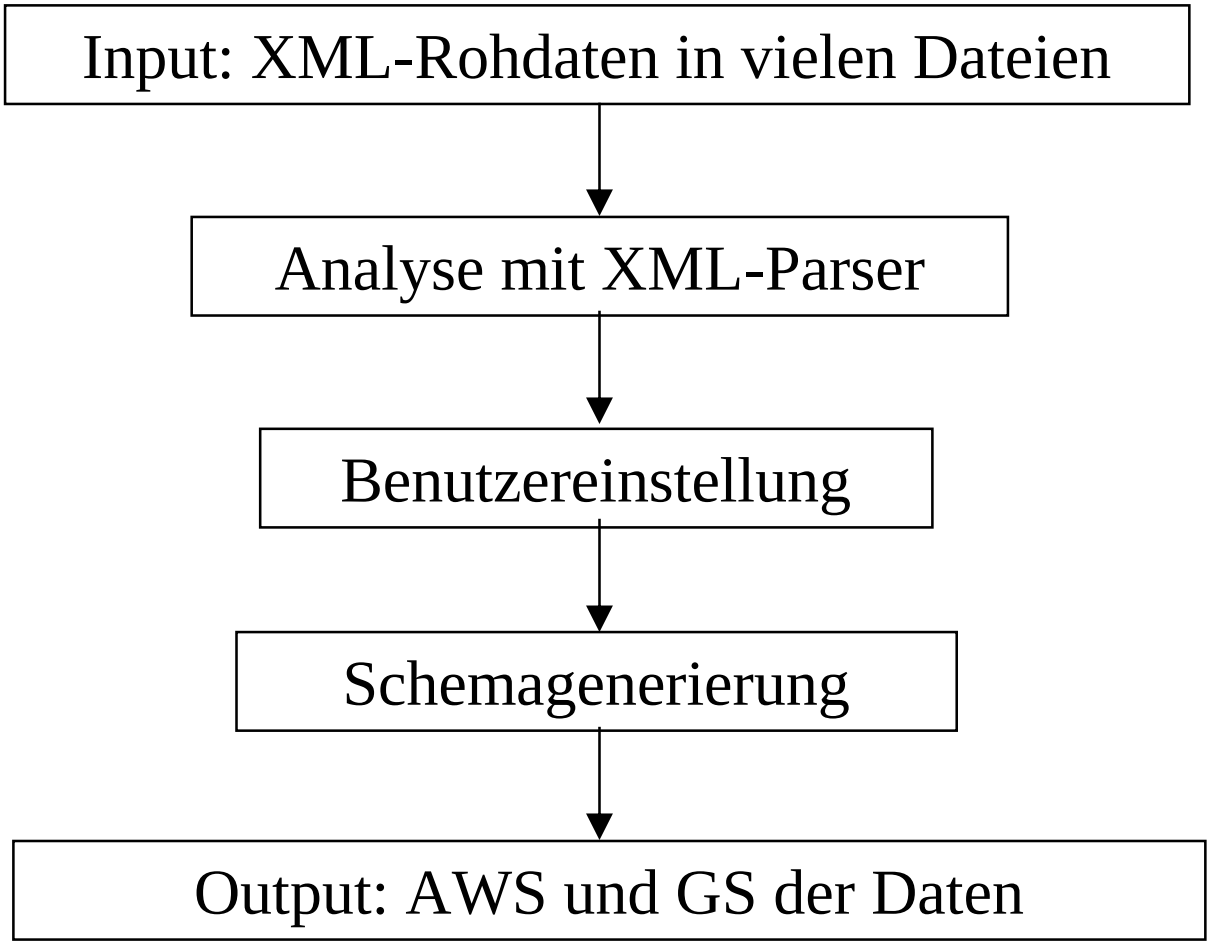

Verteilung in Abhängigkeit von SW1 bei Sw2=1.0
Anzahl
300
250
200
150
100
50
0
0,00 0,05 0,10 0,15 0,20 0,25 0,30 0,35 0,40 0,45 0,50 0,55 0,60 0,65 0,70 0,75 0,80 0,85 0,90 0,95 1,00 1,05
Relative Häufigkeit
Kontenanzahl
Blätter in GS
Blätter in AWS
AWS-Tabellen
AWS-Spalten


Verteilung in Abhängigkeit von SW1 bei SW2=1.1
Anzahl
200
180
160
140
120
100
80
60
40
20
0
0,00 0,05 0,10 0,15 0,20 0,25 0,30 0,35 0,40 0,45 0,50 0,55 0,60 0,65 0,70 0,75 0,80 0,85 0,90 0,95 1,00 1,05
Relative Häufigkeit
Kontenanzahl
Blätter in GS
Blätter in AWS
AWS-Tabellen
AWS-Spalten


Verteilung in Abhängigkeit von SW1 bei SW2=1.2
Anzahl
180
160
140
120
100
80
60
40
20
0
0,00 0,05 0,10 0,15 0,20 0,25 0,30 0,35 0,40 0,45 0,50 0,55 0,60 0,65 0,70 0,75 0,80 0,85 0,90 0,95 1,00 1,05
Relative Häufigkeit
Kontenanzahl
Blätter in GS
Blätter in AWS
AWS-Tabellen
AWS-Spalten


Verteilung in Abhängigkeit von SW1 bei SW2=10.0
Anzahl
180
160
140
120
100
80
60
40
20
0
0,00 0,05 0,10 0,15 0,20 0,25 0,30 0,35 0,40 0,45 0,50 0,55 0,60 0,65 0,70 0,75 0,80 0,85 0,90 0,95 1,00 1,05
Relative Häufigkeit
Kontenanzahl
Blätter in GS
Blätter in AWS
AWS-Tabellen
AWS-Spalten

Ablauf:


Input: XML-Rohdaten in vielen Dateien
Analyse mit XML-Parser
Benutzereinstellung
Schemagenerierung
Output: AWS und GS der Daten

3. Implementierungsaspekte

- **Umsetzung in drei Schritten: Analyse, Schemagenerierung, Zerlegung mit Befüllung der Zieltabellen**
- **XML-Dokumente direkt in relationalen Tabellen speichern**
- **Zerlegungsabfragen auf XML-Dokumente in relationaler Tabelle**

3.1. *pure*XML

- **neuer Datentyp *XML* gestattet direktes Speichern von XML-Dokumenten in relationaler Tabelle**
- **Zugriff auf XML-Daten über vier Abfragearten:**
  1. **mit reinem SQL vollständiges XML-Dokument abfragbar**
  2. **SQL/XPath auf ausgewählten XML-Dokumenten sind einzelne XML-Werte abfragbar**
  3. **XPath/SQL ausgewählte XML-Werte werden mengenwertig abgefragt**
  4. **mit rein XPath einzelne XML-Werte abfragen**

- **Abfrageart 2 – Kombination von SQL- und XPath-Abfrage für Dekomposition**
- **Abfrage mit SQL-Rumpf und eingelagerter Funktion mit XPath-Abfrage**
- **direkte Abfragen von XML-Daten mit neuen Funktionen in DB2 V9.1.2**

3.2. Dekomposition mit annotierten XML-Schema (XSR)

- **Voraussetzung: XSR gegeben**
- **elegante und detailreiche Zerlegung möglich**
- **Annotationen vorab in XSR einbringen**
- **XSR mit Annotationen in DB2 registrieren und zur Dekomposition freigeben**

3.3. Einfache Form der Dekomposition

- **ohne XSR**
- **SQL/XPath-Abfrage mit SQL-Tabellenfunktion *XMLTable***
- **SQL-Tabellenfunktion operiert mit XPath und Inputfunktionen (SQLQuery, XMLColumn) bzw. Passing-Klausel**

	Einfache Form der Dekomposition	Dekomposition mit annotiertem XSR
Vorteile	§ einfache Realisierung bzw. automatische Generierung der Zerlegungsabfragen § XPath 2.0 unterstützt	§ Befüllung aller Zieltabellen in einem Schritt § einfache Realisierung von generischer Speicherung
Nachteile	§ für jede Zieltabelle separate Abfrage § x-facher Durchlauf der XML-Daten für Zerlegung zur Laufzeit § generische Speicherung aufwendig	§ Erstellung der Annotationen sehr aufwendig § Blockierung aller Zieltabellen der Zerlegung

Beispiel mit SQL-Tabellenfunktion XMLTable

- XMLTable erzeugt eine relationale Tabellensicht auf die XML-Daten.
  Output von XMLTable direkt relational speicherbar mit INSERT INTO
- Aufbau der Abfrage:

```
[INSERT INTO Zieltabelle ( )]
SELECT X.*
FROM Bezugstabelle_mit_XML-Spalte AS BT
        XMLTABLE ('BTXS.//Wurzelelement'
        PASSING BT.XMLSpaltenname AS BTXS
        COLUMNS
        "Expliziter Spaltenname" relationaler Datentyp
                                                PATH '//aktueller XML-Knoten'
        …
        ) AS X;
```

Zerlegungsabfrage mit Speicherung in Zieltabelle *Author*

Author:

AID	LName	FName

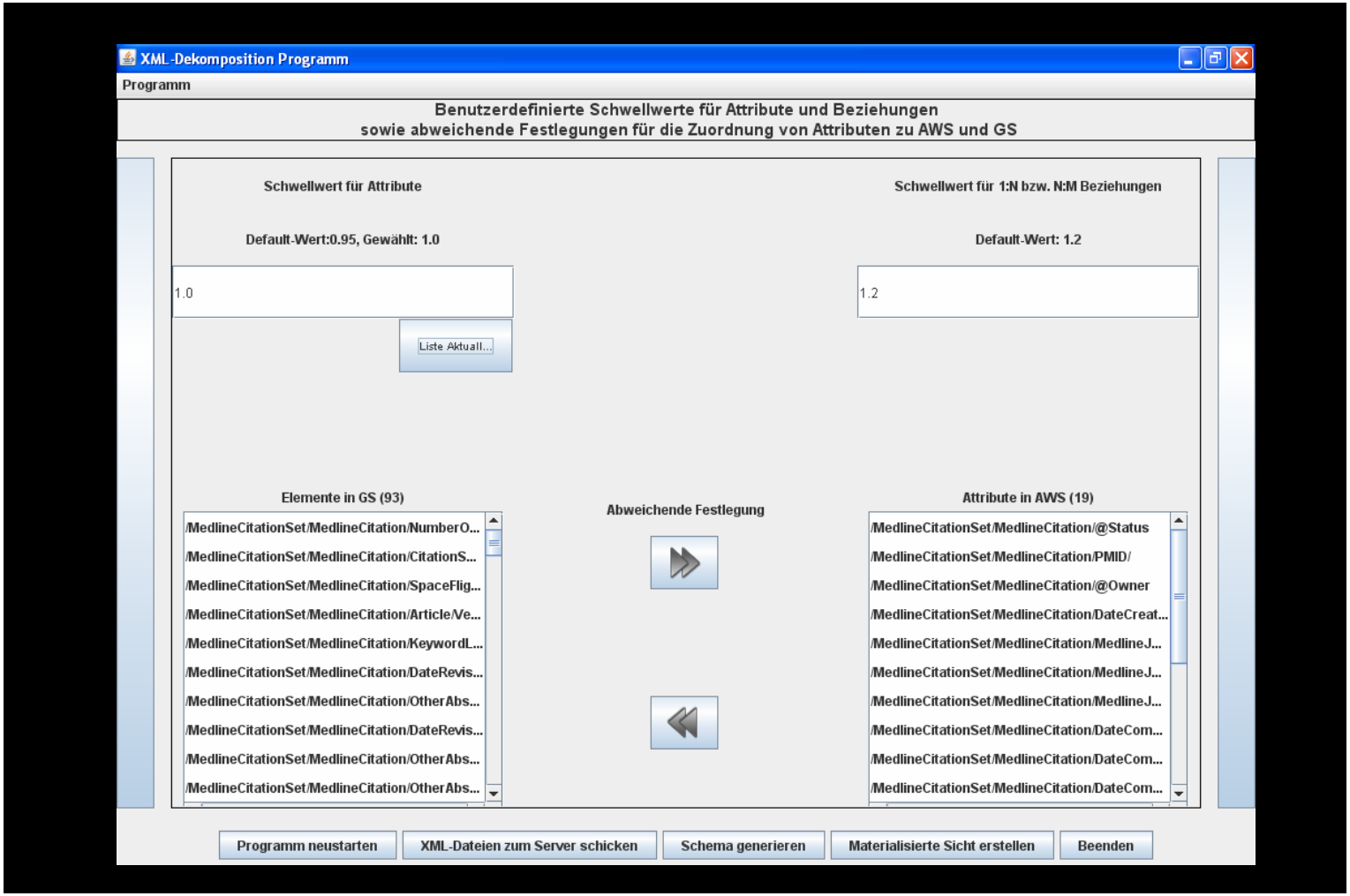
```
INSERT INTO Author (LName, FName)
SELECT DISTINCT X.*
FROM pureXMLTabelle AS XT
    XMLTABLE ('XTS. // AUTHOR'
    PASSING XT.XML-Spaltenname AS XTS
    COLUMNS
            "LName" VARCHAR (100) PATH './LastName'
            "FName" VARCHAR (100) PATH './ForeName'
            ) AS X;
```

Verbesserung der einfachen Form der Dekomposition

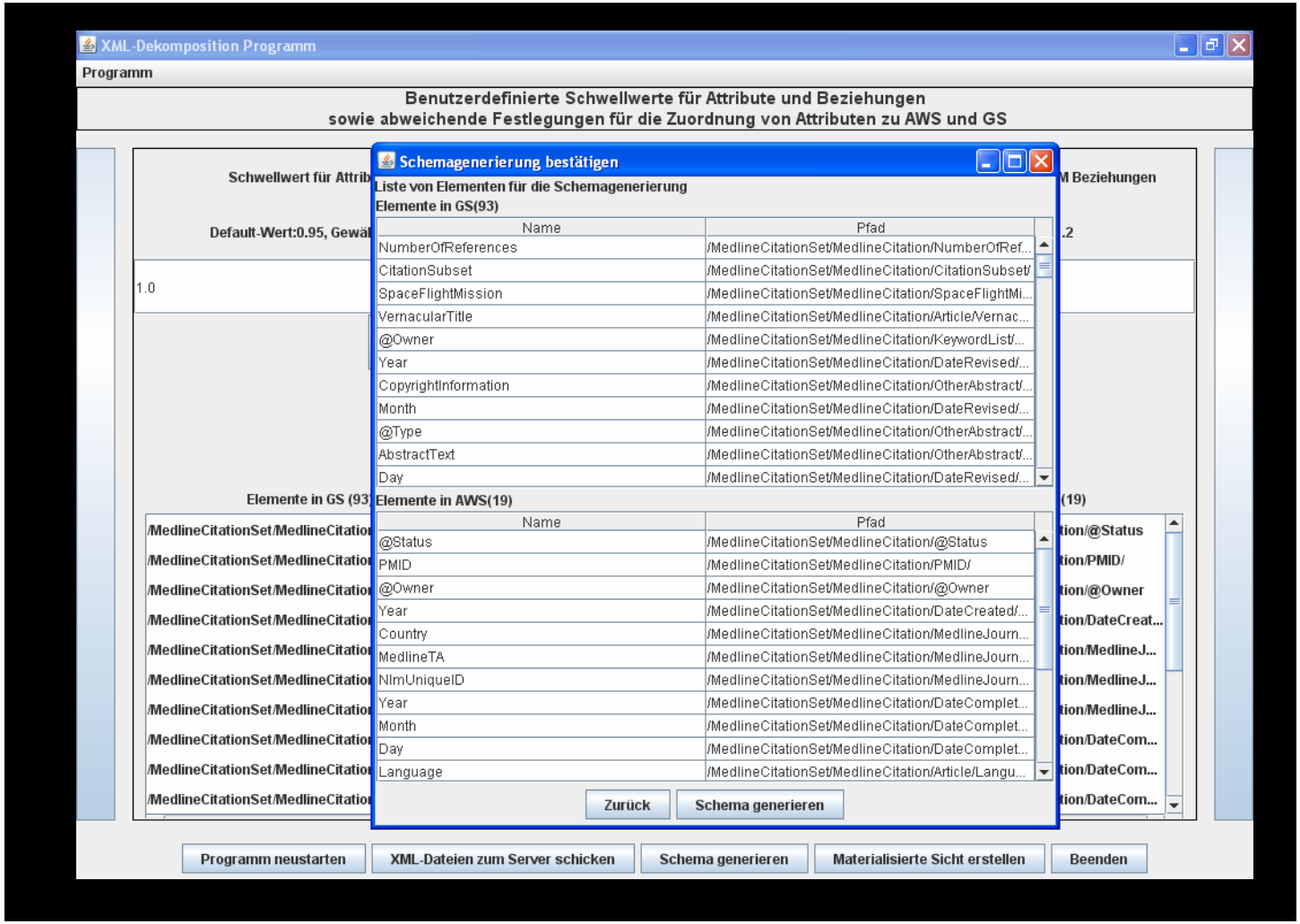
- Optimierung der einfachen Form der Zerlegung über vorgelagerte Aufbereitung der XML-Daten
- erneute *pure*XML-Speicherung für optimierte XML-Dokumente
- separate XML-Dokumente für generische Speicherung und direkter relationaler Abbildung auf Attribute

4. Umsetzung in der Diplomarbeit

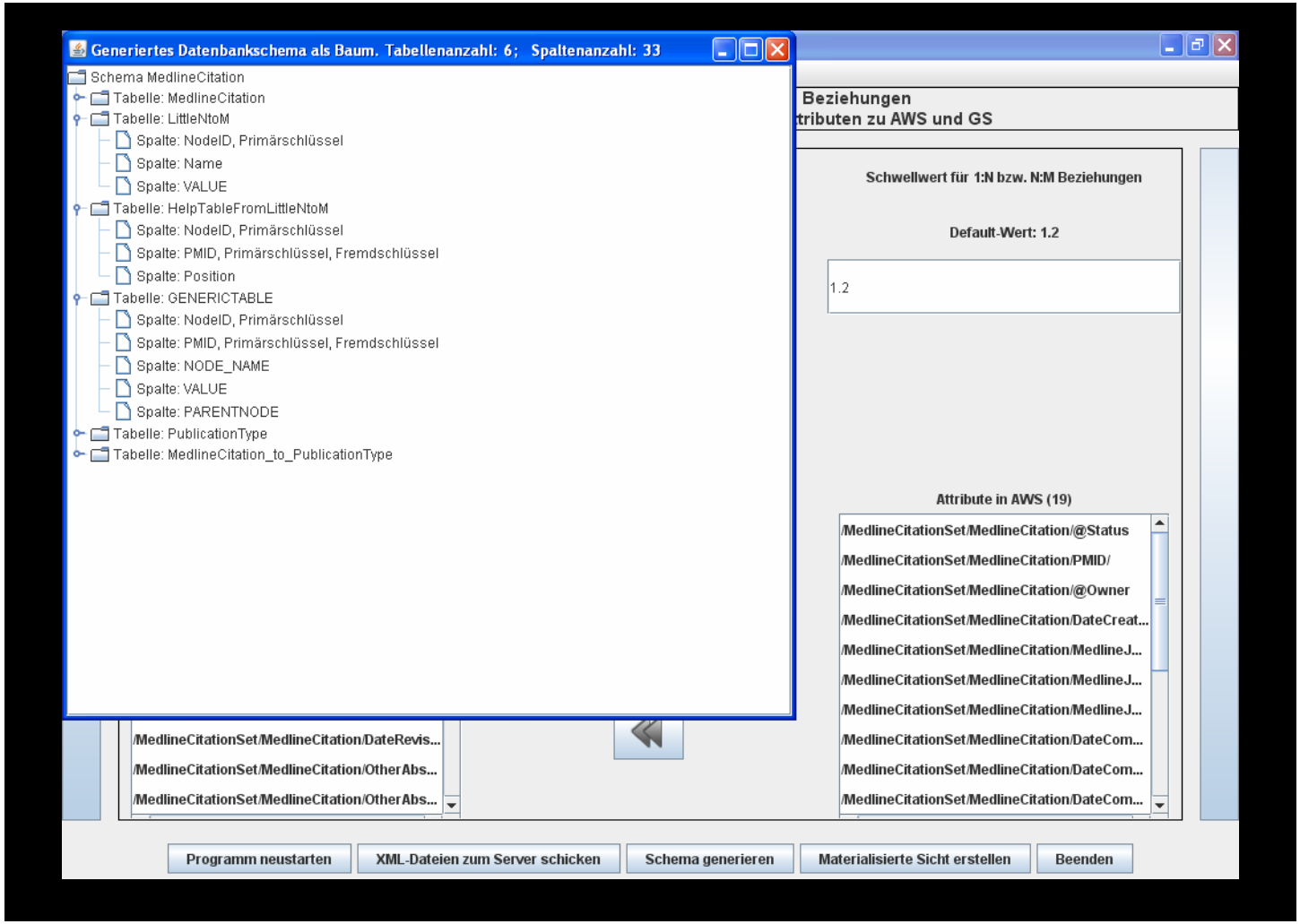
Hauptformular: Schwellwerte



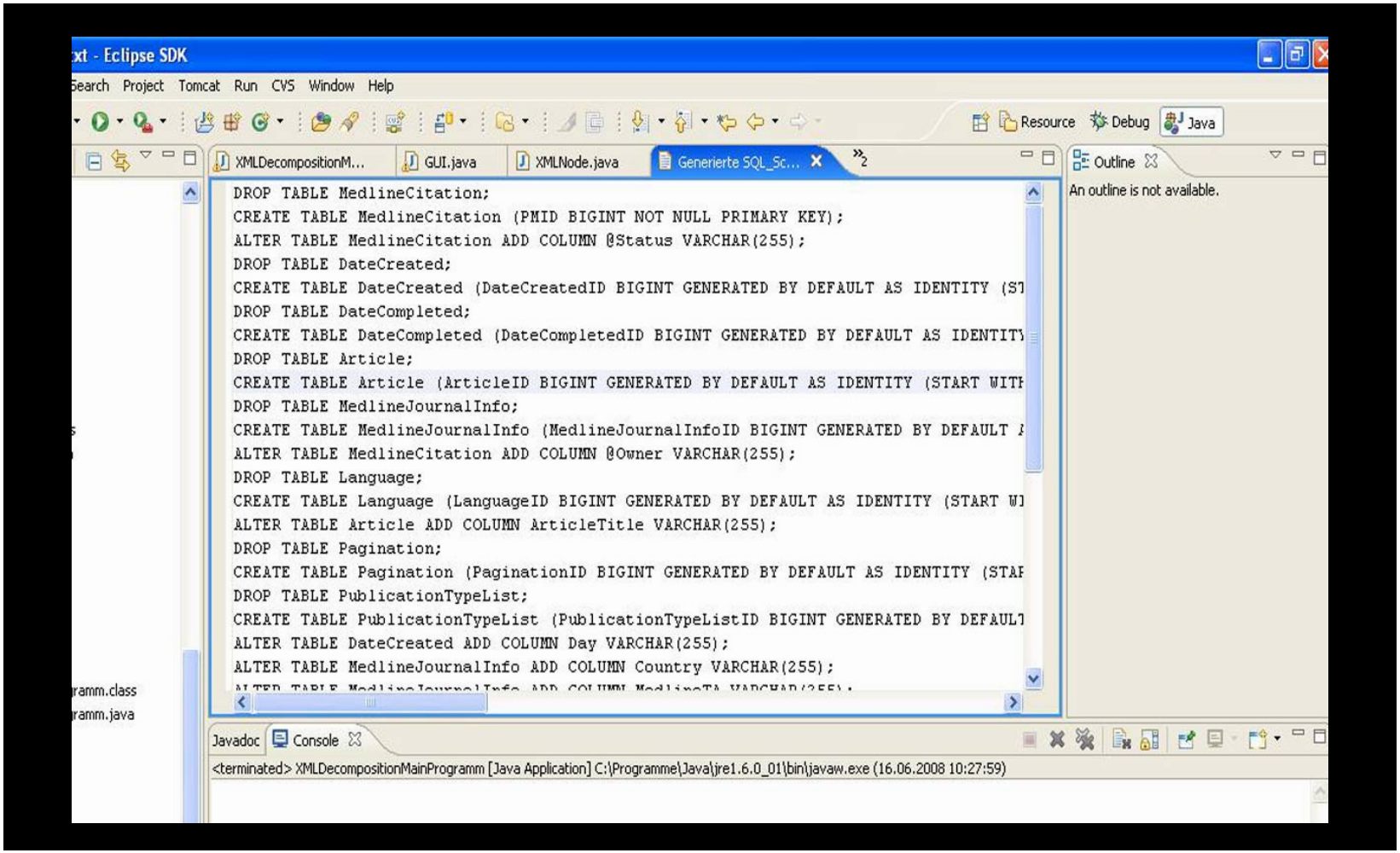
Fenster: Namensänderung



Fenster: Datenbankschema in Baumdarstellung



Tabellengenerierung



Zusammenfassung

- hybrider Ansatz von AWS und GS:
  - basierend auf statistischer Analyse
  - mit kleinem (überschaubarem) Schema, ohne Schemaevolution sowie Anfrageformulierung zu vereinfachen und Anfrageausführung zu verbessern
- einfache Form der Dekomposition auf aufbereiteten XML-Dokumenten
- PubMed-Daten bei Schwellwert 0.95 für Attribute und 1.2 für Beziehungen -> nur 9 Tabellen für AWS und 1 für GS
- nur 31 Blattknoten in AWS
- ggf. mat. Sichten für häufige Abfragen