Seminar Simpl

Thema:Datenmodelle und Datenbanksprachen

Tu,Xi

Inhaltverzeichnis

1 Motivation,Einführung.

- Das Ziel des Studienprojekts SIMPLs ist es ein erweiterbares, generisches Rahmenwerk zu erstellen, welches ermöglicht, beliebige Datenmanagement-Funktionalitäten zum Zugriff auf beliebige Datenquellen in BPEL einzubinden. Es soll weitere Möglichkeiten zur Verarbeitung von Daten in wissenschaftlichen Workflows anbieten, insbesondere die Möglichkeit, konkrete Datenmanagement-Funktionalitäten auf einfache Weise zu entwickeln und als Plug-ins in das Rahmenwerk zu integrieren.
- In unserem Projekt SIMPL müssen wir große sowie heterogene Datenmengen verarbeiten. Datenbanken spielen hier eine wichtige Rolle. Jede DB basiert auf einem Datenmodell, welches Regeln definiert. Durch die Regeln können die Objekte der DB erzeugt und verändert werden .[1]Hier benutzen wir ein Relationsmodell. Im Folgenden wird deshalb versucht, die folgenden 4 Bereiche zu erklären: die grundlegende Struktur, Operationen von Relationsdatenbanken und die SQL-Sprache. Aber auch XQuery von Xml, damit man bei dem Projekt Erfahrung hat, wie man die heterogene Datenquelle bearbeiten kann.
- Ich werde im Kapitel 2 über das relationale Modell, in Kapital 3 über die Relationsoperationen und im Kapital 4 über die SQL-Sprache schreiben. Im Kapitel 5 wird ein Überblick über XPath und im Kapitel 6 ein Überblick über die XQuery Sprache gegeben.
- Durch solche Erklärungen wird verständlich, wie man die Daten im Rahmenwerk des relationalen Datenmodells organisieren, manipulieren und kontrollieren soll.

2 Relationales Datenmodell

- Dieses Kapital basiert hauptsächlich auf [1]. Es handelt sich die Form und ein paar grundlegenden Definitionen von relationales Datenmodell.

2.1 Einführung

- Man kann eine Relation im relationalen Datenmodell durch eine Tabelle R(A1,A2...An) darstellen , wobei R der Name der Relation ist, A1 bis An die Attribute sind. Jedes Attribut besetzt eine Spalte in der Tabelle, und ist von einem Definitionsbereich D beschränkt. Deshalb beschreibt man genaue Notaion für einen relationales Datenmodell durch R(A1:D1,A2:D2...An:Dn), R ist eine Untermenge von kartesischen Produkt aller Definitionsbereichen der Spalten. Eine/Ein Zeile/Tupel ist ein Element einer Relation.Eine Relation ist

eine Menge,d.h. die Tupel müssen eindeutig sein. Die Eindeutigkeit der Zeilen/Tupel kann durch Primärschlüssel oder ggf. mehrere Schlüsselkandidaten garantiert werden.

- Beispiel 2.1:
  Hier ist die MatrNr der Primärschlüssel. Student ist Tabellenname. Alle fünf Spalten sind Attributen. Jede Zeile ist gleichzeitig auch ein Tupel. Dabei sind Domänen von MatrNr.. Semester und PLZ Integer, Domänen von Name und Wohnort sind Char.

Student

MatrNr	Name	Semester	PLZ	Wohnort
1001	Andy	6	70100	Stuttgart
1002	Rene	7	70569	Stuttgart
1003	Stephen	7	80100	München
1004	Martin	6	80100	München

2.2 Fremdschlüssel

- Wenn es ein Attribut oder eine Attributkombination von Relation A gibt, seine jeder Wert von dem Attribut oder der Kombination ein gleicher Wert in der Realtion B hat, und das Attribut oder die Kombination in B ist ein Primärschlüssel oder ein Schlüsselkandidaten von B, dann sagen wir, dass der Attribut oder die Attributkombination von A ein Fremdschlüssel bzgl. B ist.
- Beispiel 2.2:

Prüfung

VorlNr	MatrNr	Datum	Note
2520	1002	01.04.2007	1.7
2628	1003	01.03.2007	2

Hier ist das Attribut MatrNr Fremdschlüssel bzgl. der Relation Student aus Beispiel 2.1.

3 Relationale Operationen

- Relationenalgebra besteht aus einer nicht leeren Menge von Objekten und einer dazugehörigen Familie von Operationen. Das ist eine Algebra im allgemeinen Sinne. Die Relationenalgebra ist ein Beispiel dafür und bei ihr gibt es dann die folgenden Operationen. Dabei bestehen die Operationen aus

Klassischen Mengenoperationen und Relationenoperationen.

- Im Kapital handelt es sich um vier klassische Mengenoperationen und vier wichtig Relationenoperationen. Ich werde durch einige Beispiele genau erklären, wie die Relationenoperationen funktionieren.

- Dieses Kapital basiert hauptsächlich auf [1].

3.1 Klassische Mengenoperationen:

- Klassische Mengenoperationen handeln sich zwei Objektmengen, die Beide gleicher Grad und Vereingungsverträglichkeit der beteiligten Relationen haben müssen.

  Vereinigung: $R \cup S$

  Differenz: $R - S$

  Durchschnitt: $R \cap S = R – (R – S)$

  Symmetrische Differenz: $R \; X \; S = (R \cup S) – (R \cap S)$

3.2 Relationenoperationen

- Relationenoperationen schließen die folgenden Operationen ein:

  Restriktion(Selektion) : σ

  Projektion : π

  Erweitertes kartesisches Produkt: X

  Verbund (Join) : join

3.2.1 Selektion

- Die Selektion ist eine Auswahl von Zeilen einer Relation R über ein Prädikat p

  $\boldsymbol{\sigma}p(R)=\{t|t \in R \wedge P(t)\}$

- Beispiel 3.1:

  $\boldsymbol{\sigma}$semester=6 $\wedge$ wohnort=Stuttgart bei der Relation Student.

Dieser Ausdruck bedeutet, dass man solche Elements aussuchen wollte, sein Werte der Attributen Semester gleichen 6 und Wohnort gleichen Stuttgart.

Ergebnisse: Erste Tupel/Zeile

MatrNr	Name	Semester	PLZ	Wohnort
1001	Andy	6	70100	Stuttgart
1002	Rene	7	70569	Stuttgart
1003	Stephen	7	80100	München
1004	Martin	6	80100	München

3.2.2 Projektion

- Die Projektion ist eine Auswahl von Spalten (Attribute) aus einer Relation R. Durch die ausgewählten Spalten wird eine neue Relation aufgebaut.
- Beispiel 3.2:

πPLZ,Wohnort(Student) oder π 4,5(Student).

Durch diesen Ausdruck kann man die gewünschten Spalten PLZ und Wohnort

auswählen, und eine neue Relation erstellen.

Das Ergebniss ist:

Neu

PLZ	Wohnort
70100	Stuttgart
70156	Stuttgart
80100	München

Hier muss man aufpassen,nachdem man projektion benutzt,wenn es bei der neue Relation gleiche Zeilen gibt,dann muss die Zeilen nur eine behalten werden. Das hißt Duplikateliminierung. Wenn es gleiche Zeilen gibt, dann gibt es keine Primärschlüssel, dann ist das keine Relation.

3.2.3 Erweiteres kartesisches Produkten

- Beispiel 3.3:
- Dieses Beispiel zeiget, wie Tabelle Teammate mit Tabelle Status ein Aktivität von erweiteren kartesischen Produkten durchgeführt ist, und eine neue Tabelle erstellt.

Teammate

PerNr	Name	Wohnort
1001	Beckenbauer	Stuttgart
1002	Müller	München

Status

Name	Höhe	Gewicht
Beckenbauer	183	85
Müller	163	70

Temmate x Status

PerNr	Name	Wohnort	Name	Höhe	Gewicht
1001	Beckenbauer	Stuttgart	Beckenbauer	183	85
1001	Beckenbauer	Stuttgart	Müller	163	70
1002	Müller	München	Beckenbauer	183	85
1002	Müller	München	Müller	163	70

3.2.4 Verbund

- Verbund wählt aus der katesischen Produkten von zwei Relationen die Tupeln aus, die bestimmte Prädikaten erfüllen.
- Seien R und S Relationen, $\Theta \in \{<, =, >, \leq, \neq, \geq\}$ (arithem. Vergleichsoperator),A Attribut von R und B Attribut von S. Θ-Verbund zwischen R und S:
- $V=(R \text{ join } S)= \sigma\ A\Theta B\ (R \times S)$

Drei Rechenschritten: 1 die beiden Attributen können vergleichen;

2 beiden Relationen machen erweitertes Kartesisches Produkt;

3 durch Vergleichensoperation macht man Selektion.

- Beispiel 3.4:
- Dieses Beispiel zeigt, wenn die Wert des Attributs C von R kleiner als die Wert des Attributs E von S ist, dann verbinden die Beiden. Und Eine neue Relation ist erstellt.

R

A	B	C
A1	B1	5
A1	B2	6

S

B	E

B1	3
B2	7
B3	10

R join S
C<E

A	R.B	C	S.B	E
A1	B1	5	B2	7
A1	B1	5	B3	10
A1	B2	6	B2	7
A1	B2	6	B3	10

- Beispiel 3.5:
- Wenn hier Vergleichensoperation „=“ ist, dann ist dieser Verbund Gleichverbund.

A	R.B	C	S,B	E
A1	B1	5	B1	3
A2	B2	6	B2	7

3.2.5 Natürlicher Verbund

- Wenn es bei zwei Tabelle gleiche Spalten gibt, dann dürfen die zwei Tabelle natürlicher Verbund durchführen, und nach natürlichen Verbund darf nur Eine von der beiden Spalten behalten.

- Beispiel 3.6:

  Man führt hier R und S natürlicher Verbund durch, dabei haben die beiden Attributen B von R und S gleiche Wert, deshalb bleibt nur eine Spalte von Beiden übrig.

A	B	C	E
A1	B1	5	3
A1	B2	6	7

4 Structured Query Language SQL

- Mit der kann man die Funktionen, die ausgeführt werden sollen, ausdrücken. Früher teilt man Datenbanksprachen Sprache DDL für die Definition von Daten,DML für die Manipulation von Daten, DCL für die Kontrolle von Daten auf. In der Datenbansprache SQL sind alle diese drei Sprachen vereint.
- Durch SQL kann man Schema, Tabelle, View, und Index definieren und kontrolliern. Aber seine kern Bereich ist, Daten anzufragen. Select Satz hat umfangreiche Funktionen.
- Dieses Kapital basiert hauptsächlich auf [1].

4.1 SQL-Anfragen:

- Form des SELECT Statements:

  select-exp ::= **SELECT** [**ALL** | **DISTINCT**] select-item-commalist
  FROM table-ref-commalist
  [**WHERE** cond-exp]
  [**GROUP BY** column-ref-commalist 1][**HAVING** cond-exp]
  [**ORDER BY** column-ref-commalist 2[ASC|DESC]

- SELECT * bedeutet, gibt die ganze Tupel aus mit aller Attributen von Tabelle;

  FROM-Klausel: Aus welcher Tabelle sind die Attributen ausgewählt
  Where-Klausel: durch Prädikate beschräkt die gewählten Attributen, dann kann die gewünschten Tupel ausgewählt werden

- Beispiel 4.1:
  Man wollte alle Zeilen, die Semester=6 und Wohnort=Stuttgart sind, aussuchen.

  SELECT *
  FROM Student;
  WHERE (Semester=6) **AND** (Wohnort=Stuttgart)

- Beispiel 4.2:

  Man wollte aus der neue Relation, die zuerst durch Natürlich Join Tabelle Student und Prüfung verbindet, danach durch Projektion Splten 3,5 aus der verbindeten neue Tabelle erstellt ist, durch Beschränkungen Semester=7 und Wohnort=Stuttgart die Zeilen aussuchen.

  SELECT *
  FROM π3,5 (Student Natürlich Join Prüfung)
  Where (Semester=7) **AND** (Wohnort=Stuttgart)

- AS: Benennung von Ergebnis-Spalten

- Beispiel 4.3:

Bennet die Spalt Name zu Vorname um.

```
SELECT Name,
        ´Vorname: ` AS TEXT
FROM Student;
```

- IN: Test auf Mengenzugehörigkeit

Ai **IN** (a1, aj, ak) explizite Mengendefinition
Ai **IN** (**SELECT** . . .) implizite Mengendefinition—Geschachtelte

- Aggregatfunktionen

Um Benuter die SQL leicht verwenden zu lassen, SQL liefert viele Aggregatfunktionen. Durch solche Funktionen kann man seine Arbeit

- Beispiel 4.4:
- Man möchte die Durchnote von Student bekommen, die in Stuttgart wohnen.

```
SELECT AVG(Note) AS Durchschnittsnote
FROM   Student
WHERE  Wohnort=Stuttgart;
```

- Es gibt noch viele andere Aggregatfunktionen z.B SUM, MAX, MIN usw. dabei rechnet SUM die Gesamtwert von Zielspalt, sucht MAX die maximal Wert des Zielspaltes, MIN ist dagegen.

- Anfragen mit Mengenoperationen

Die Ergebniss von SQL Anfragen sind Mengen der Zeilen oder Elements. Deshalb kann man bei mehreren Ergebnissen von SQL Anfragen Mengenoperationen einführen.

- Beispiel 4.5:

Man wollte einen Ergebniss mit Wohnort in Stuttgart und(oder, differenz) die 6 Semester aussuchen.

```
SELECT  *
FROM    Student
WHERE  Wohnort=‚Stuttgart‘
INTERSECT(UNION,EXCEPT)
SELECT  *
FROM    Student
```

WHERE Semester=‚6‘

4.2 Datenmanipulation: Einfügen, Löschen und Ändern von Tupeln

- Datenmanipulation hat drei Operationen, man fügt ein, löscht oder ändert ein paar Zeilen von einer Tabelle.

4.2.1 Einfügen von Tupeln

- Hier table ist die Zieltabelle, dahinten sind die Spaltenlisten von der Zieltabelle. Mit VALUES kann man gewünschte Werte einfügen.

- **INSERT INTO** table [(column-commalist)]

  { **VALUES** row-constr.-commalist | table-exp | **DEFAULT VALUES** }

- Beispiel 4.6:

  Bei den Beispiel fügt man eine Tupel mit Wert(MatrNr=1006, Name=´Xi`, Semester=7, PLZ=70569, Wohnwort="Stuttgart") in die Tabelle Student.

  INSERT INTO Student (MatrNr, NAME, Semester, PLZ, Wohnort) **VALUES** (1006, „xi“,7, 70569, "Stuttgart");

4.2.2 Löschen VON TUPELN

- Löschen die Zeilen durch Beschränkung von WHERE.

- DELETE FROM table [WHERE cond-exp]

- Beispiel 4.7:

  Löschen Die Zeilen, die von Tabelle Student und Matrnr 1001 ist.

  DELETE FROM Student **WHERE** MatrNr = 1001;

4.2.3 UPDATE VON TUPELN

Akutualisieren den Inhalt mit der List von Tabelle durch Beschränkung von WHERE Satz

UPDATE table

SET update-assignment-commalist
[WHERE cond-exp**]**

– Beispiel 4.8:

Man wollte hier die Spalten PLZ alle Zeilen, die Wert von Wohnort Stuttgart sind, mit 70569 sein.

UPDATE Student
SET PLZ = 70569
WHERE Wohnort = "Stuttgart"

4.3 Datendefinition

4.3.1 Erstellen einer Schema

CREATE SCHEMA[schema] [**AUTHORIZATION** user]
[**DEFAULT CHARACTER SET** char-set]
[schema-element-list]

Man definiert hier eine Schema mit Name Student,und gebe Tu Zertifikate.

– Beispiel 4.9:

CREATE SCHEMA Student **AUTHORIZATION** TU

4.3.2 Erzeugen von Relationen

Erzeugen eine Relation hinter der Relationsname sind die Liste von gewünscht Attributen. Man kann folgende die Beschränkungen für Spalten geben.

CREATE TABLE base-table (base-table-element-commalist)
base-table-element ::= column-def | base-table-constraint-def

– Beispiel 4.9:

Hier erzeugt man eine Tabelle Teammate, mit Spalten PerNr, Höhe usw, hinter der Spatlten sind die Typen von Spalten mit Voraussetzungen.

CREATE TABLE Teammate
(PerNr Int NOT NULL,
Höhe Int NOT NULL**,**

Gewicht Int NOR NULL,
NAME CHAR (30) NOT NULL,
Wohnort CHAR (30) NOT NULL,
Semester INT NOT NULL**,**
PLZ INT NOT NULL
PRIMARY KEY (PerNr))

4.3.3 Löschen von Objekt

Nach DROP steht die Name von Tabelle ,View, Domain oder Schema, wobei Man eine von Restrict und Cascade wählen muss. Mit Restrict bedeutet, wenn es schon einen Datenbank von der Schema gibt, dann darf man Aktivität nicht verwenden Mit Cascade bedeutet, löscht man nicht nur die Schema sondern auch seine Datenbankobjekten.

DROP

DROP { **TABLE** base-table | **VIEW** view | **DOMAIN** domain |
SCHEMA schema } { **RESTRICT** | **CASCADE** }

Man möchte Tabelle Pers löschen mit Restrict.

– Beispiel 4.10:

DROP TABLE Pers **RESTRICT**

4.3.4 Änderung des Objekts

Hier ist Schlüsselwort add, für neuen Spalt oder neue Integritätsbedingung, Schlüsselwort Alter ändert die Name und Type von Spalten. Drop löscht die.

ALTER TABLE base-table
{**ADD** [**COLUMN**] column-def
| **ALTER** [**COLUMN**] column {**SET** default-def | **DROP DEFAULT**}
| **DROP** [**COLUMN**] column {**RESTRICT** | **CASCADE**}
| **ADD** base-table-constraint-def
| **DROP CONSTRAINT** constraint {**RESTRICT** | **CASCADE**}}

– Beispiele 4.11:

Man will den Spalt Alter mit Type INT in Tabelle Teammate einfügen, und den Spalt Wohnort löschen mit Beschränkung Restrict.

ALTER TABLE Teammate **ADD Alter INT**
ALTER TABLE Teammate **DROP COLUMN** Wohnort **RESTRICT**

***Restrict bedeutet,darf diese Attribut nicht von andere Table über Fremdschlüsser referenziert werden. Ansonsten wird die Operation zurückgewiesen. Cascade bedeutet, man kann den Spalten ganz löschen, auch wenn er in anderen Tabellen bleibt.**

5 XPath

- XPath ist Verkürzung von XML Path Language, es wird benutzt, um Teile von XML Dokument zu selektieren. Bei XPath gibt es auch Funktionen, die Char-, Int-, und Bulldaten behandeln.
- XPath benutzt eine kleine string-basierte Syntax. Die Syntax gehört nicht zum XML-Standard.
- XPath kann mit XQuery kombiniert werden.
- Dieses Kapital basiert hauptsächlich auf [2][3]

5.1 Datenmodell

XPath basiert auf einen Baumodell von XML Daten. Es gibt folgenden 7 Knoten Typen

- Wurzel Knoten
- Element Knoten
- Attribute Knoten
- „Namespace" Knoten
- „Processing Instruciton "Knoten
- Kommentar Knoten
- Text Knoten

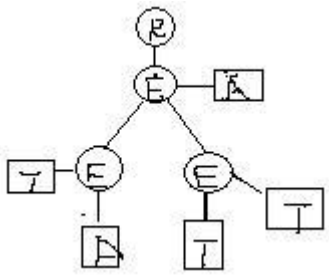

Abb5.1:Baummodell von XML-Daten vgl. Vorlessung AIM

5.2 XPath-Funktionen

4 Untermengen von Funktionen:

Node Set Functions; String Functions; Boolean Functions; Number Functions.
Kontext

Kontext ist besondere außer XPath-Ausdruck. Die besteht aus: Ein Knoten, Ein paar positive Integer(context position, context size), Eine Menge von Variablen, Ein Funktionsbibliothek, Eine Menge von Deklaration der Namespace. Manche Expressions ändern die Knoten von Kontext. Die Position und Size von Kontext kann nur von Prädikaten geändert werden

Location Path

relative location Path Jede Schritt von Path ist durch „/“ separiert von links nach rechts relative mit dem Knote

relative locationpath

ist immer relative mit dem Root vom Dokument. Location Schritten:

1 eine Axis; 2 a Knoten Test; 3 null oder mehrere Prädikate

Axes

- Eine Axis ist eine Menge von Knoten,die relative mit dem Zielknoten sind. X::Y
- bedeutet, dass man aus X Axis Y wählt.

- Beispiel 5.1:

```
<Teammate>
    <peron Höhe=„180“>
        <name>
            Beckenbauer
        </name>
        <job>Fussballer</job>
        <job>Manager</job>
    </person>
      .....
```

</Teammate> (XML-Dokument)

/child::Teammate
/child::person[child::name=„Beckenbauer“]
/attribute::Höhe (XPath-Ausdrücken)

- Das bedeutet, von obene XML-Dokument sucht man Höhe durch Weg /Teammate/Person[Name=„Beckenbauer“]/Attribute/Höhe

6 XQuery

- XQuery ist eine standarde Zielanfragen-Sprachen für XML-Daten.
- XQuery ist sehr flexibel, mit flexibele XML-Dokument zusammenzuarbeiten.
- Dieses Kapital basiert hauptsächlich auf [2][3]

6.1 Die standarde Ausdruck-Flowr

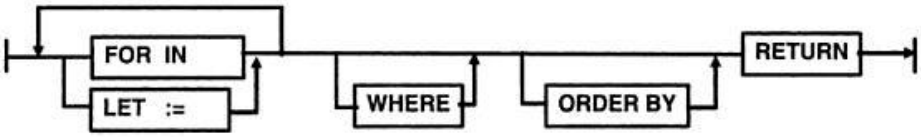

Abb 6.1: Syntax von FLWOR-Ausdrücken (DM, DBs und DBMS: Gottfried Vossen)

- Verglichen mit SQL:

For⇔SQL from
Where⇔SQL where
order by⇔SQL order by
Return⇔SQL select

- FOR, LET entwickeln eine List der Tuples, die mit Varablen zusammenbinden.

WHERE gibt eine Vorausetzung ab, durch die können Zieltuples ausgewählt aus.
ORDER BY hat gleiche Funktion wie im SQL
RETURN clause ist implementiert für übrigen Tuples, entwicklt eine Output-List.
LET und WHERE sind nicht notwendig.

- Beispiel 6.1:

  suche alle Teammate, die Höhe mehr als 180 haben. Und versammelt alle Ergebniss in eine List, die Name heisst.

  FOR $x **IN** /Teammate/Höhe
  LET $pn:= $x/@Name
  WHERE $x/Höhe > 180
  RETURN <Name> {$pn} </Name>

- Beispiel 6.2
  Joins in Xquery

  FOR $t **IN** /Person/teammate,
  $c **IN** /Person/coach,
  WHERE $a/Staat = $d/Staat
  RETURN <National>{ $c $a }</National>

- Beispiel 6.3:
  Kartesiches Produkt:

  FOR $t **IN** doc("tt.xml")/person/teammate,
  $c **IN** doc("tt.xml")/person/coach
  RETURN <national>{$c}{$a}</national>

6.2 Datenmanipulation

- Durch Datenmanipulation kann man Inhalt der XML-Dokumenten löschen, ändern und neue Inhalt einzufügen.

6.2.1 Einfügen von Knoten

- Syntax: **do insert**<source-expression> ([**as(first| last)] into| after| before)** <Ziel-Ausdruck>.

- Beispiel 6.4:

  Fügen ein Jahr Element nach Geburtstag von teammate

  do insert <year>1949</year> after
  fn:doc("tt.xml")/person/teammate/Geburtstag

6.2.2 Löschen null oder mehr Knoten

- Syntax: **do delete**<target-expression>

– Beispiel 6.5:

Löschen den Wohnort von teammate

do delete fn:doc("tt.xml")/person/teammate/wohnort

6.2.3 Ersetzen von Knoten/Werten

– Syntax: **do replace[value of]** <target-expression> **with**<new-expression>

– Beispiel 6.6:

Die Name von Teammate ist von Name der Coach ersetzt.

do replace fn:doc("tt.xml")/person/teammate/name
with fn:doc("tt.xml")/person/coach/name

Bei Ersetzen von Wert braucht man nur die „value of" vor fn:doc einfügen.

6.3 Erstellen einer geänderten Kopie

– Syntax: transform

copy<var> :=<expr> {,<var> :=<expr>}*
modify<updating-expression>
return<return-expression>

– Beispiel 6.7:

FOR $b **IN** fn:doc("tt.xml")/person/teammate[contains(name, "Müller")]
RETURN
transform
copy $xb:= $b
modify do delete $xb/Höhe
return $xb

7 Zusammenfassung

Ich habe in der Ausarbeitung über die relationalen Modell, die Relationale Operation, die SQL-Sprache von Relationdatenbank beschrieben, und über die XPath und XQuery grob erklärt. Es handelt sich in SIMPLs viele über Daten-Austauschen mit heterogene Datenquelle, müssen wir dafür eine erweiteres generisches Ramenwerk erstellen. Deshalb erfassen wir oben gennanten Inhalt, sind notwendig.

Literature

[1] Kemper, A.; Eickler, A.: Datenbanksysteme - Eine Einführung, Oldenbourg, 6. Auflage, 2006

[2] Lehner, W.; Schöning, H.: XQuery: Grundlagen und fortgeschrittene Methoden, dpunkt, 2004

[3] W3C: XQuery – Technical Report, http://www.w3.org/TR/xquery/