



SQL/XML

Aktueller Stand der Standardisierung

Rüdiger Eisele
IBCPartner GmbH
Stuntzstraße 65
81677 München

Tel: 089/92401181 Fax: 089/92401182
Internet: EISELE@SOFTEISCONS.DE
EISELE@IBCPARTNER.DE

Vorbemerkung:



SQL/XML: Stand der Entwicklung

- Der Vortrag beschäftigt nur mit ISO:9075-200x-14 SQL/XML.
- Nicht mit den bereits am Markt vorhandenen Produkten.
- Guter Artikel zu diesem Thema:
“XML in relationalen Datenbanken”

Jost Enderle Universität Ulm

Informatik Spektrum Band 24 Heft 6 Dezember 2001

■ Themen

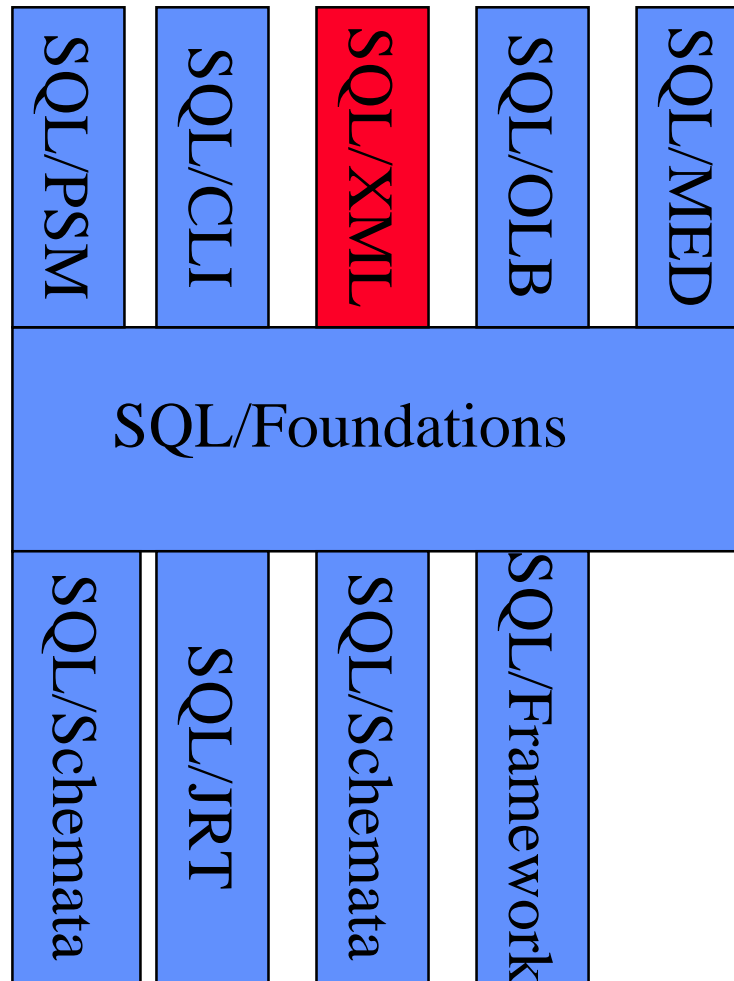
- Status von SQL-*
- Wie kommt es zu SQL und XML
- Stand der Standardisierung von SQL/XML

SQL/XML: Stand der Entwicklung

- “SQL99“
 - » Foundations
 - » Einige Packages
 - SQL/PSM
 - SQL/CLI
 - SQL/MED
- SQL2002
 - » Im Laufe des Jahres 2002

Aktueller Status von SQL

SQL/XML: Stand der Entwicklung



– SQL ist

- » der internationale Standard in der relationalen Datenbankwelt
- » eine Datenbanksprache
- » Herstellerunabhängig
- » gibt es seit 1987
- » wurde und wird von ISO/SC32WG3 entwickelt
- » wurde auch beeinflusst von der SQL-ACCESS-Group
- » wurde bis einschl. SQL92 von NIST zertifiziert

- XML ist
 - » eine Präsentationssprache
 - » eine Sprache zur Speicherung von Daten
 - » ist eine ‘Weiterentwicklung‘ von SGML und HTML
 - » soll die Präsentationssprache im Web werden.
 - » herstellerunabhängig
 - » ist noch kein ISO-Standard
 - » wird von von W3C entwickelt
 - » keine Zertifizierung

■ Technischer Aspekt

– SQL

- » ist ausgereift
- » Der Standard ist wesentlich weiter als die Implementierungen
- » viele Produkte sind am Markt vorhanden
- » sehr viele Applikationen
- » ‘nur‘ eine Datenbanksprache und damit nur Datenverwaltung, keine Datenpräsentation

■ Technischer Aspekt

– XML

- » ist neu
- » wenig Implementierungen
- » wenige Applikationen
- » keine Business kritischen Applikationen
- » hauptsächlich Präsentation von Daten

■ Marketing Aspekt

- SQL ist alt
- SQL sieht man nicht
- XML ist das WEB
- XML ist neu (hype)

SQL/XML: Stand der Entwicklung

- Es gibt W3C mit all seinen Initiativen
 - XQuery, XPath, XMLSchema, XForms,.....

Konkurrenzveranstaltung zu SQL, die aber noch nicht soweit ist wie SQL.

z.B. ist man bei XQuery gerade dabei die Sprachelemente zu definieren. Eine Arbeit, die bei SQL schon Mitte der 1980er Jahre getan wurde, wobei das Ergebnis das gleiche ist.

Es gibt Datentypen, Operatoren auf den Datentypen, etc. (Déjà vû)

■ W3C XML Query WG - Goals

“The goal of the XML Query WG is to produce a data model for XML documents, a set of query operators on that data model, and a query language based on these query operators“

Auszug aus einem Vortrag dem

“Querying XML Documents“

vom Cotton/Robie,

gehalten auf der Unicode Conference im Januar 2002

SQL/XML: Stand der Entwicklung

- Es gab zwei Ansätze SQL und XML zu verheiraten, beide kamen von Herstellern von Datenbankmanagementsystemen.
- Ansatz 1.
 - SQL wird von XML langfristig abgelöst
- Ansatz 2.
 - SQL bietet eine Schnittstelle zu XML an.

■ Ansatz 1:

– Behauptungen:

- » SQL Datenmodell und XML-Datenmodell sind nahezu identisch.
- » Nur durch eine geringfügige Änderung in SQL kann man die beiden Datenmodelle identisch machen.

– Annahme:

- » XML wird SQL ersetzen !

■ Ansatz 2:

– Voraussetzungen:

- » SQL bleibt was es ist, eine Datenbanksprache.
- » SQL wird weiterhin seine Pflicht erfüllen.
- » Die Mehrheit der Daten sind und werden weiterhin in einer SQL-Datenbank gespeichert werden.
- » XML wird die Schnittstelle zwischen dem Benutzer und SQL herstellen.
- » XML wird es ermöglichen, Daten zwischen den verschiedenen Datenhaltungssystemen auszutauschen.
(EXPORT/IMPORT)

■ Zusammenfassung der beiden Ansätze

- Ansatz 1 will eine neue Sprache.
 - » Dieser Ansatz geht ein bißchen in die Richtung, die auch W3C verfolgt.
- Ansatz 2 will eine Erweiterung von SQL, eine Schnittstelle zu SQL.
 - » Dieser Ansatz verfolgt weiter die von ISO eingeschlagene Richtung der Packages.

■ Fehler im Ansatz 1

- Die beiden Datenmodelle sind nicht gleich.
 - » XML liegt ein hierarchisches Datenmodell zu Grunde
 - » SQL dagegen besitzt ein relationales Datenmodell.

SQL/XML: Stand der Entwicklung

- In einem relationalen Modell kann man 1:1, 1:m und m:n Beziehungen herstellen.
- In einem hierarchischen Modell kann man keine m:n Beziehungen herstellen.

■ Änderung in SQL

TABLE EXPRESSION

<table expression> ::=

~~<from clause>~~ | **<for clause>**

[<where clause>]

[<group by clause>]

[<having clause>]

<for clause> ::=

FOR <iterator list>

Diese stellt einen eklatanten Bruch in SQL dar !

SQL/XML: Stand der Entwicklung

<iterator list> ::= <iterator> ::=

<correlation name> IN <table or query name>

| <correlation name> IN <derived table>

| <correlation name> IN <lateral derived table>

| <correlation name> IN <collection derived table>

| <correlation name> IN <only spec>

| ~~<left paren> <joined table> <right paren>~~



■ Was hat ISO gemacht ?

- Hat sich mit den Vorschlägen intensiv beschäftigt und hat beschlossen ein Package für XML zu standardisieren.
- SQL/XML ISO:9075-14
- Status im Februar 2002:
 - » ISO FCD-Ballot
 - Ballotperiode dauert 4 Monate
 - d.h. SQL/XML wird voraussichtlich in 2003 ein internationaler Standard sein.

■ Was beinhaltet der Standard ?

- Präsentation von XML in SQL und umgekehrt
- Abbilden von XML Schemata in SQL Schemata
- Repräsentation von SQL Schemata in XML
- Repräsentation von SQL Aktionen in XML
- SQL-freundliche Syntax zur Abfrage XML-Dokumenten

■ Augenblicklicher Inhalt von SQL/XML

– Mapping:

- » SQL Zeichensatz in XML Zeichensatz
- » SQL <identifiers> in XML Names
- » SQL Datentypen in XML Schema Daten Typen
- » SQL Datenwerte in XML Datenwerte
- » SQL Tabelle in XML Dokument und XML Schema Dokument
- » SQL Schema in XML Dokument und XML Schema Dokument
- » SQL Katalog in XML Dokument und XML Schema Dokument

■ Augenblicklicher Inhalt von SQL/XML.....

- Mapping:
 - » XML Unicode in SQL Zeichensatz
 - » XML Namen in SQL <identifiers>
- Eigener Datentyp XML

SQL/XML: Stand der Entwicklung

Der SQL/XML referiert verschiedene Namespaces, die entweder von W3C oder vom SQL-Standard definiert wurden; die Namespaces werden über sog. Namespacevariablen angesprochen.

Namespacevariable

Namespace URI

xsd:

<http://www.w3.org/2001/XMLSchema>

xsi:

<http://www.w3.org/2001/XMLSchema-instance>

sqlxml:

<http://www.iso-standards.org/mra/9075/2001/12/sqlxml>

■ Mapping SQL nach XML

- SQL Character Set nach XML Character Set
- SQL <identifiers>s nach XML Names
- SQL datatypes nach XML Schema data types
 - » vordefinierte und benutzerdefinierte
- SQL data values nach XML data values
- SQL table nach einem XML Document und einem XML Schema Document
- SQL schema nach einem XML Schema document
- SQL catalog nach einem XML Schema document

Mapping SQL Character Set nach XML Character Set

- Für jeden SQL Character Set eines SQL-Environments muß eine Abbildung (Mapping) definiert sein, die den SQL Character Set in Unicode Zeichenketten überführt.

■ Mapping SQL <identifiers>s nach XML Names

- Regular Identifier stellen kein Problem dar
 - » employee ==> EMPLOYEE
- Delimited Identifier machen die Sache etwas komplizierter.
 - » “Employee“ ==> Employee
 - » “Name&/Employee“ ==> ??

■ Mapping SQL <identifiers>s nach XML Names

- Lösung: Umsetzung der nicht darstellbaren Zeichen im Unicode unter Benutzung sog. “escape“ Notation
 - » / ==> `_x002F_` (Unicode Wert)
 - » “Name&/Employee“ ==> `Name_x0020__x002F_Employee`
- Das Ganze ist reversibel, d.h. die Unicodewerte können in gültige SQL - Zeichen übersetzt werden.

Mapping SQL datatypes nach XML Schema data types

- Jeder vordefinierte SQL Typ wird in einen XML Schema built-in Typ umgesetzt, der dem SQL Typ am nächsten kommt.
- Mit Hilfe der *XML facets* werden die Restriktionen der SQL Typen realisiert.
 - Restriktionen: Precision, scale, Character Varying
- Es gibt in XML keine Unterscheidung zwischen CHARACTERVARYING und CHARACTER LARGE OBJECT; diese Unterschiede werden in XML durch *Annotations* realisiert. Die Annotations werden im Standard definiert.

■ Mapping SQL Datatypes auf XML Schema Type Character String Types

SQL: CHAR(10) CHARACTER SET LATIN1
COLLATION DEUTSCH

```
<xsd:simpleType>  
  <xsd:restriction base="xsd:string">  
    <xsd:length value="10"/>  
    <xsd:annotation>  
      <sqlxml:sqltype name="CHAR" length="10"  
        charcaterSetName="LATIN1"  
        collation="DEUTSCH" />  
    </xsd:annotation>  
  </xsd:restriction>  
</xsd:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

Binary String Types

SQL: BLOB(1000)

```
<xsd:simpleType>
```

```
<xsd:restriction base="sqlxml:binaryhex">
```

```
<xsd:maxLength value="2000"/>
```

```
<xsd:annotation>
```

```
<sqlxml:sqltype name="BLOB"  
                maxLength="1000"/>
```

```
</xsd:annotation>
```

```
</xsd:restriction>
```

```
</xsd>:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

NUMERIC und DECIMAL

SQL: DECIMAL(8,2) wobei die Implementierung 9 benutzt

```
<xsd:simpleType>  
  <xsd:restriction base="xsd:decimal">  
    <xsd:precision value="9"/>  
    <xsd:scale value="2"/>  
    <xsd:annotation>  
      <sqlxml:sqltype name="DECIMAL"  
        userprecision="8" scale="2" />  
    </xsd:annotation>  
  </xsd:restriction>  
</xsd:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

INTEGER und SMALLINT

SQL: SMALLINT

```
<xsd:simpleType>
```

```
  <xsd:restriction base="xsd:integer">
```

```
    <xsd:maxInclusive value="32767"/>
```

```
    <xsd:minInclusive value="-32768"/>
```

```
  <xsd:annotation>
```

```
    <sqlxml:sqltype name="SMALLINT">
```

```
  </xsd:annotation>
```

```
</xsd:restriction>
```

```
</xsd:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

FLOAT(p), REAL, DOUBLE PRECISION

SQL: REAL

```
<xsd:simpleType>
```

```
<xsd:restriction base="xsd:float">
```

```
<xsd:annotation>
```

```
<sqlxml:sqltype name="REAL"
```

```
precision="24"
```

```
minExponent="-149"
```

```
maxExponent="104"/>
```

```
</xsd:annotation>
```

```
</xsd:restriction>
```

```
</xsd:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

BOOLEAN

SQL: BOOLEAN

```
<xsd:simpleType>
```

```
  <xsd:restriction base="xsd:boolean"/>
```

```
  <xsd:annotation>
```

```
    <sqlxml:sqltype name="BOOLEAN"/>
```

```
  </xsd:annotation>
```

```
</xsd:restriction>
```

```
</xsd>:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

DATE

SQL: BOOLEAN

```
<xsd:simpleType>
```

```
<xsd:restriction base="xsd:date"/>
```

```
<xsd:pattern value =
```

```
"\p{Nd}{4}-\{Nd\}{2}-\p{Nd}{2}">
```

```
<xsd:annotation>
```

```
<sqlxml:sqltype name="DATE"/>
```

```
</xsd:annotation>
```

```
</xsd:restriction>
```

```
</xsd>:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

TIME und TIME WITH TIME ZONE

SQL: TIME(2)

```
<xsd:simpleType>
```

```
<xsd:restriction base="xsd:time">
```

```
<xsd:pattern value =
```

```
"\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}.\p{Nd}{2}">
```

```
<xsd:annotation>
```

```
<sqlxml:sqltype name="TIME"
```

```
scale="2"/>
```

```
</xsd:annotation>
```

```
</xsd:restriction>
```

```
</xsd:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

INTRVAL YEAR(4) TO MONTH

```
<xsd:simpleType>  
  <xsd:restriction base="xsd:timeDuration">  
    <xsd:pattern value =  
      "-?p{Nd}{1,4}Y\p{Nd}{2}M"/>  
  <xsd:annotation>  
    <sqlxml:sqltype name="INTERVAL YEAR TO MONTH"  
      leadingPrecision="4"/>  
  </xsd:annotation>  
</xsd:restriction>  
</xsd:simpleType>
```

■ Mapping SQL Datatypes auf XML Schema Type

- Benutzerdefinierte Datentypen
- CREATE DOMAIN Dumb.Deer.Dumdum
AS VARCHAR(1) DEFAULT ''
CONSTRAINT Dumb.Deer.DumdumConstraint
INITIALLY IMMEDIATE NOT DEFERABLE
CHECK (CHAR_LENGTH(VALUE) < 1)

SQL/XML: *Stand der Entwicklung*

Wird abgebildet in die XML type Deklaration

```
<xsd:simpleType name="Domain.DUMB.DEE.DUMDUM">
  <xsd:annotation>
    <xsd:appinfo>
      <sqlxml:sqltype kind='DOMAIN'
        catalogName='DUMB'
        schemaName='DEE'
        typeName='DUMDUM'
        mappedType='VARCHAR_1'>
      </sqlxml:sqltype>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:restriction base='VARCHAR_1'/>
</xsd:simpleType>
```

SQL/XML: *Stand der Entwicklung*

ROW (Name VARCHAR(50), Birthdate DATE)

```
<xsd:complexType name='ROW.idi'>
  <xsd:annotation>
    <xsd:appinfo>
      <sqlxml:sqltype kind='ROW'>
        <sqlxml:field name='NAME' mappedType='VARCHAR_50'/>
        <sqlxml:field name='BIRTHDATE' mappedType='DATE'/>
      </sqlxml:sqltype>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name='NAME' nillable='true'
      type='VARCHAR_50'/>
    <xsd:element name='BIRTHDATE' nillable='true'
      type='DATE'/>
  </xsd:sequence>
</xsd:complexType>
```

SQL/XML: Stand der Entwicklung

```
CREATE TYPE Most.Unlikely.Possibility
AS NUMERIC(1)
<xsd:simpleType name='UDT.MOST.UNLIKELY.POSSIBILITY'>
  <xsd:annotation>
    <xsd:appinfo>
      <sqlxml:sqltype kind='DISTINCT'
        catalogName='MOST'
        schemaName='UNLIKELY'
        typeName='POSSIBILITY'
        mappedType='NUMERIC_1_0'
        final='true'/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:restriction base='NUMERIC_1_0'/>
</simpleType>
```

SQL/XML: Stand der Entwicklung

VARCHAR(20) ARRAY[10]

```
<xsd:complexType name='ARRAY_10.VARCHAR_20'>
  <xsd:annotation>
    <xsd:appinfo>
      <sqlxml:sqltype kind='ARRAY' maxElements='10'
        mappedElementType='VARCHAR_20'/>
    </xsd:appinfo>
  </xsd:annotation>
```

SQL/XML: Stand der Entwicklung

```
<xsd:sequence>
  <xsd:element name='element' minOccurs='0'
                maxOccurs='10' nillable='true'
                type='VARCHAR_20'/>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
```

SQL/XML: *Stand der Entwicklung*

VARCHAR(20) MULTISSET

```
<xsd:complexType name='MULTISSET.VARCHAR_20'>
  <xsd:annotation>
    <xsd:appinfo>
      <sqlxml:sqltype kind='MULTISSET'
        mappedElementType='VARCHAR_20'/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name='element' minOccurs='0'
      maxOccurs='unbounded' nillable='true'
      type='VARCHAR_20'/>
  </xsd:sequence>
</xsd:complexType>
```

SQL/XML: Stand der Entwicklung

- Mapping SQL data values nach XML data values
 - Die Umsetzung der Werte wird bestimmt durch die Typdefinition zu der der Wert gehört.

■ Der Datentyp XML

- Noch sehr rudimentär
- Warum führt man einen Datentyp XML ein ?
 - » Es soll auf XML innerhalb von SQL zugegriffen werden können.
 - » Es sollen XML Dokumente erzeugt werden können, einzelnen Elemente soll extrahiert werden können.
 - » XML Elemente sollen dauerhaft in relationalen Tabellen gespeichert werden können.
 - » SQL soll dahingehend erweitert werden, daß man XML-Elemente innerhalb des RDBMS erzeugen kann.

SQL/XML: Stand der Entwicklung

```
CREATE TABLE job_angebote (  
    name VARCHAR(128),  
    resume XML  
);
```

So könnte eine Tabellendefinition mit einem Feld vom Typ XML aussehen.
Es fehlen im Standard noch die Operationen zu dem Typ XML.

SQL-Tabelle: Mitarbeiter

lfdnr	V_name	N_name	Einstelldatum
1001	Herbert	Meyer	10-Dez-2000
1029	Karin	Schmitz	01-Jan-2002

SQL/XML: *Stand der Entwicklung*

XMLElement: erzeugt ein XML Element

```
SELECT m.lfdnr,  
       XMLELEMENT (NAME "Emp", m.v_name || ' ' || m.n_name ),  
       XMLELEMENT (NAME "Einstelldatum", m.einstelldatum)  
AS Mitarbeiterdaten  
FROM MITARBEITER m  
WHERE ..... ;
```

lfdnr	Mitarbeiterdaten
1001	<emp> Herbert Meyer </emp> <Einstelldatum>10-Dez-2000 </Einstelldatum>
1029	<emp> Karin Schmitz</emp> <Einstelldatum>01-Jan-2002</Einstelldatum>

SQL/XML: Stand der Entwicklung

SELECT e.id,

```
    XMLELEMENT ( NAME "Emp",  
    XMLELEMENT ( NAME "name",e.fname || ' ' || e.lname),  
    XMLELEMENT ( NAME "hiredate", e.hire )  
  ) AS "result"  
FROM employees e  
   WHERE ... ;
```

SQL/XML: *Stand der Entwicklung*

ID	result
1001	<pre><Emp> <name>John Smith</name> <hiredate>2000-05-24</hiredate> </Emp></pre>
1206	<pre><Emp> <name>Mary Martin</name> <hiredate>1996-02-01</hiredate> </Emp></pre>

SQL/XML: Stand der Entwicklung

XMLForest: erzeugt ein XML Forestelement

```
SELECT m.lfdnr,  
       XMLELEMENT (NAME "Emp",  
                  XMLATTRIBUTES ( m.v_name || ' ' || m.n_name as "Name"),  
                  XMLFOREST ( m.einstelldatum, m.abtl as "Abteilung")  
       AS "Mitarbeiterdaten"  
FROM MITARBEITER m  
WHERE ..... ;
```

SQL/XML: *Stand der Entwicklung*

Lfdnr	Mitarbeiterdaten
1001	<pre><emp Name= Herbert Meyer> <Einstelldatum>10-Dez-2001</Einstelldatum> <Abteilung>Konstruktion</Abteilung> </emp></pre>
1029	<pre><emp Name=Karin Schmitz> <Einstelldatum>01-Jan-2002</Einstelldatum> <Abteilung>Vorstandsbüro</Abteilung> <emp></pre>

XMLGen

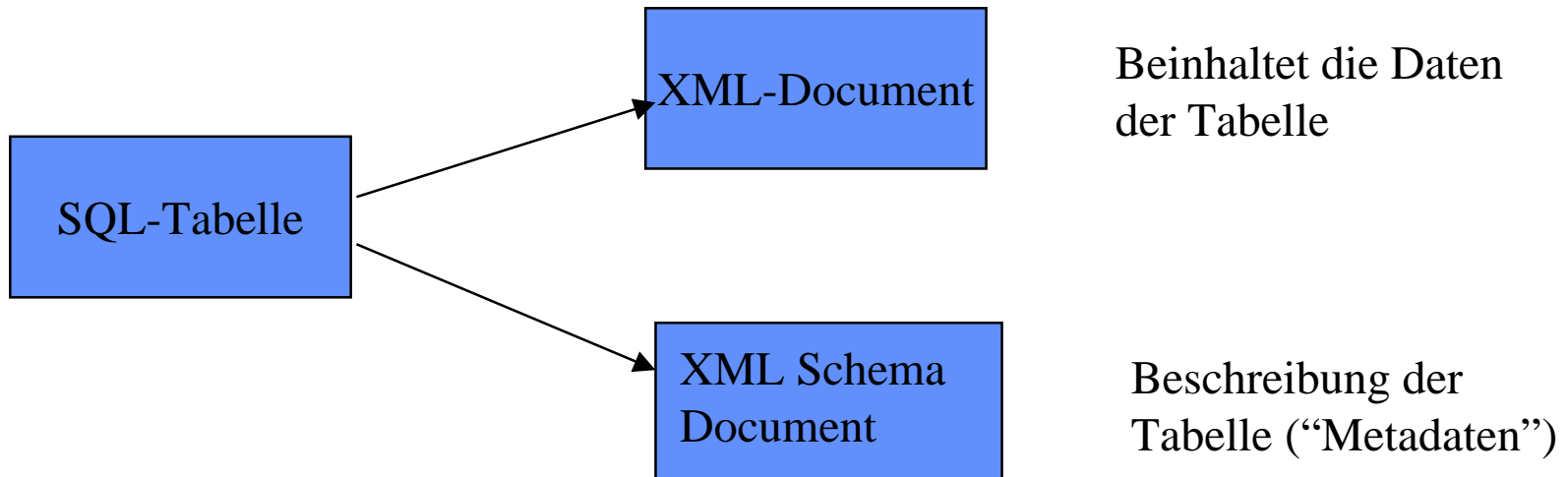
```
SELECT ma.ma_nr,  
       XMLGEN ( '<Mitarbeiter Name="{ $NAME }">  
                <Einstelldatum>{ $HIRE }</Einstelldatum>  
                <Abteilung>{ $DEPT }</Abteilung>  
                <Mitarbeiter>',  
               ma.v_name || ' ' || ma.n_name AS name,  
               ma.einstelldatum , ma.abteilung  
             ) AS "Ergebnis"  
FROM mitarbeiter ma  
WHERE ... ;
```

SQL/XML: *Stand der Entwicklung*

Ma_ nr	Ergebnis
1001	<pre><Mitarbeiter name="Gerhard Hinz"> <Einstelldatum>24-05-2000</Einstelldatum> <Abteilung>Buchhaltung</Abteilung> </Mitarbeiter></pre>
1206	<pre><Mitarbeiter name="Heinz Kunz"> <Einstelldatum>01-02-1999</Einstelldatum> <Abteilung>Versand</Abteilung> </Mitarbeiter></pre>

SQL/XML: *Stand der Entwicklung*

- SQL table wird in ein
 - XML Document und ein XML Schema Document übertragen.



SQL/XML: *Stand der Entwicklung*

<EMPLOYEE>

<row>

<EMPNO>000010</EMPNO>

<FIRSTNME>CHRISTINE</FIRSTNME>

<LASTNAME>HAAS</LASTNAME>

<BIRTHDATE>1933-08-24</BIRTHDATE>

<SALARY>52750.00</SALARY>

</row>

<row>

<EMPNO>000020</EMPNO>

<FIRSTNME>MICHAEL</FIRSTNME>

<LASTNAME>THOMPSON</LASTNAME>

<BIRTHDATE>1948-02-02</BIRTHDATE>

<SALARY>41250.00</SALARY>

</row>

</EMPLOYEE>

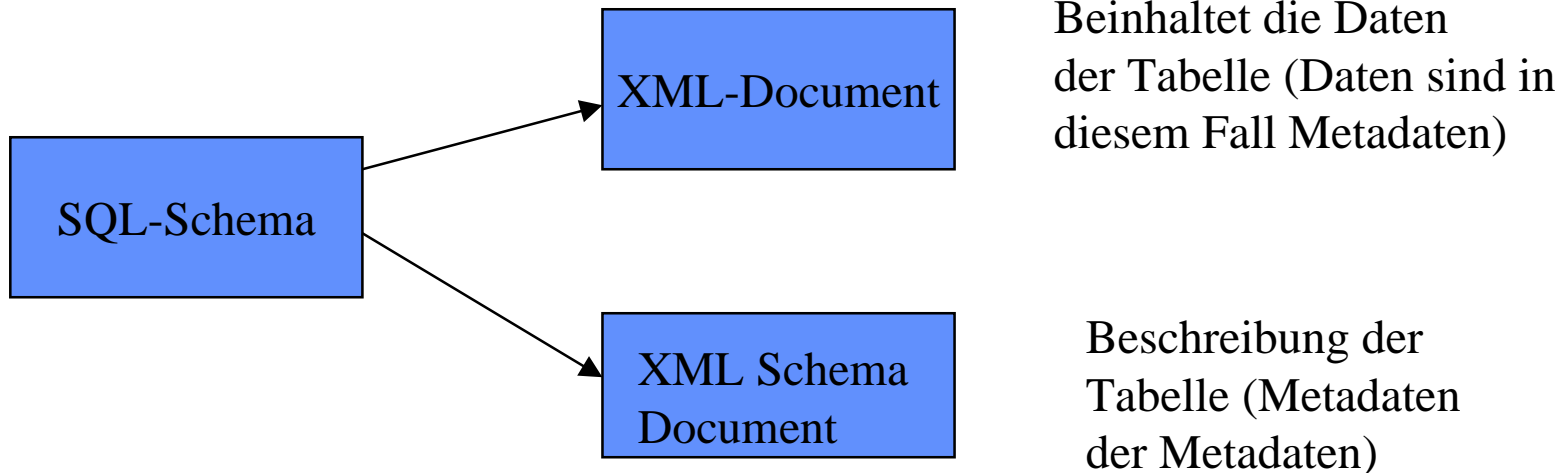
SQL/XML: *Stand der Entwicklung*

```
<xsd:schema>
  <xsd:element name="EMPLOYEE">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="row"
          minOccurs="0"
          maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="EMPNO">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:length value="6"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```
<xsd:element name="FIRSTNAME">  
    ...  
    </xsd:element>  
    .  
    </xsd:sequence>  
    </xsd:complexType>  
    </xsd:element>  
    </xsd:sequence>  
    </xsd:complexType>  
    </xsd:element>  
#</xsd:schema>
```

SQL/XML: *Stand der Entwicklung*

- SQL Schema wird in ein
 - XML Document und ein XML Schema Document übertragen.



■ Abbildung einer SQL Tabelle in XML Schema Typen

```
CREATE TABLE Employees (  
    EmpName    CHARACTER (20),  
    Salary     NUMERIC(8,2),  
    HireDate   DATE);
```

■ Abbildung einer SQL Tabelle in XML Schema Typen

```
<xsd:complexType name="EMPLOYEES">  
  <xsd:sequence>  
    <xsd:element name="EMPNAME">  
      <xsd:simpleType>  
        <xsd:restriction base="string">  
          <xsd:length value="20"/>  
        <xsd:restriction>  
      <xsd:simpleType>  
    <xsd:element>
```

■ Abbildung einer SQL Tabelle in XML Schema Typen

```
<xsd:element name = "SALARY">  
  <xsd:simpleType>  
    <xsd:restriction base = "decimal">  
      <xsd:precision value="8"/>  
      <xsd:scale value="2"/>  
    <xsd:restriction>  
      <xsd:simpleType>  
      <xsd:element>
```

■ Abbildung einer SQL Tabelle in XML Schema Typen

```
<xsd:element name = "HIREDATE">  
  <xsd:simpleType>  
    <xsd:restriction base = "date">  
      <xsd:pattern value = "\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}" />  
    <xsd:restriction>  
  <xsd:simpleType>  
  <xsd:element>  
  <xsd:sequence>  
<xsd:complexType>
```

■ Abbildung einer SQL Tabelle in XML Schema Typen

```
<xsd:complexType name="EMPLOYEES">  
  <xsd:sequence>  
    <xsd:element name="EMPNAME">  
      <xsd:simpleType>  
        <xsd:restriction base="string">  
          <xsd:length value="20"/>  
        <xsd:restriction>  
      <xsd:simpleType>  
    <xsd:element>
```

■ Mapping XML nach SQL

- UNICODE nach SQL character set
- XML Names nach SQL <identifier>s

■ Mapping UNICODE nach SQL character set

Abbildung von UNICODE nach Zeichenketten des jeweiligen SQL Character sets.

Genau der umgekehrte Weg wie bei SQL Character Set nach UNICODE.

Mapping XML Names nach SQL <identifizier>s

Algorithmus, der die das Mapping der SQL <identifizier>s nach XML Names, umgekehrt.

Scanning des XML-Names von links-nach-rechts nach den Escape sequenzen und umsetzen derselben in SQL_TEXT Zeichen.

SQL/XML: Stand der Entwicklung

SELECT fragen

FROM zuhörer

WHERE thema = 'SQL/XML' ;

FOR \$q IN

document ("decus2002") // zuhörer

WHERE \$q/thema = "SQL/XML"

RETURN \$q/fragen/text()