Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Chapter 9 – SQL/XML

Outline

Overview

I. Object-Relational Database Concepts

1. User-defined Data Types and Typed Tables
2. Object-relational Views and Collection Types
3. User-defined Routines and Object Behavior
4. Application Programs and Object-relational Capabilities

II. Online Analytic Processing

5. Data Analysis in SQL
6. Windowed Tables and Window Functions in SQL

III. XML

7. XML Data Modeling
8. Xquery
9. **SQL/XML**

IV. More Developments (if there is time left)

temporal data models, data streams, databases and uncertainty, …

SQL and XML?!

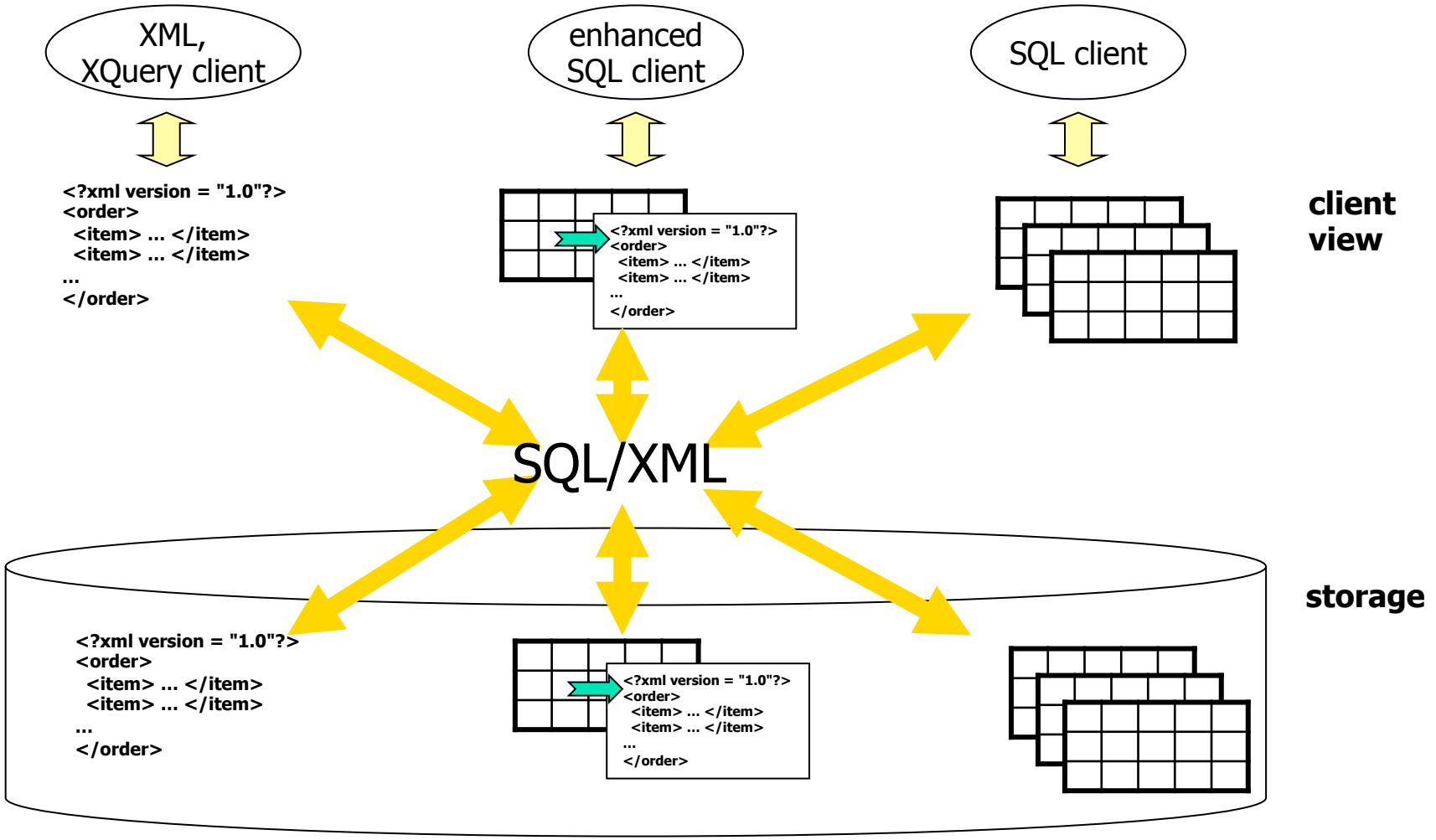
- Two major perspectives
  - Flexible exchange of relational data using XML
    - publish relational as XML
    - decompose or "shred" XML into relational
  - Reliable XML data management
    - manage, search, maintain, update, …
    - integrate with relational data
- Native-XML databases? No significant customer interest!
  - reluctance to introduce new DBMS environment
  - limited integration with relational DBMS products
  - lack of maturity (scalable, reliable, highly available, …)
  - skill revolution (not evolution) required

Remember OO-DBMS?

SQL and XML

- Use existing (object-)relational technology?
  - Large Objects: granularity understood by DBMS may be too coarse!
    - search/retrieval of subsets, update of documents
  - Decompose into tables: often complex, inefficient
    - mapping complexity, especially for highly "denormalized" documents
  - Useful, but not sufficient
    - should be standardized as part of SQL
    - but needs further enhancement to support "native" XML support in SQL
- Enable "hybrid" XML/relational data management
  - supports both relational and XML data
    - storage, access
    - query language
    - programming interfaces
  - ability to view/access relational as XML, and XML as relational
  - all major relational DBMS vendors are moving into this direction

SQL/XML Big Picture



XML Data Type

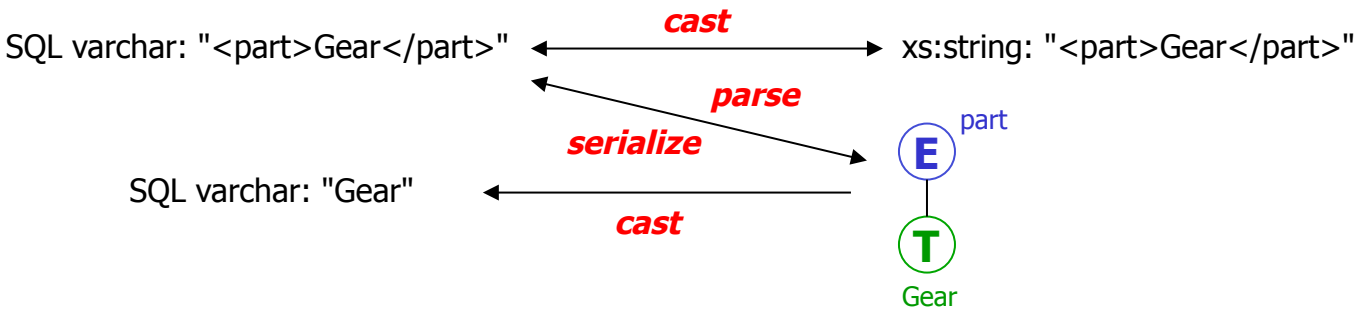
- New SQL type "XML"
  - value of type XML is an instance of the XQuery data model
    - not just a well-formed XML document
  - can have optimized internal representation (different from character string)
  - used for storing XML data "natively" in the database and capturing the data type of results and input values of SQL/XML functions that work with XML data
    - optional: schema validity

```
CREATE TABLE employees
  ( id          CHAR(6),
    lastname    VARCHAR (30),
    ...
    resume      XML
  )
```

ID	LASTNAME	...	RESUME
940401	Long	...	<?xml version="1.0"?> <resume xmlns="http://www.res.com/resume"> <name> … </name> <address> … </address> ... </resume>
862233	Nicks	...	null
766500	Banner	...	<resume ref="http://www.banner.com/resume.html"/>

Converting SQL to XML, XML to SQL

- XMLPARSE and XMLSERIALIZE convert to/from character strings and BLOBs
  - explicit invocation of XMLPARSE and XMLSERIALIZE functions
  - implicit conversion (during host language interaction)
- XMLCAST converts SQL values into an XML values (and vice versa)
  - Values of SQL predefined types are cast to XQuery atomic types using
    - The defined mapping of SQL types/values to XML Schema types/values
    - The semantics of XQuery's cast expression
  - XML values are converted to values of SQL predefined types using a combination of
    - The defined mapping of SQL types to XML Schema types and SQL's CAST specification
    - XQuery's fn:data() function and cast expression
- Note: XMLCAST to/from character strings is different from XMLSERIALIZE and XMLPARSE



SQL/XML "Constructor Functions"

- Functions/operators for generating XML constructs (elements, attributes, ...) within an SQL query
- Function syntax for generating XML nodes of various types
  - XMLELEMENT, XMLATTRIBUTE, XMLCOMMENT, XMLPI, XMLTEXT
  - XMLDOCUMENT wraps an XQuery document node around an XML value
- Producing sequences of values/nodes
  - XMLFOREST generates sequence of element nodes
  - XMLCONCAT concatenates XML values
- Concatenation over sets of tuples
  - XMLAGG aggregates XML across multiple tuples
- Naming of elements and attributes is either explicit or implicit (based on column names)

XMLELEMENT

- Produces an XML value that corresponds to an XML element, given:
  - An SQL identifier that acts as its **name**
  - An optional list of **namespace** declarations
  - An optional list of named expressions that provides names and values of its **attributes**, and
  - An optional list of expressions that provides its **content**
- Multiple options for NULL content (NULL, empty element, nil='true', ...)
- Support for nested elements (with mixed content) and subqueries

```
SELECT   e.id,
  XMLELEMENT (NAME "Emp",
    XMLELEMENT (NAME "name", e.lname ),
    XMLELEMENT (NAME "dependants",
              (SELECT COUNT (*)
               FROM dependants d
               WHERE d.parent = e.id))
               ) AS "result"
FROM employees e
WHERE ... ;
```

ID	result
1001	<Emp> <name>Smith</name> <dependants>3</dependants> </Emp>

XMLATTRIBUTES (within XMLELEMENT)

- Attribute specifications must appear directly after element name and optional namespace declaration.
- Each attribute can be named implicitly or explicitly.

```
SELECT   e.id,
         XMLELEMENT (NAME "Emp",
                     XMLATTRIBUTES (e.id, e.lname AS "name")
         ) AS "result"
FROM employees e
WHERE … ;
```

➔

ID	result
1001	<Emp ID="1001" name="Smith"/>
1006	<Emp ID="1206" name="Martin"/>

XMLAGG

- An aggregate function, similar to SUM, AVG, etc.
  - The argument for XMLAGG must be an expression of XML type.
  - For each row in a group G, the expression is evaluated and the resulting XML values are concatenated to produce a single XML value as the result for G.
  - An ORDER BY clause can be specified to order the results of the argument expression before concatenating.

```
SELECT   XMLELEMENT (
           NAME "Department",
           XMLATTRIBUTES ( e.dept AS "name" ),
           XMLAGG (XMLELEMENT
                     (NAME "emp", e.lname)
             ORDER BY e.lname)
           ) AS "dlist",
           COUNT(*) AS "dcount"
FROM employees e
GROUP BY dept ;
```

dlist	dcount
<Department name="Accounting"> <emp>Smith</emp> <emp>Yates</emp> </Department>	2
<Department name="Shipping"> <emp>Martin</emp> <emp>Oppenheimer</emp> </Department>	2

SQL/XML Constructor Function Usage

- Dynamically retrieve SQL data in XML format (optionally mixed with SQL)
- Use query results to update/insert into tables with XML columns
- Use standard SQL views to create logical tables with XML columns

```
CREATE VIEW XMLDept (DeptDoc XML) AS (
    SELECT   XMLELEMENT ( NAME "Department",
                          XMLATTRIBUTES ( e.dept AS "name" ),
                          XMLATTRIBUTES ( COUNT(*) AS "count",
                          XMLAGG (XMLELEMENT (NAME "emp",
                                              XMLELEMENT(NAME "name", e.lname)
                                              XMLELEMENT(NAME "hire", e.hire)))
    FROM employees e
    GROUP BY dept) ;
```

id	lastname	dept	hire
1	Smith	Accounting	…
2	Martin	Shipping	…
3	Yates	Accounting	…
4	Oppenheimer	Shipping	…

DeptDoc

```
<Department name="Accounting" count="2" >
 <emp> <name>Smith</name> <hire>…</hire> </emp>
 <emp> <name>Yates</name> <hire>…</hire> </emp>
</Department>
```

```
<Department name="Shipping" count="2">
 <emp> <name>Martin</name> <hire>…</hire> </emp>
 <emp> <name>Oppenheimer</name>
          <hire>…</hire> </emp>
</Department>
```

Manipulating XML Data

- Constructor functions
  - focus on publishing SQL data as XML
  - no further manipulation of XML
- More requirements
  - how do we select or extract portions of XML data (e.g., from stored XML)?
  - how can we decompose XML into relational data?
    - XMLCAST is not sufficient
  - both require a language to identify, extract and possibly combine parts of XML values

SQL/XML utilizes the XQuery standard for this!

XMLQUERY

- Evaluates an XQuery or XPath expression
  - Provided as a character string literal
- Allows for optional arguments to be passed in
  - Zero or more named arguments
  - At most one unnamed argument can be passed in as the XQuery context item
  - Arguments can be of any predefined SQL data type incl. XML
  - Non-XML arguments will be implicitly converted using XMLCAST
- Returns a sequence of XQuery items

```
SELECT XMLQUERY(
  'for $e in $dept[@count > 3]/emp
   where $e/hire > 2004-12-31 return $e/name'
  PASSING BY REF deptDoc AS "dept"
  RETURNING SEQUENCE) AS "Name_elements"
FROM XMLDept
```

=>

Name_elements
<name>Miller</name>
<name>Smith</name> <name>Johnson</name>
<name>Martin</name>

XMLTABLE

- Transforming XML data into table format (aka "Shredding")
- Evaluates an XQuery or XPath expression – the **"row pattern"**
  - each item of result sequence is turned into a row
  - allows for optional arguments to be passed in, just like XMLQuery
- Element/attribute values are mapped to column values using path expressions (PATH) – the **"column pattern"**
  - Names and SQL data types for extracted values/columns need to be specified
  - Default values for "missing" columns can be provided
  - ORDINALITY column (sequence number) can be generated

```
SELECT X.*
FROM XMLDept d,
     XMLTABLE ('$dept/emp' PASSING d.deptDoc AS "dept"
      COLUMNS
      "#num" FOR ORDINALITY,
      "name" VARCHAR(30)          PATH 'name',
      "hire"   DATE               PATH 'hire',
      "dept"  VARCHAR(40)         PATH '../@name'
     ) AS "X"
```

#num	name	hire	dept
1	Smith	2005-01-01	Accounting
2	Yates	2002-02-01	Accounting
3	Martin	2000-05-01	Shipping

SQL Predicates on XML Type

- IS DOCUMENT
  - Checks whether an XML value conforms to the definition of a well-formed XML document
- IS CONTENT
  - Checks whether an XML value conforms to the definition of either a well-formed XML document or a well-formed external parsed entity
- IS VALID
  - Checks whether an XML value is valid according to a given XML Schema
  - Does not validate/modify the XML value; i.e., no default values are supplied.
- XMLEXISTS
  - Checks whether the result of an XQuery expression (an XQuery sequence) contains at least one XQuery item

SQL/XML Mapping Definitions

- Mapping SQL identifiers to XML Names and vice versa
  - rules for mapping regular and delimited identifiers
  - encoding/decoding of illegal character or character combinations
- Mapping SQL (built-in) data types to XML Schema types
  - best match, additional XML schema facets
  - schema annotations
- Mapping of values based on the type mappings
- Mapping of SQL tables, schemas, catalogs to XML documents
  - options for fine-tuning the XML schema structure
  - can be used to produce an XML-only "view" of a relational database
  - potential basis for XQuery over SQL data

Mapping SQL Tables to XML Documents

- The following can be mapped to an XML Document:
  - Table
  - Tables of an SQL Schema
  - Tables of an SQL Catalog
- The mapping produces an XML Document and an XML Schema Document
- These XML Documents may be physical or virtual
- The mapping of SQL Tables uses the mapping of SQL identifiers, SQL data types, and SQL values
  - either as single element with <row> subelements
  - or as sequence of elements
- Two choices for the mapping of null values:
  - nil: use xsi:nil="true"
  - absent: column element is omitted
- XML Schema that is generated
  - provides named type for every column, row, table, schema, and catalog
  - allows annotation to be included in each of these definitions

Mapping Example – Sequence of Elements

- Map the EMPLOYEE table ("sequence of elements option"):

```
<EMPLOYEE>
  <EMPNO>000010</EMPNO>
  <FIRSTNME>CHRISTINE</FIRSTNME>
  <LASTNAME>HAAS</LASTNAME>
  <BIRTHDATE>1933-08-24</BIRTHDATE>
  <SALARY>52750.00</SALARY>
</EMPLOYEE>

<EMPLOYEE>
  <EMPNO>000020</EMPNO>
  <FIRSTNME>MICHAEL</FIRSTNME>
  <LASTNAME>THOMPSON</LASTNAME>
  <BIRTHDATE>1948-02-02</BIRTHDATE>
  <SALARY>41250.00</SALARY>
</EMPLOYEE>

  ...
```

Product Support

- The "big three" support XML in SQL databases
  - IBM, Oracle implement (almost) complete support of SQL/XML
  - Microsoft supports similar capabilities using proprietary syntax
  - all three support XQuery inside SQL
  - differences in implementation of XML storage
- IBM DB2 V9 (SIGMOD2005, VLDB2005)
  - CLOB-based as well as native storage for XML values
    - efficient storage, indexing, processing techniques
  - allows to include SQL requests in XQuery expressions, too
- Oracle 10g (Oracle XML-DB technical whitepaper, VLDB2004)
  - storage based on CLOBs or object-relational tables
    - additional indexing capabilities, XML query rewrite
  - protocols (ftp, WebDAV, …) for supporting file-oriented XML storage/access
- Microsoft SQL Server 2005 (MSDN whitepaper, VLDB2005)
  - stored as BLOB in an internal format
    - primary (B+ tree) and secondary indexes, query processing based on mapping to RDM

Summary

- Increasing importance of XML in combination with data management
  - flexible exchange of relational data using XML
  - managing XML data and documents
  - trend towards "hybrid" approaches for relational DBMS
- SQL/XML standard attempts to support the following
  - "Publish" SQL query results as XML documents
  - Ability to store and retrieve (parts of) XML documents with SQL databases
  - Rules and functionality for mapping SQL constructs to and from corresponding XML concepts
- Relies partly on XQuery standard
  - XML data model
  - queries over XML data
- Broad support by major SQL DBMS vendors