University of Lübeck
Institute of Information Systems

XOBE$_{DBPL}$: A JAVA-based Database Programming Language for XML Data

Volker Linnemann
Institute of Information Systems
University of Lübeck

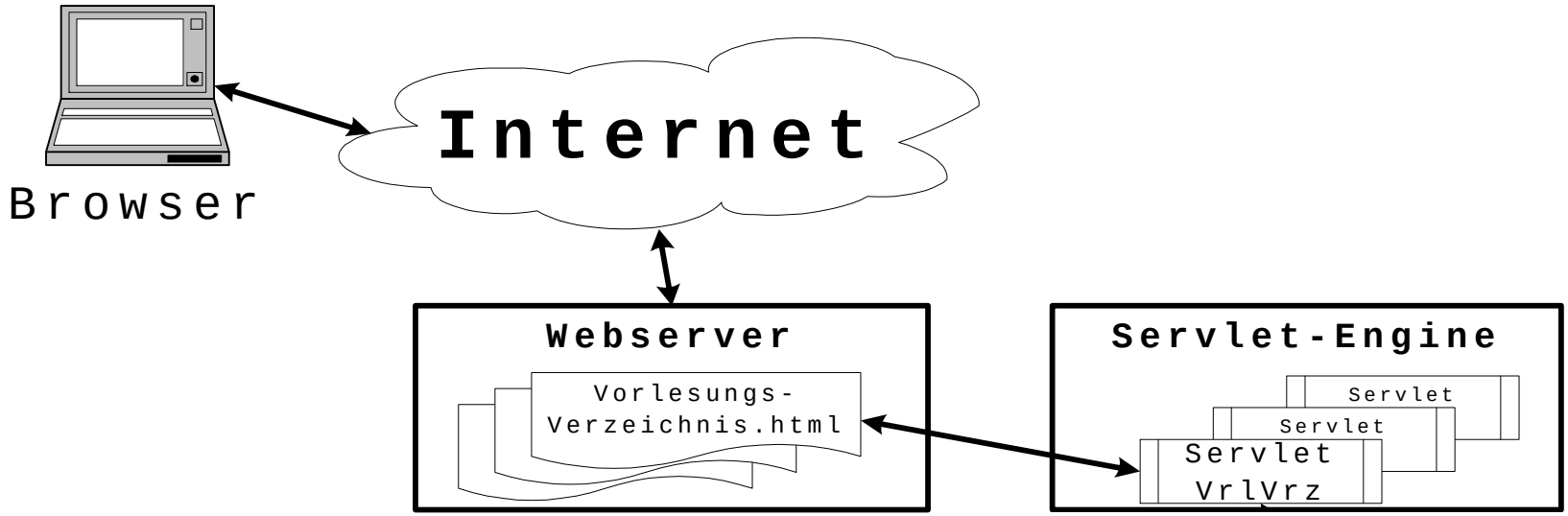
Table of Contents

1. Introduction

- For programming of web applications many tools exist, for example Java Server Pages, Java Servlets, PHP, ...
- Web applications generate HTML or XML documents dynamically
- HTML or XML documents are generated as strings

1. Introduction

Source: Kemper/Eickler



Validity of generated documents according to a DTD or an XML Scheme is checked only at runtime $\rightarrow$ extensive Testing necessary

1. Introduction

- Better: Check validity at compile time

  → only correct XML structures at runtime

  → more reliable programs

  → simple program structures

  → no dynamic tests, i.e. more efficient

  → faster program development

1. Introduction

Example: JAVA Server Page

```
<HTML>
  <HEAD> <TITLE> A Simple Server Page </TITLE> </HEAD>
  <BODY>
    <% if(timeOfDay() == "AM") { %>
      <UL>
          <LI> Good Morning </LI>
      </UL>
    <% } else { %>
      <UL>
          <LI> Good Afternoon </LI>
      </UL>
    <% } %>
  </BODY>
</HTML>
```

1. Introduction

Example: JAVA Server Page

```
<HTML>
 <HEAD> <TITLE> A Simple Server Page </TITLE> </HEAD>
 <BODY>
  <% if(timeOfDay() == "AM") { %>
   <UL>
        Good Morning        ← wrong syntax
   </UL>
  <% } else { %>                                  → runtime error
   <UL>
        Good Afternoon      ← wrong syntax
   </UL>
  <% } %>
 </BODY>
</HTML>
```

2. XML, DTDs and XML Scheme

- XML:

```
<bookstore>
   <book>
      <author> Alfons Kemper  </author>  <author> Andre Eickler   </author>
      <title>  Datenbanksysteme  </title>
   </book>
   <book>
      <author> Donald E. Knuth    </author>
      <title>  The Art of Computer Programming     </title>
   </book>
</bookstore>
```

- DTDs and XML Scheme:
  - Description Languages for XML Documents

2. XML, DTDs and XML Scheme

- DTD (Document Type Definition):

```
<!DOCTYPE bookstore [
    <!ELEMENT bookstore (author*,title)>
    <!ELEMENT author (#PCDATA)>
    <!ELEMENT title (#PCDATA)>
]>
```

2. XML, DTDs and XML Scheme

XML Scheme

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="bookstore">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="book" minOccurs="0"
                                          maxOccurs="unbounded">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="author" minOccurs="0"
                                                maxOccurs="unbounded"
                                                type = "xsd:string"/>
                     <xsd:element name="title" type = "xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

2. XML, DTDs and XML Scheme

[well formed XML data] - correct nesting of tags and elements

[valid XML data] - well formed XML data
- XML data corresponds to declared XML Schema

[dynamic validation] - validity is checked at run time
- at run time valid and unvalid XML data

[static validation] - validity is checked only once at compile time
- at run time XML data is guaranteed to be valid

3. XML Objects in XOBE

- **X**ML **OBJE**CTS XOBE
  - Programming language für XML based applications based on JAVA

- Basic Ideas:
  - Every elementtype declaration in an XML schema acts as a class definition for JAVA
  - Values : XML structures according to the underlying elementtype definition
  - Generation of values by XML constructors

3. XML Objects in XOBE

```
bookstore collectBooks() {
    XML<book*> books = <>;
    int countBooks = keyboard.readint();
    for (int i=1; i<countBooks; i++) {
        XML<author*> authors = <>;
        int countAuthors = keyboard.readint();
        for (int j=1; j<countAuthors; j++) {
            String author = keyboard.readString();
            authors = authors +
                    <author> {author} </author>;
        }
        String title = keyboard.readString();
        book b = <book>
                   {authors}
                   <title>  {title}  </title>
                 </book>;
        books = books + b;
    }
    return
        <bookstore>
            { books }
        </bookstore>;
}
```

XML Object Constructor

4. The XOBE Type System

- Formalization of XML by Regular Hedge Grammars

- B: Basic Types **`int,string`**
- E: Element Names **`bookstore,book,author,title`**
- T = B ∪ E: Terminal Symbols
- N: Nonterminal Symbols *bookstore*, *book*, *author*, *title*
- s : Starting Expression *bookstore*
- P: Regular Production Rules

Nonterminal Symbol → Regular Hedge Expression

$$bookstore \rightarrow \texttt{bookstore}[book*]$$
$$author \rightarrow \texttt{author}[\texttt{string}]$$
$$book \rightarrow \texttt{book}[author* \,;\, title]$$
$$title \rightarrow \texttt{title}[\texttt{string}]$$

Recursive nonterminals only at the end of an expression

4. The XOBE Type System

More examples for regular hedge expressions:

$$\mathbf{book}[author\ ;\ author*\ ;\ author\ ;\ title]$$

$$\mathbf{book}[title]|\mathbf{book}[author;\ title]$$

$L(r)$: language generated by a regular hedge expression r

Regular Inequality:

$$r \leq s \Leftrightarrow L(r) \subseteq L(s)$$

Examples of regular inequalities:

$$\mathbf{book}[author*\ ;\ title] \leq \mathbf{book}[author*\ ;\ title]$$

$$\mathbf{book}[author\ ;\ author*\ ;\ author\ ;\ title] \leq \mathbf{book}[author*\ ;\ title]$$

4. The XOBE Type System

```
bookstore collectBooks() {
    XML<book*> books = <>;
    int countBooks = keyboard.readint();
    for (int i=1; i<countBooks; i++) {
        XML<author*> authors = <>;
        int countAuthors = keyboard.readint();
        for (int j=1; j<countAuthors; j++) {
            String author = keyboard.readString();
            authors = authors +
                    <author> {author} </author>;
        }
        String title = keyboard.readString();
        book b = <book>
                    {authors}
                    <title>  {title}  </title>
                 </book>;
        books = books + b;
    }
    return
        <bookstore>
            { books }
        </bookstore>;
}
```

Inferred type of the left hand side:

$$book$$

Inferred type of the right hand side:

$$\mathtt{book}[author* \,;\, \mathtt{title}[\mathtt{string}]]$$

Inequality to be proven:

$$\mathtt{book}[author* \,;\, \mathtt{title}[\mathtt{string}]] \;\leq\; book$$

-> 5. Implementation of XOBE

4. The XOBE Type System

Leading Terminal Symbols of Regular Hedge Expressions:

$$term(\texttt{author[string]};authors^*) = \{\ \texttt{author}\ \}$$

$$term(authors^*;title) = \{\ \texttt{author}, \texttt{title}\ \}$$

4. The XOBE Type System

Partial Derivatives of Reg. Hedge Exp. With Respect to a Terminal Symbol:

Set of Pairs:

1st component: Content of an element after removing the given leading terminal symbol

2nd component: Expression for the remainder

$$der_{\texttt{author}}(\texttt{author}[\texttt{string}]; author*) = \{(\texttt{string}, author*)\}$$

$$der_{\texttt{book}}(\texttt{book}[title]|\texttt{book}[author; title]) = \{(title, \epsilon), (author; title, \epsilon)\}$$

$$der_{\texttt{author}}((\texttt{author}[\texttt{string}]; title)*) =$$

$$\{(\texttt{string}, title; (\texttt{author}[\texttt{string}]; title)*)\}$$

4. The XOBE Type System

Partial Derivatives of Reg. Hedge Exp. With Respect to a Terminal Symbol:

$$der_{\texttt{author}}(author*; title) = \{(\texttt{string}, author*; title)\}$$

$$der_{\texttt{author}}(author; author*; title|title) = \{(\texttt{string}, author*; title)\}$$

$$der_{\texttt{title}}(author*; title) = \{(\texttt{string}, \epsilon)\}$$

$$der_{\texttt{title}}(author; author*; title|title) = \{(\texttt{string}, \epsilon)\}$$

4. The XOBE Type System

Partial Derivatives of Reg. Hedge Exp. With Respect to a Terminal Symbol:

General case: (*Reg*: Set of all Regular Hedge Expressions)

$$part_x(r \leq s) = \{\ (c_r \leq \Big|_{i \in I} c_s^i)\ \vee\ (r_r \leq \Big|_{i \in \overline{I}} r_s^i) \mid (c_r, r_r) \in der_x(r)\ \wedge\ der_x(s) = \{(c_s^1, r_s^1), \ldots, (c_s^n, r_s^n)\}\ \wedge\ I\ \in \mathcal{P}(\{1, \ldots, n\})\ \wedge\ \overline{I} = \{1, \ldots, n\} \backslash I\ \}$$

$$c_r, r_r, c_s^i, r_s^i \in Reg \qquad \mathcal{P}(I) = \{J \mid J \subseteq I\}$$

Special case: $der_x(r) = \{\ (c_r, r_r)\ \}$ and $der_x(s) = \{\ (c_s, r_s)\ \}$:

$$part_x(r \leq s) = \{\ (c_r \leq c_s) \vee (r_r \leq \emptyset), (c_r \leq \emptyset) \vee (r_r \leq r_s)\ \}$$

4. The XOBE Type System

Partial Derivatives of Reg. Hedge Exp. With Respect to a Terminal Symbol:

$$part_{\texttt{author}}(author*; title \leq author; author*; title|title) =$$

$$\{ \quad (\texttt{string} \leq \texttt{string} \vee author*; title \leq \emptyset), \qquad (1)$$
$$(\texttt{string} \leq \emptyset \vee author*; title \leq author*; title) \ \} \qquad (2)$$

$$part_{\texttt{title}}(author*; title \leq author; author*; title|title) =$$

$$\{ \quad (\texttt{string} \leq \texttt{string} \vee \epsilon \leq \emptyset), \qquad (3)$$
$$(\texttt{string} \leq \emptyset \vee \epsilon \leq \epsilon) \ \} \qquad (4)$$

4. The XOBE Type System

The predicate

$$isnullable? \ : \ Reg \rightarrow \{\textbf{true},\textbf{false}\}$$

decides $\epsilon \in L(r)$ for $r \in Reg$

A regular inequality $r \leq s$ is called *trivially inconsistent* if *inc*($r \leq s$) holds with:

$$inc(r \leq s) = (isNullable?(r) \wedge \neg isNullable?(s))$$

4. The XOBE Type System

Subtype Algorithm

$$\frac{r \leq s \in \Gamma}{\Gamma \vdash r \leq s \Rightarrow \Gamma} \quad \text{(HYP)}$$

$$\frac{\begin{array}{c} r \leq s \notin \Gamma, \\ \Gamma \cup \{r \leq s\} \vdash^* r \leq s \Rightarrow \Gamma' \end{array}}{\Gamma \vdash r \leq s \Rightarrow \Gamma'} \quad \text{(ASSUM)}$$

$$\frac{\begin{array}{c} \neg inc(r \leq s), \\ \text{for all } x \in term(r) \text{ and for all } i \in \{1, \ldots, k\} \text{ with } part_x(r \leq s) = \\ \{(c_{r,1} \leq c_{s,1} \vee r_{r,1} \leq r_{s,1}), \ldots, (c_{r,k} \leq c_{s,k} \vee r_{r,k} \leq r_{s,k})\} \text{ is} \\ \Gamma_{i-1} \vdash c_{r,i} \leq c_{s,i} \Rightarrow \Gamma_i \vee \Gamma_{i-1} \vdash r_{r,i} \leq r_{s,i} \Rightarrow \Gamma_i \end{array}}{\Gamma_0 \vdash^* r \leq s \Rightarrow \Gamma_k} \quad \text{(REC)}$$

4. The XOBE Type System

Course of the Algorithm for $author*; title \leq author; author*; title|title$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\text{REC}}{\Gamma_7 \vdash^* \epsilon \leq \epsilon \Rightarrow \Gamma_3}}{\Gamma_6 \vdash \epsilon \leq \epsilon \Rightarrow \Gamma_3} \vee \dfrac{Error}{\Gamma_6 \vdash \epsilon \leq \emptyset}, \quad \dfrac{Error}{\Gamma_3 \vdash \epsilon \leq \emptyset} \vee \dfrac{\text{HYP}}{\Gamma_3 \vdash \epsilon \leq \epsilon \Rightarrow \Gamma_3}}{\Gamma_6 \vdash^* \texttt{string} \leq \texttt{string} \Rightarrow \Gamma_3}}{\Gamma_1 \vdash \texttt{string} \leq \texttt{string} \Rightarrow \Gamma_3} \vee \dfrac{\begin{array}{c} Error \\ \vdots \end{array}}{\Gamma_1 \vdash author*; title \leq \emptyset}, \qquad (1)
$$

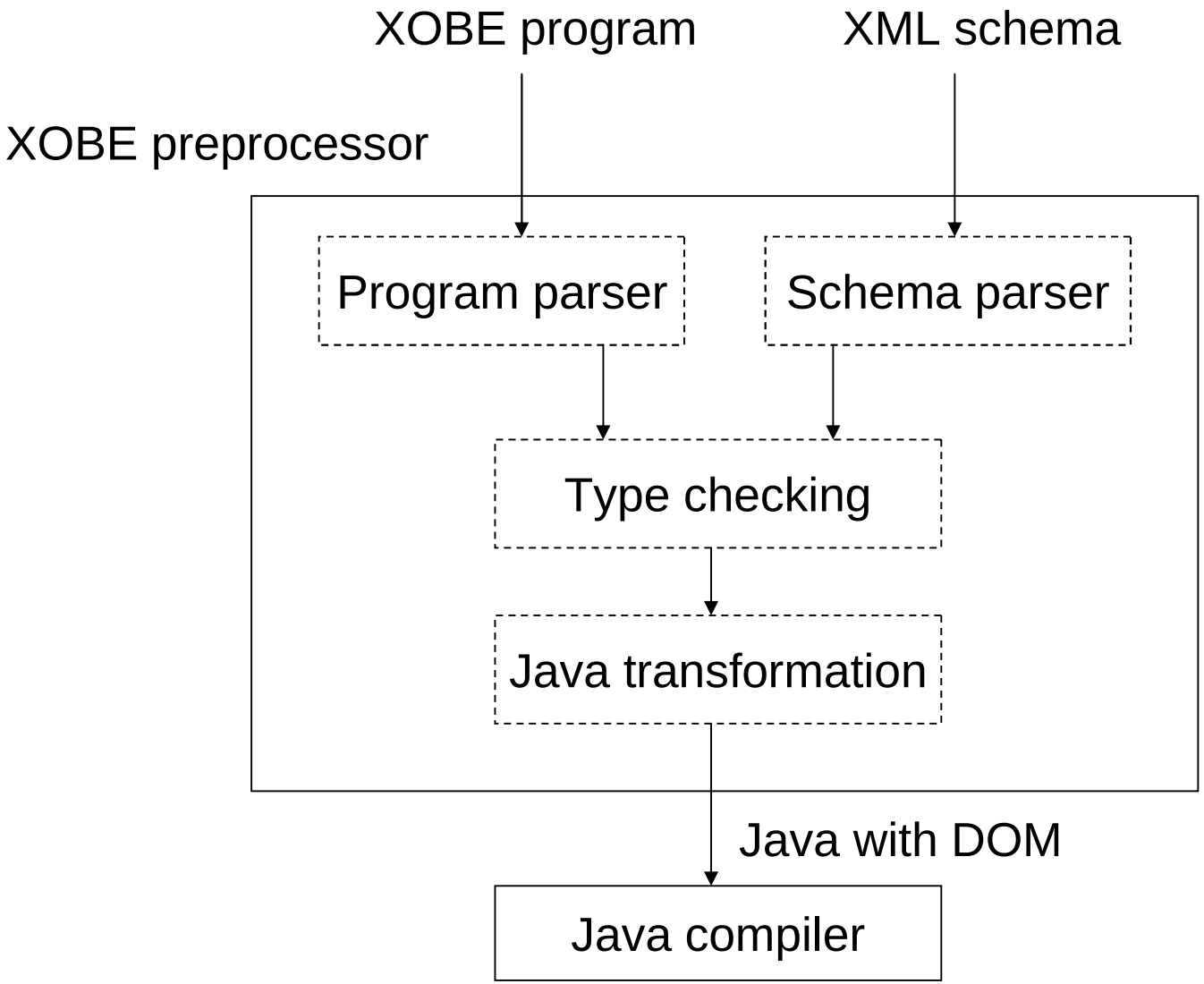
$$
\dfrac{Error}{\Gamma_3 \vdash \texttt{string} \leq \emptyset} \vee \dfrac{\begin{array}{c} \text{HYP} \\ \vdots \end{array}}{\Gamma_3 \vdash author*; title \leq author*; title \Rightarrow \Gamma_5}, \qquad (2)
$$

$$
\dfrac{\text{HYP}}{\Gamma_5 \vdash \texttt{string} \leq \texttt{string} \Rightarrow \Gamma_5} \vee \dfrac{Error}{\Gamma_5 \vdash \epsilon \leq \emptyset}, \qquad (3)
$$

$$
\dfrac{Error}{\Gamma_5 \vdash \texttt{string} \leq \emptyset} \vee \dfrac{\text{HYP}}{\Gamma_5 \vdash \epsilon \leq \epsilon \Rightarrow \Gamma_5} \qquad (4)
$$

$$
\dfrac{\dfrac{(1),\ (2),\ (3),\ (4)}{\Gamma_1 \vdash^* author*; title \leq author; author*; title|title \Rightarrow \Gamma_5}}{\Gamma_0 \vdash author*; title \leq author; author*; title|title \Rightarrow \Gamma_5}
$$

5. Implementation of XOBE



6. Applications of XOBE

- Web based Applications developed by using XOBE:
  - Real Estate Management System
  - Assignments Management System

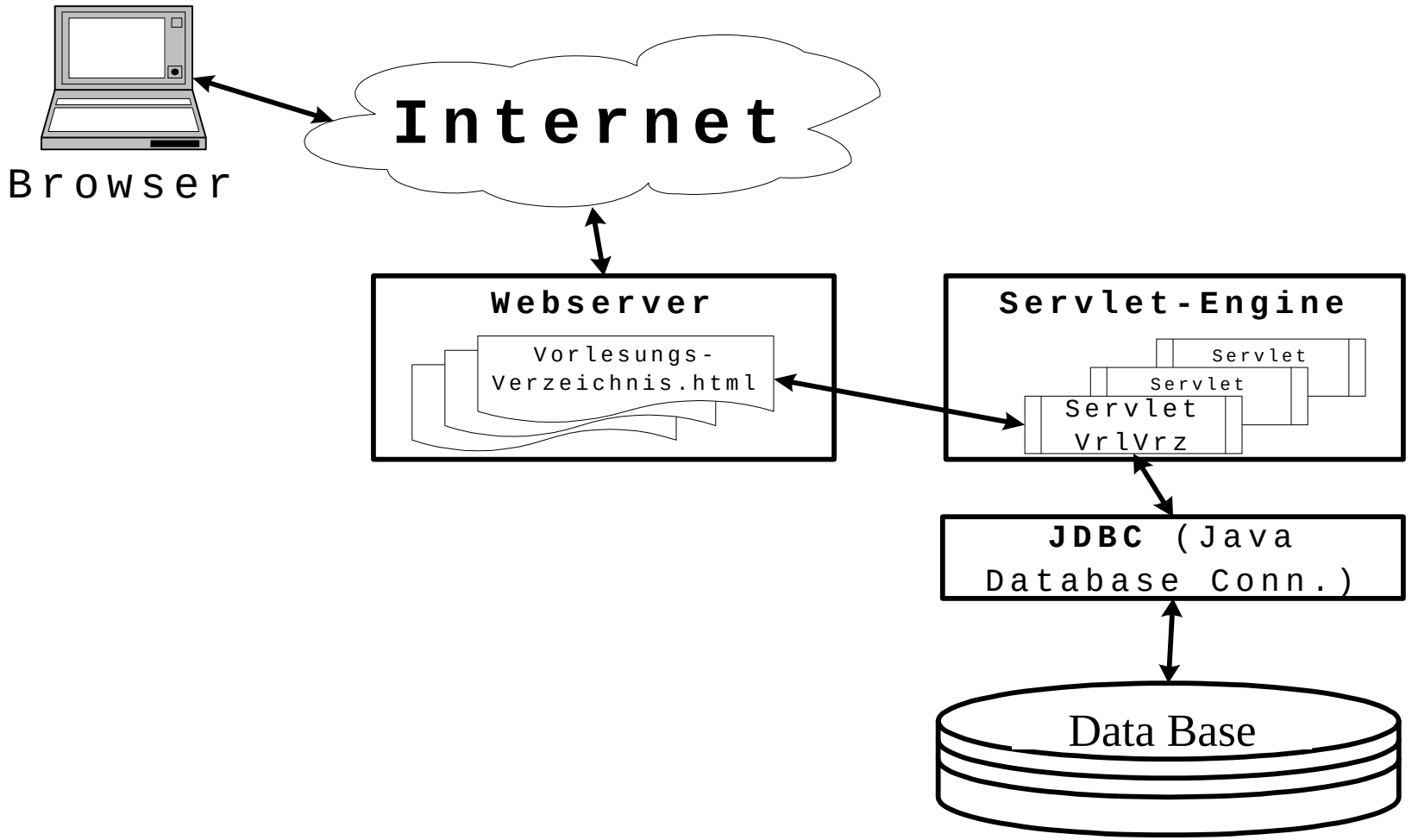
7. Persistent Data in the Web

- Web Applications with persistent relational data:
  - Servlet with JDBC-Connection to relational data base (JDBC = Java Data Base Connectivity)

7. Persistent Data in the Web

Relational Data:

Source: Kemper/Eickler

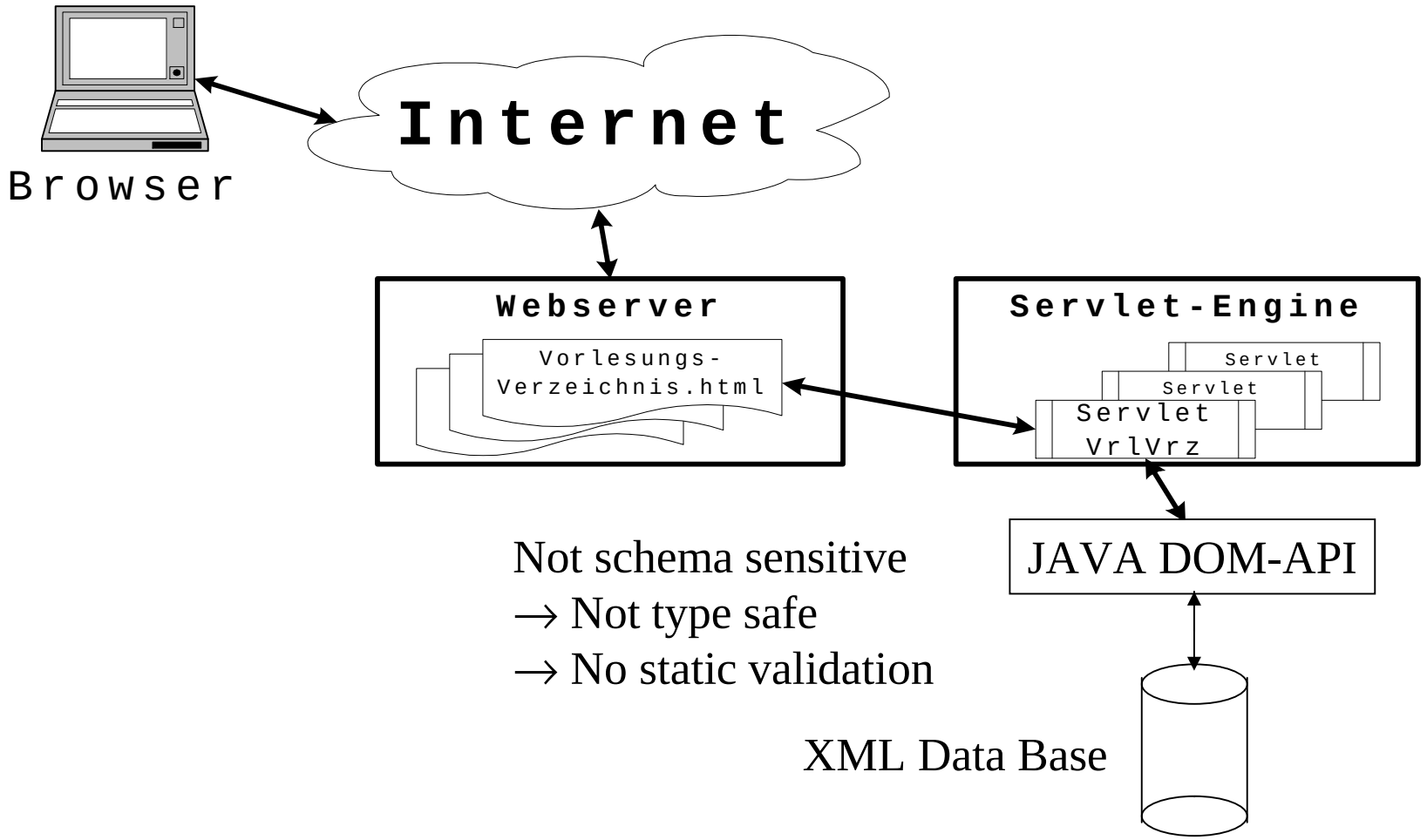


7. Persistent Data in the Web

- Web Applications with persistent XML data:
  - Servlet with DOM (Document Object Model) and plain XML-Files or native XML data base
  - DOM:
    - XML data modelled as a tree
    - DOM provides methods to traverse the XML tree
    - DOM is **not** type safe, i.e. it is not schema sensitive
    - DOM does not allow static validation

7. Persistent Data in the Web

XML Data with DOM :



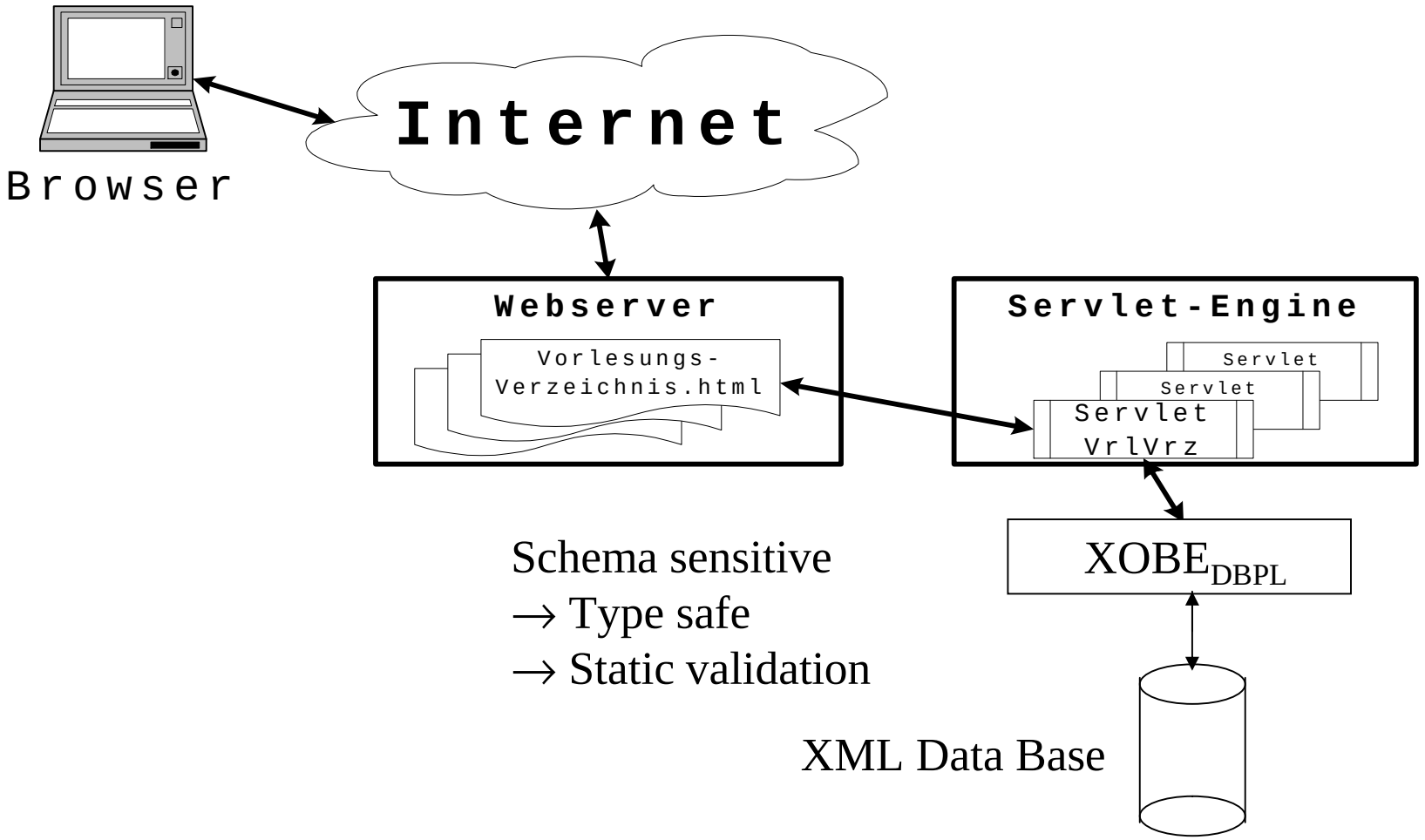
Not schema sensitive
→ Not type safe
→ No static validation

7. Persistent Data in the Web

- Web Applications with persistent XML data:
  - Servlet written in $XOBE_{DBPL}$
    - Persistent type safe XML data
    - Validated statically

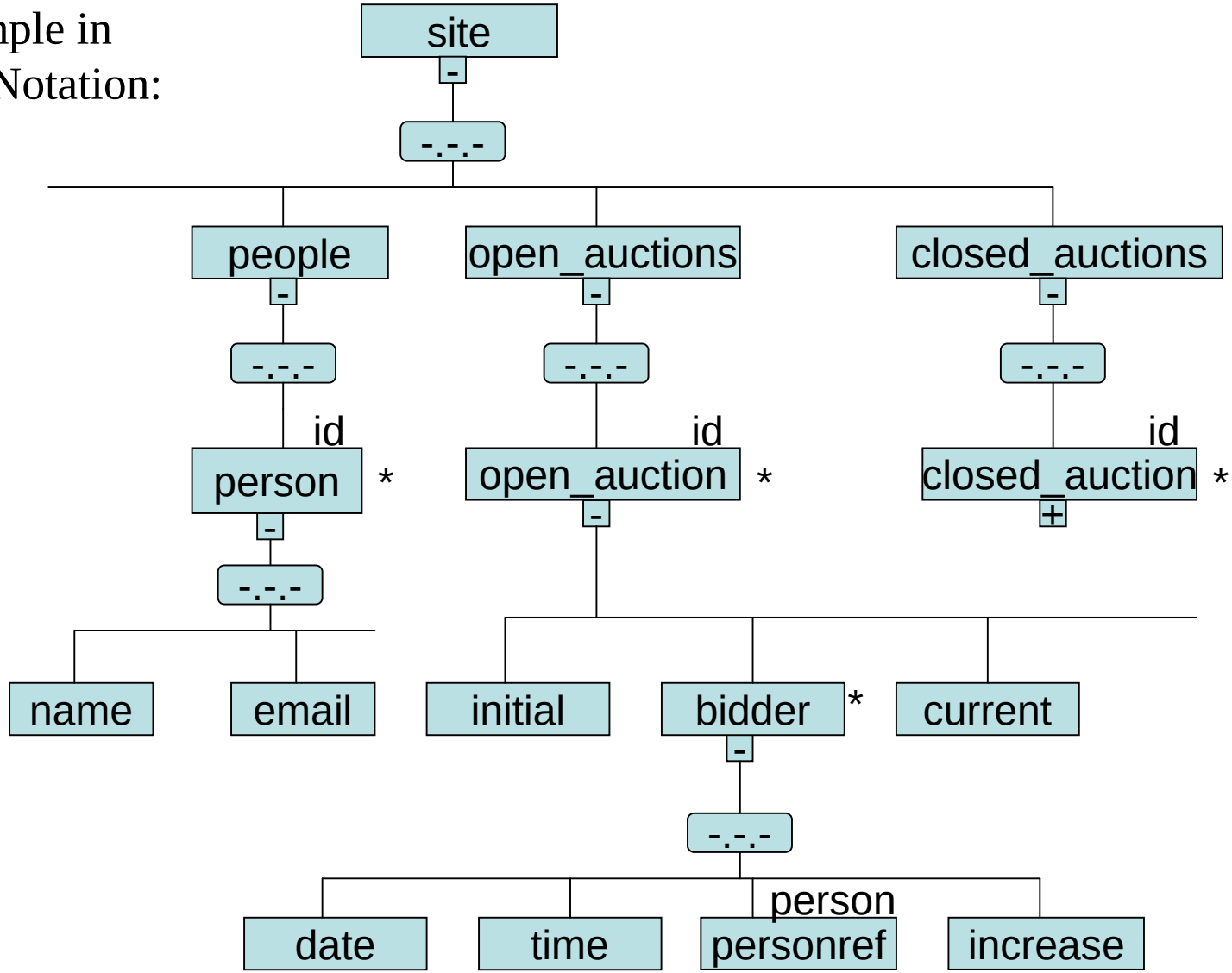
7. Persistent Data in the Web

XML Data with $XOBE_{DBPL}$:



8. XOBE$_{DBPL}$

XML Example in
XML-Spy Notation:



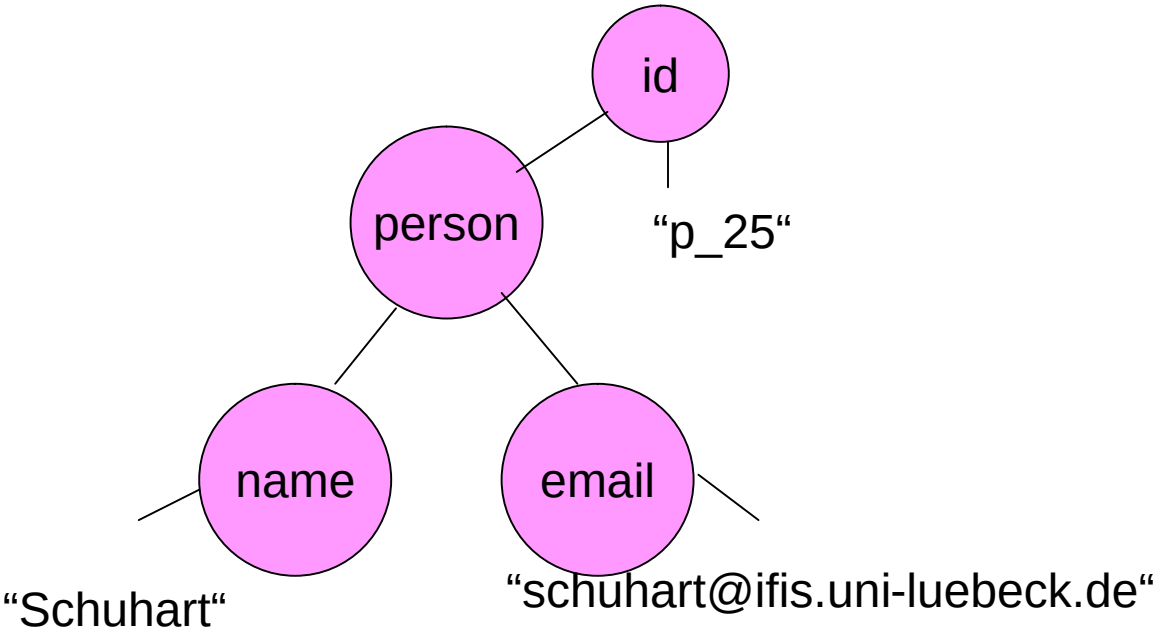
8. XOBE$_{DBPL}$ Example constructor

```
ximport auction.xsd;

public class XMarkTest{

  ...

  public person createPerson(String p_name, String p_email, String p_id){

     person p = <person id={p_id}>
                    <name>{p_name}</name>
                    <email>{p_email}</email>
                </person>;

    return p;
  }

}
```

8. $XOBE_{DBPL}$ Example constructor (2)

- **createPerson(“Schuhart“,“schuhart@ifis.uni-luebeck.de“,“p_25“);**

- **<person id=“p_25“>**
  <name>Schuhart</name>
  <email>schuhart@ifis.uni-luebeck.de</email>
  </person>



8. XOBE$_{DBPL}$ stat. Validation constructor

person p = <person id = {p_id} >
<name> {p_name} </name>
<email> {p_email} </email>
</person>;

- the inferred type

person[@id[string] ,name[string] ,email[string]]

- test whether inferred type is a valid subtype of the schema-type person

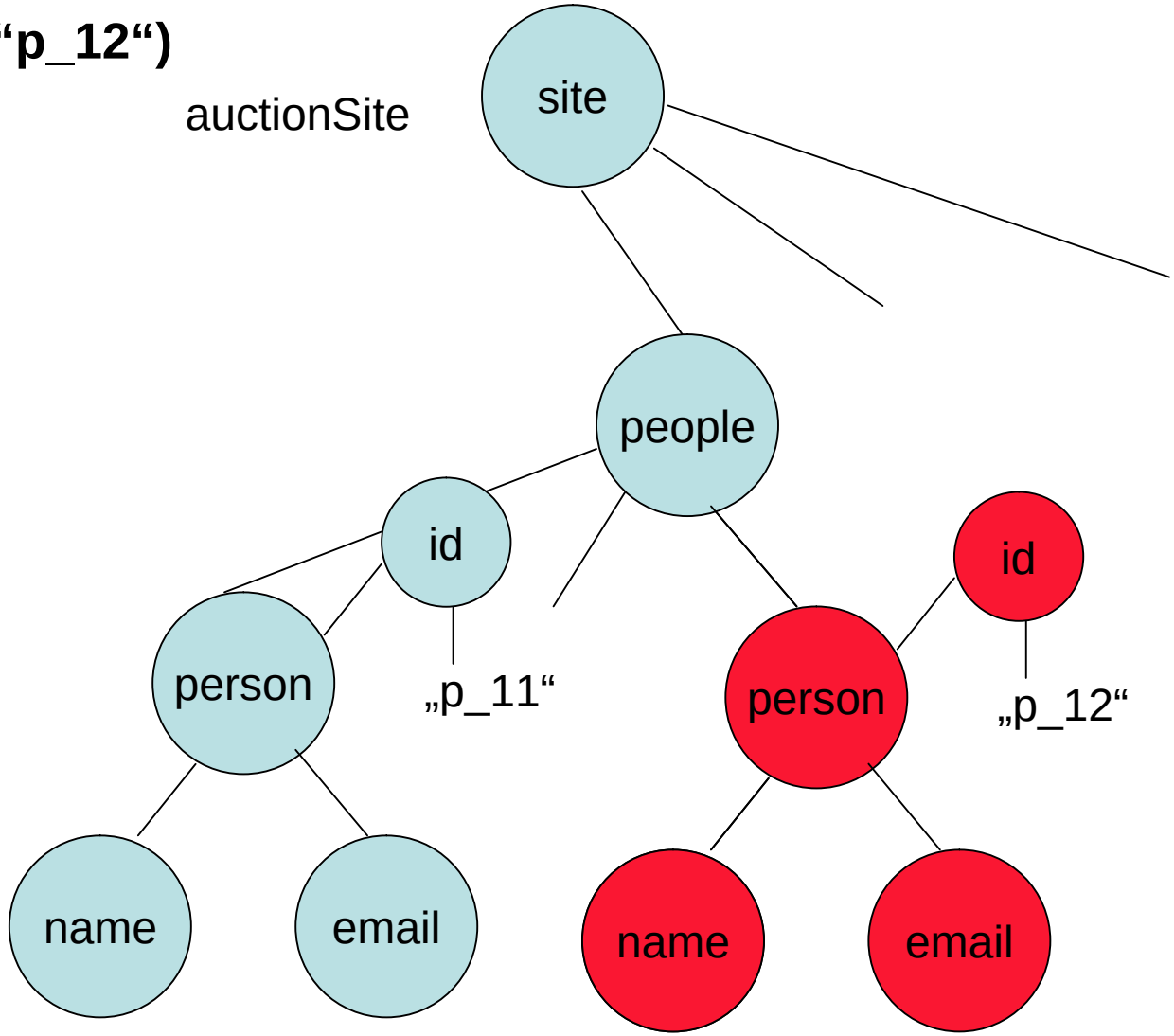
person **à** person[@id[string],name[string],email[string]]

8. XOBE$_{DBPL}$ Example for XPath

```
ximport auction.xsd;

public class XMarkTest{

  xml<site> auctionSite;
  …

  public boolean personExists(String p_id){

    xml<person*> list = $auctionSite//person[@id={p_id}]$;

    if(list.getLength()>0) return true;
    else return false;
  }

}
```

8. XOBE$_{DBPL}$ Example for Xpath

- **personExists(“p_12“)**



8. XOBE$_{DBPL}$ Stat. Validation XPath

xml<person*> list = $ auctionSite //person [@id={p_id}] $;

- Formalized types for corresponding schema types

site **à** **site**[*people*,*open_auctions,closed_auctions*]

people **à** **people**[*person**]

- inferred type

person*

- Test whether inferred type is a valid subtype of the declared type xml<person*>

8. $XOBE_{DBPL}$ Stat. Validation XPath

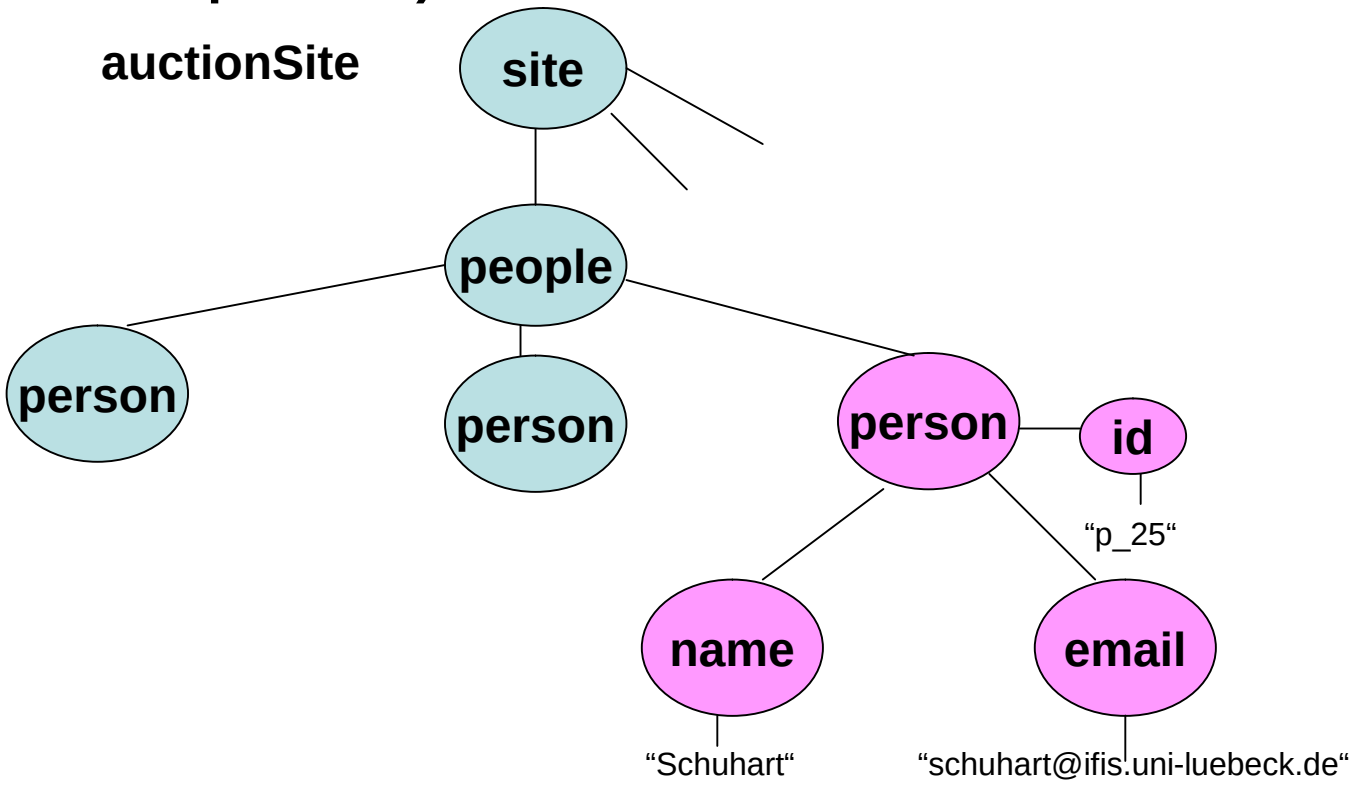
- So far: Only XML generation and queries
- No update of existing XML data
- Only transient XML Objects
- Full Data Base Programming Language:
- à Integration of XML Updates in Java
- à Static validation of XML Updates
- à Only valid XML data at runtime
- à Persistency
- à Transactions

8. XOBE$_{DBPL}$ Example Insert

```
ximport auction.xsd;

public class XMarkTest{

  xml<site> auctionSite;
  …

  public void registerPerson(person p){
    $
     UPDATE auctionSite INSERT {p} INTO auctionSite/people
    $;
  }

}
```

8. $XOBE_{DBPL}$ Example Insert (2)

- registerPerson(<person id="p_25">
  <name>Schuhart</name>
  <email>schuhart@ifis.uni-luebeck.de</email>
  </person>)

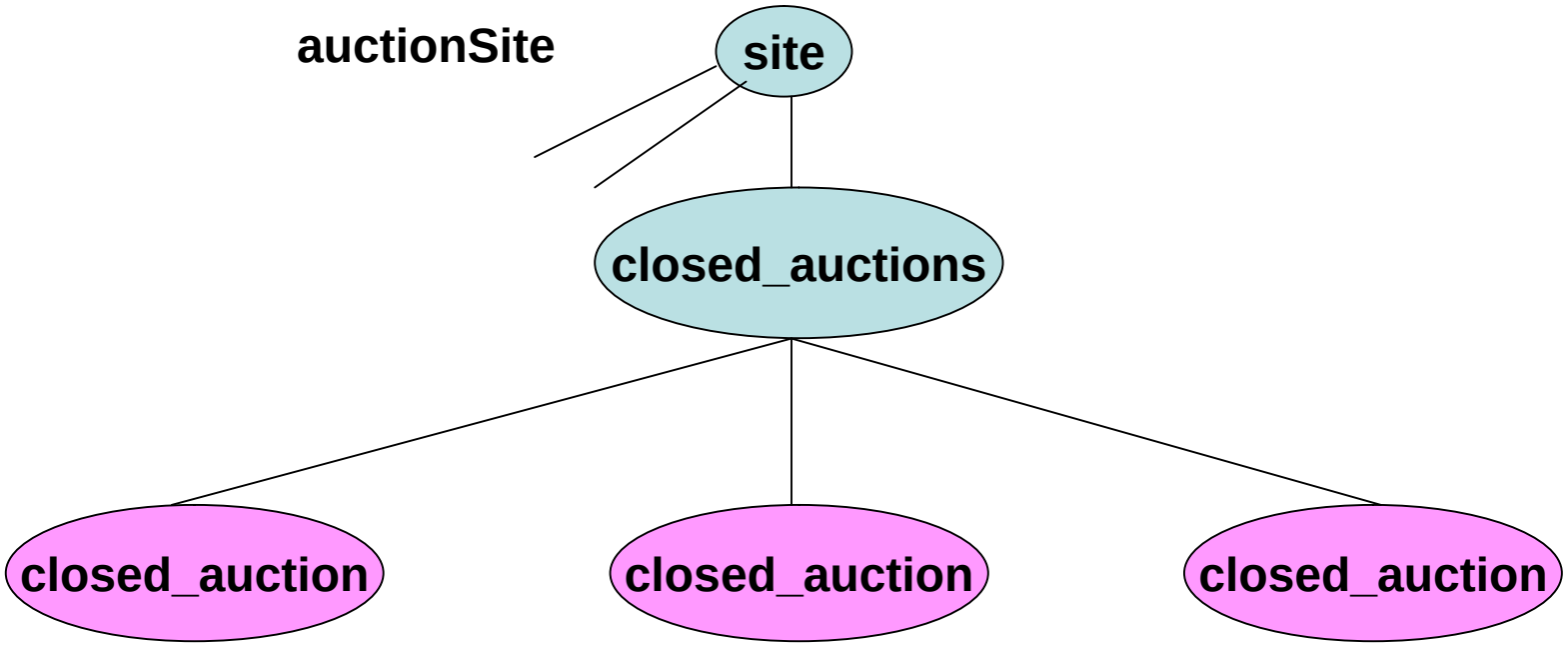


8. XOBE$_{DBPL}$ Example Delete

```
ximport auction.xsd;

public class XMarkTest{

  xml<site> auctionSite;
  …

  public void deleteClosedAuctions(){
    $
     UPDATE auctionSite DELETE auctionSite//closed_auction
    $;
  }

}
```

8. XOBE$_{DBPL}$ Example Delete (2)

- **deleteClosedAuctions()**

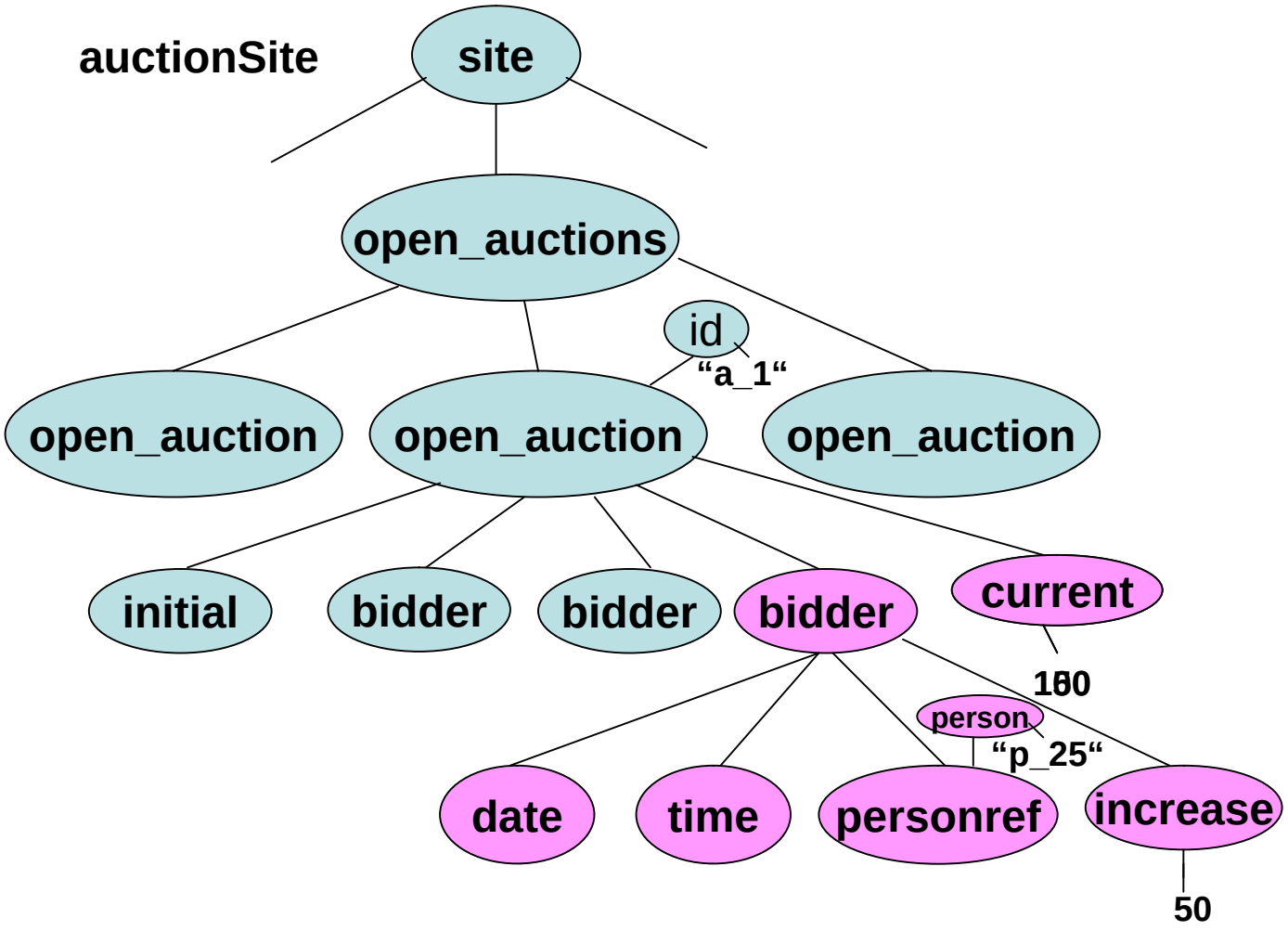


8. $XOBE_{DBPL}$ Example Update

```
ximport auction.xsd;

public class XMarkTest{

  xml<site> auctionSite;
  …

  public int bid(String p_id, int incr, String a_id){
    xml<current> n_cur = calculateCurrent(incr,a_id);
    xml<bidder> n_bidder = createBidder(p_id,incr);
    $
      UPDATE auctionSite
        INSERT {n_bidder} INTO auctionSite//open_auction[@id={a_id}] ,
        REPLACE auctionSite//open_auction[@id={a_id}]/current WITH {n_cur}
    $;
  }

}
```

8. XOBE$_{DBPL}$ Example Update (2)

- bid(“p_25“,50,“a_1“)



8. XOBE$_{DBPL}$ Static Validation Update

$**UPDATE** auctionSite **DELETE** auctionSite//closed_auction $;

- formalized types corresponding to the schema

site **à site**[*people, open_auctions, closed_auctions*]

closed_auctions **à closed_auctions**[*closed_auction**]

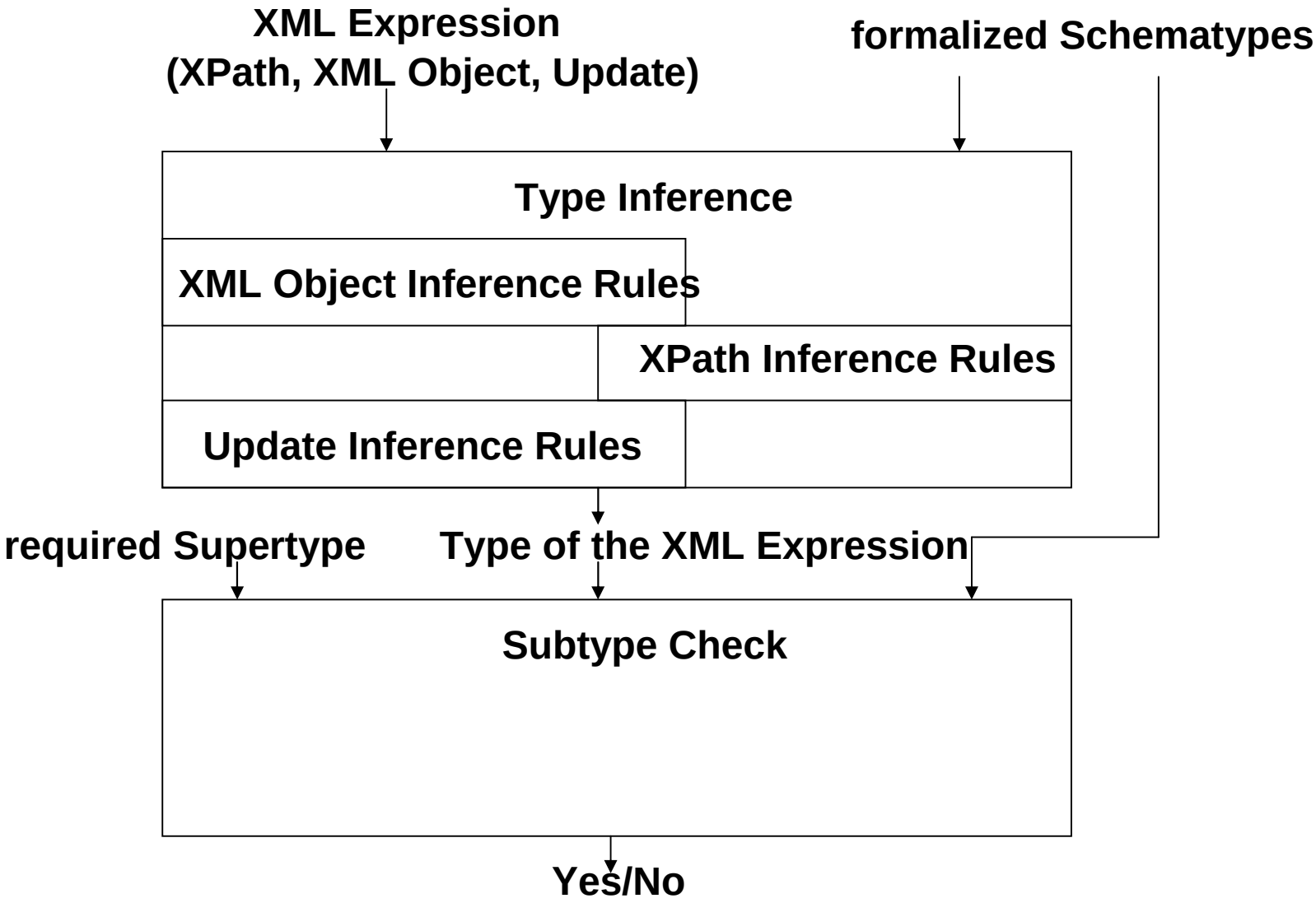
- the inferred type

site[*people, open_auctions,* **closed_auctions**[ ]]

test whether inferred type is a valid subtype of the schema-type *site*

site **à site**[*people, open_auctions*, **closed_auctions**[*closed_auction**]]

8. $XOBE_{DBPL}$ Implementation Typecheck



8. XOBE$_{DBPL}$ Persistency

- Data base: set of objects of a persistent class
- Keyword **database** instead of **class**

8. $XOBE_{DBPL}$ Persistency

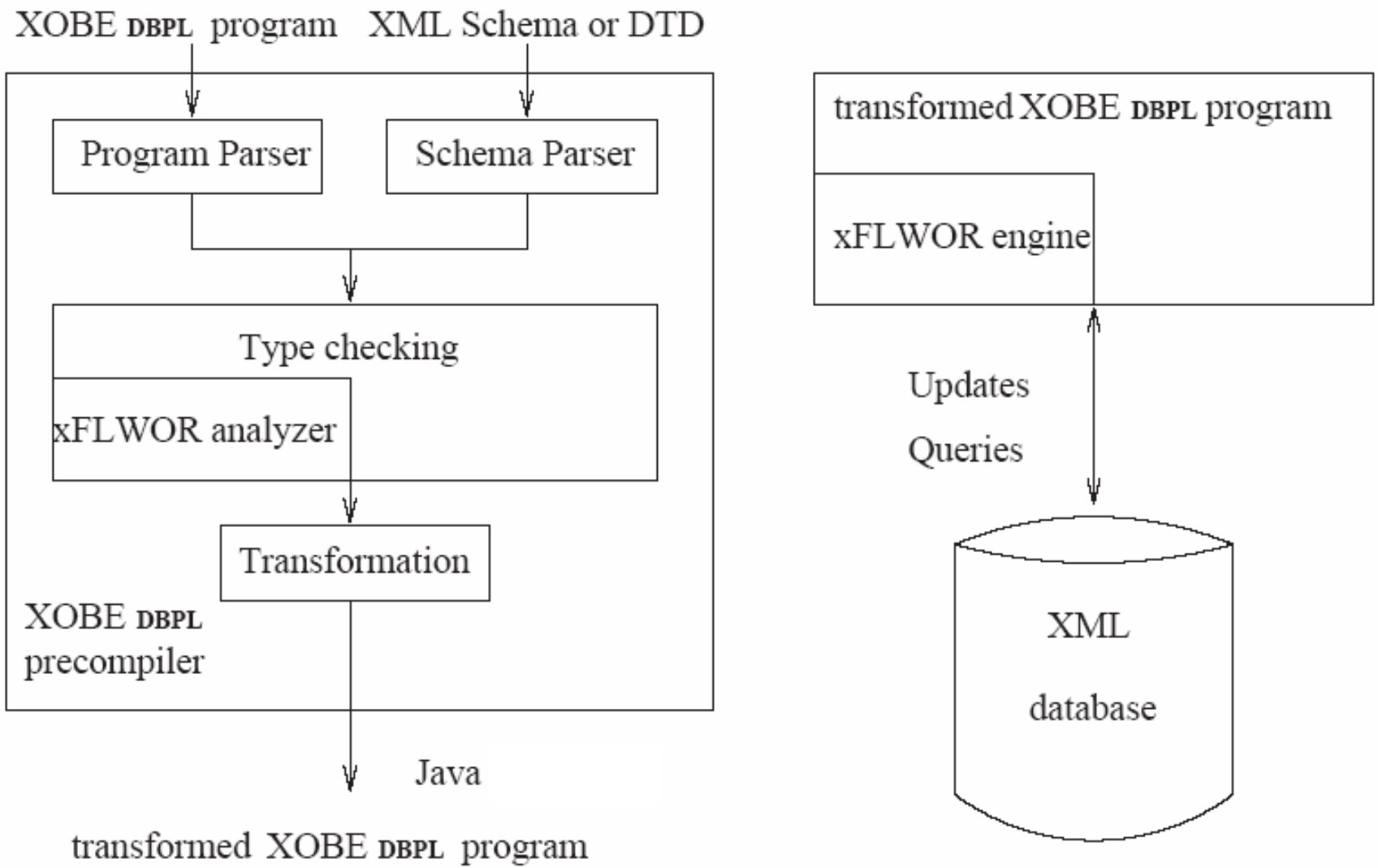
```
public database AuctionSite
   public site auctionSite;
   public String description;

   public AuctionSite ( String description) {
   …
   }
   …
   public boolean registerPerson(…) {
    …
   }
   …
}
```

8. $XOBE_{DBPL}$ Persistency

```
// create a new persistent Auctionsite object
   AuctionSite auction =
      new AuctionSite("an example");
   ...

// search for specific auction sites
   String description = "an example";
   List auctionSites =
      $AuctionSite[/description={description}]$;
...
```

Denotes the set of all persistent AuctionSite objects

8. $XOBE_{DBPL}$ Implementation



9. Concluding Remarks

- **XOBE$_{DBPL}$** is a data base programming language für XML-based applications
- Static type checking for XML Objects und XML Operations
- **XOBE$_{DBPL}$** guarantees to work only with valid XML data
- Not mentioned in this talk: transactions

9. Concluding Remarks

People:

- **PhD-Dissertations:**
  - Dr.rer.nat. Martin Kempa (XOBE)
  - Dr.rer.nat. Henrike Schuhart ($XOBE_{DBPL}$)
- **Master Theses**
  - Dipl.-Inf. Jens Kramer (Real Estate Management)
  - Dipl.-Inf. Torben Spiegler (Assignment Management)
  - Dipl.-Inf. Martin Lipphardt (Efficient Object Model)
- …

University of Lübeck
Institute of Information Systems

XOBE$_{DBPL}$: A JAVA-based Database Programming Language for XML Data

Volker Linnemann
Institute of Information Systems
University of Lübeck