

# Tamino

## The NATIVE XML Server

Tony Yip

Software AG (Hong Kong) Ltd.

# Content

---

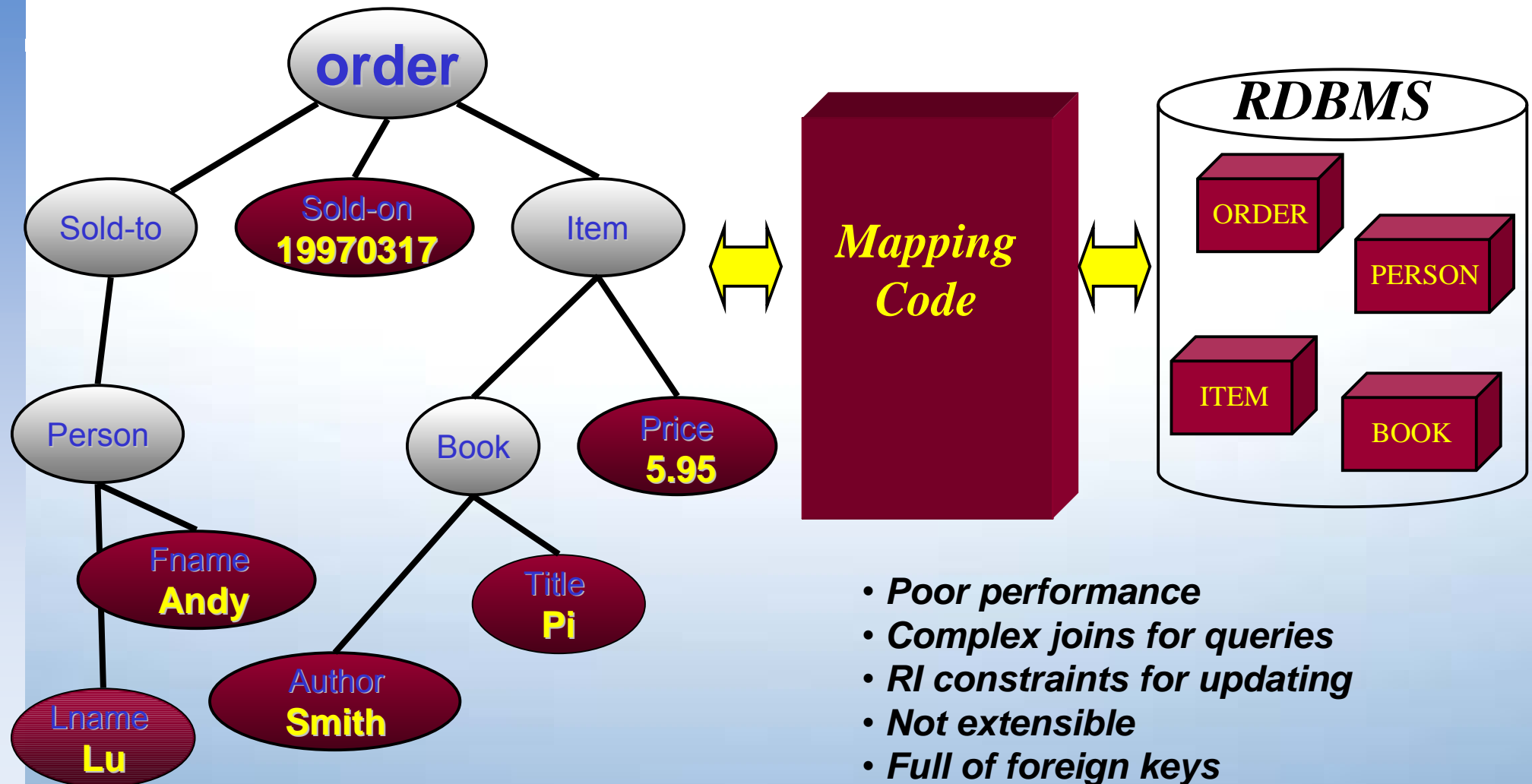
- **Tamino Architecture & Features**
- **Comparison with RDBMS**
- **Usage of Tamino**
- **Demonstration**

# Tamino XML Server

- Supports Internet and W3C standards
- Stores/Retrieves native XML documents and non-XML
- Offers full text search on XML documents
- Accesses Rdbms databases
- Integrates with other business applications
- Supports any programming language
- Works with major Web servers and Applications servers

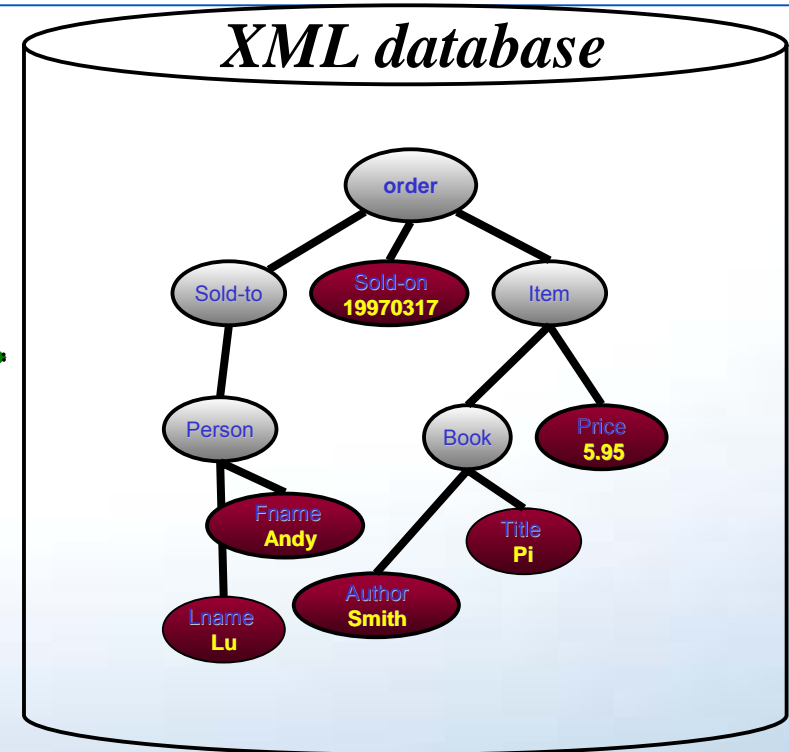
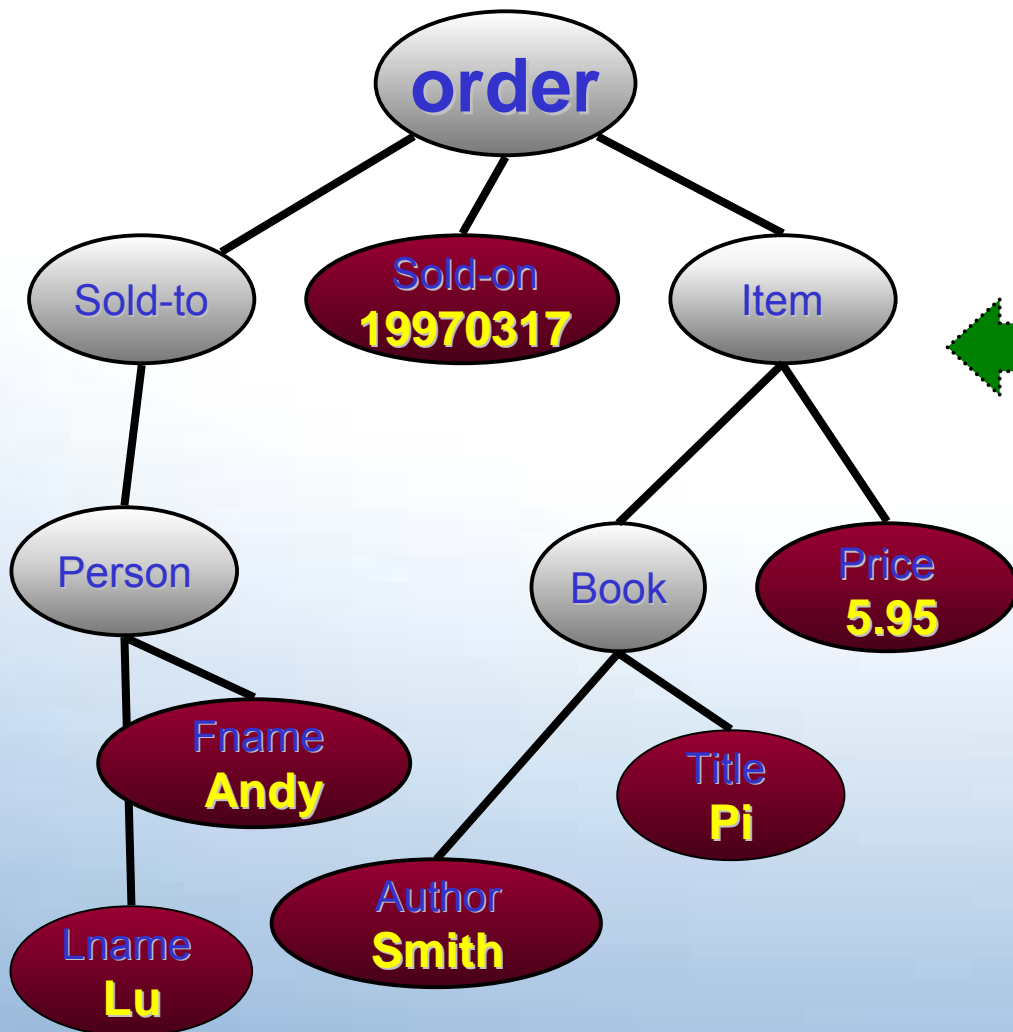


# Transformation is a Slow Expensive Process



- *Poor performance*
- *Complex joins for queries*
- *RI constraints for updating*
- *Not extensible*
- *Full of foreign keys*

# XML Data Works Best in a Hierarchical Database



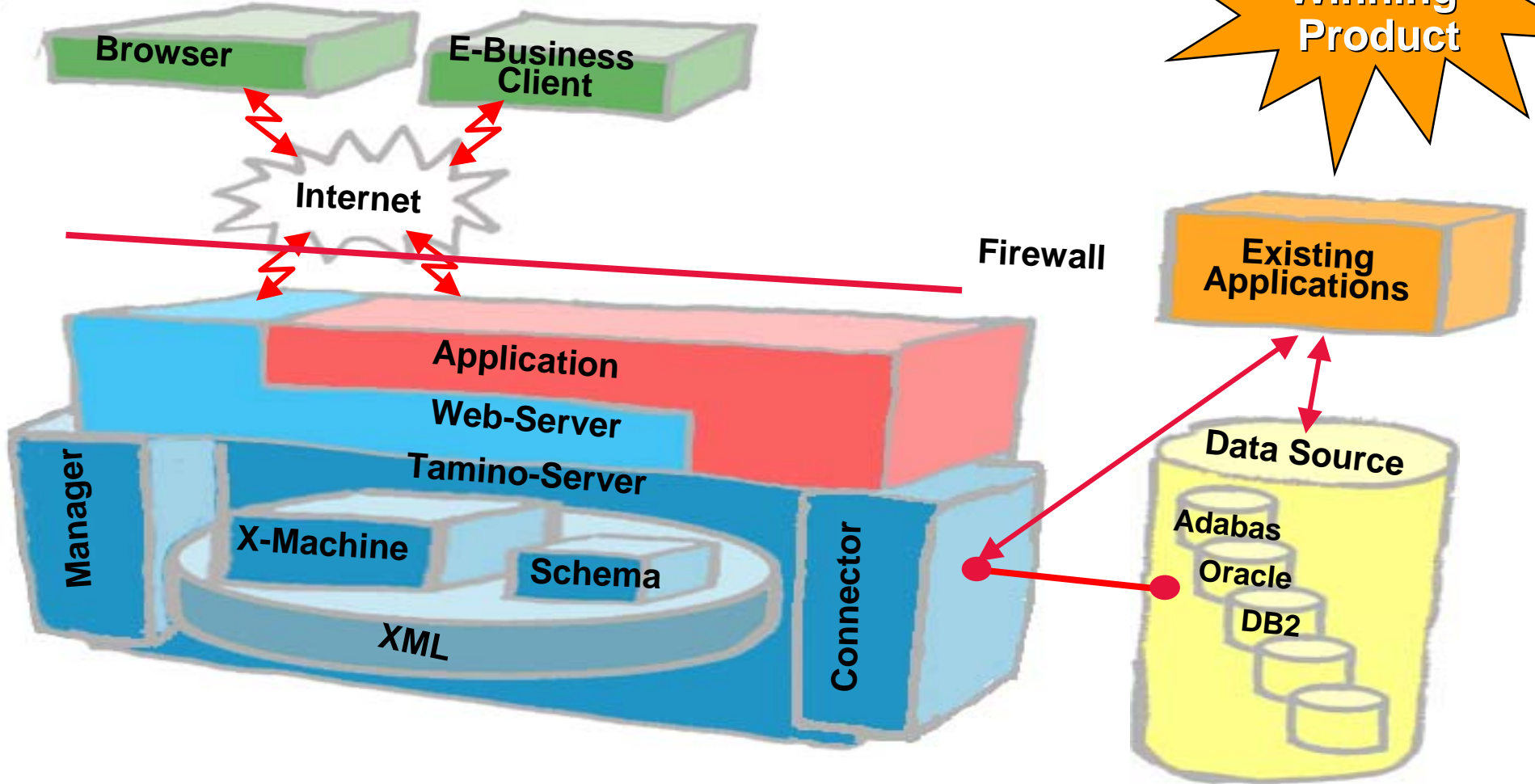
- Database schema automatically built from DTD
- No Mapping required
- Scalability
- High Performance



# Tamino

## XML Database

## Architecture



# **Tamino** Features & Benefits

## **XML Database**

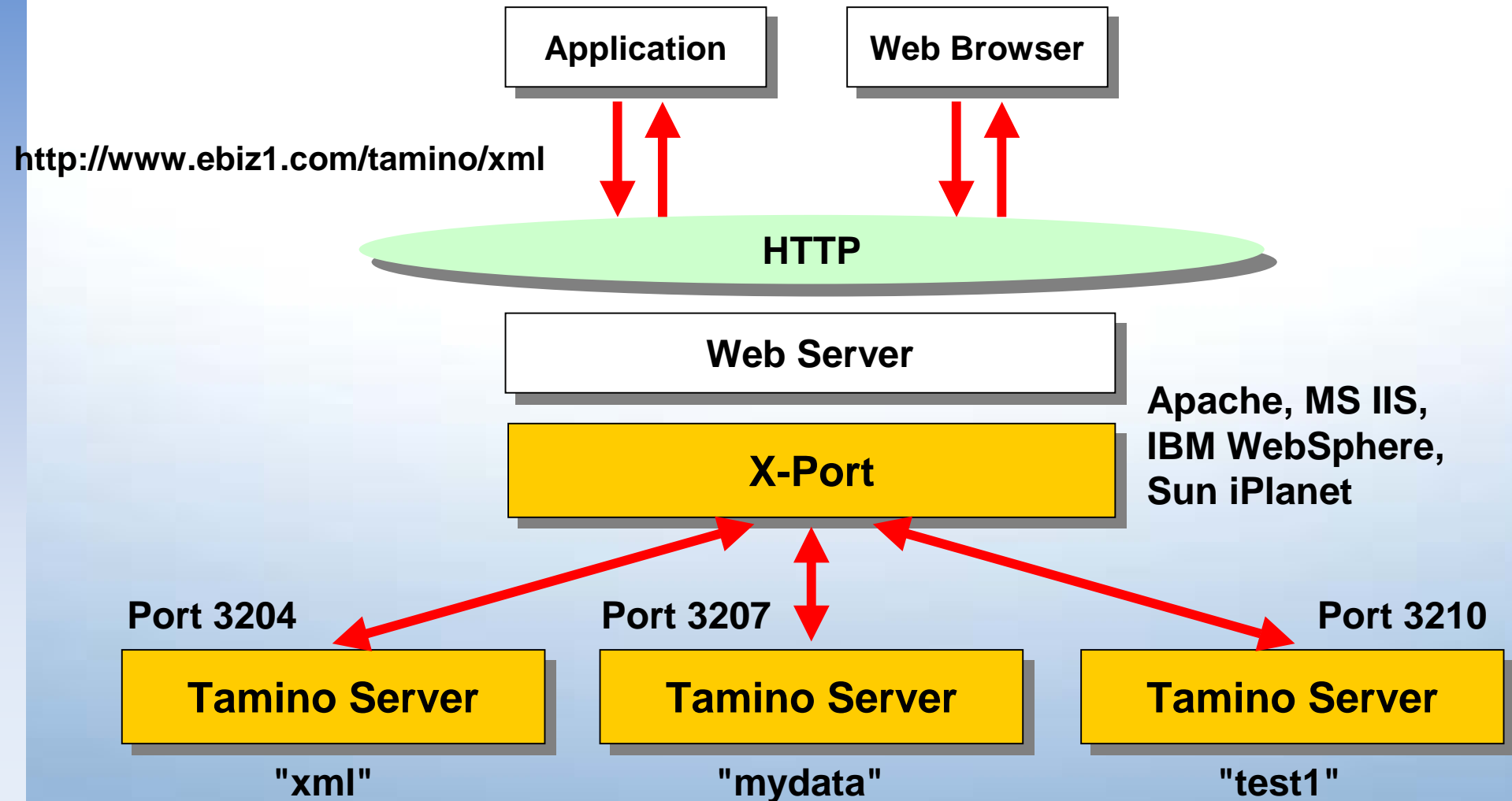
- **The world's fastest XML Database / Server**
  - Native Storage of XML-Objects w/o conversion into other formats
- **Integration of data in existing, external data sources**
  - Access to and modification of data from diverse systems (relational DBs, object DBs, Office-Systems...)
- **Database Queries with 'XQuery'**
  - XPath-based - regarding document structure and content
- **Simple administration**
  - Browser-based control via any PC having Internet access
- **Simple connection to Internet via standard Web-Server**

# **Tamino** Features & Benefits

## **XML Database**

- **Security**
  - Group / User authorization rights down to XML Element level
- **Read-Only databases supported**
- **Dynamic Style-Sheet processing supported**
  - e.g. HTML formatted output of XML documents

# The Communication Interface



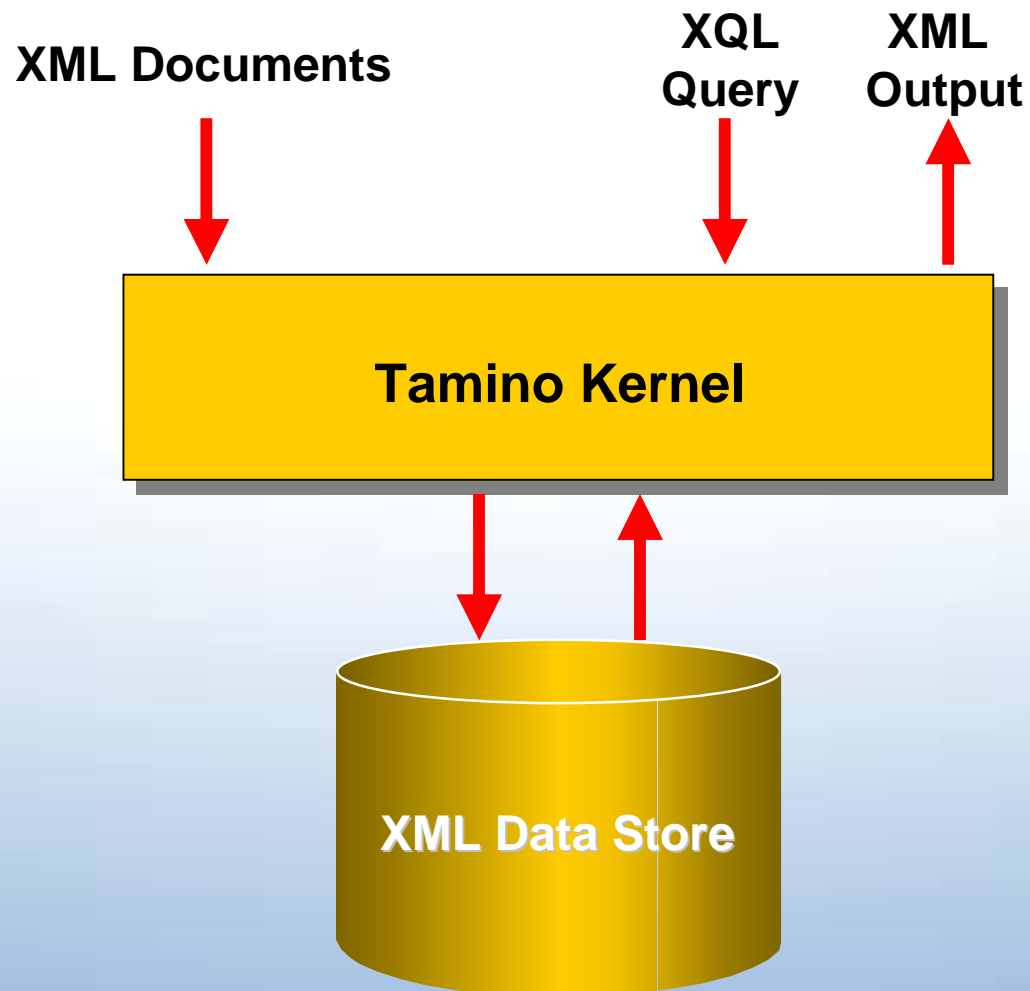
# **Tamino** Native XML Store

**XML Database**

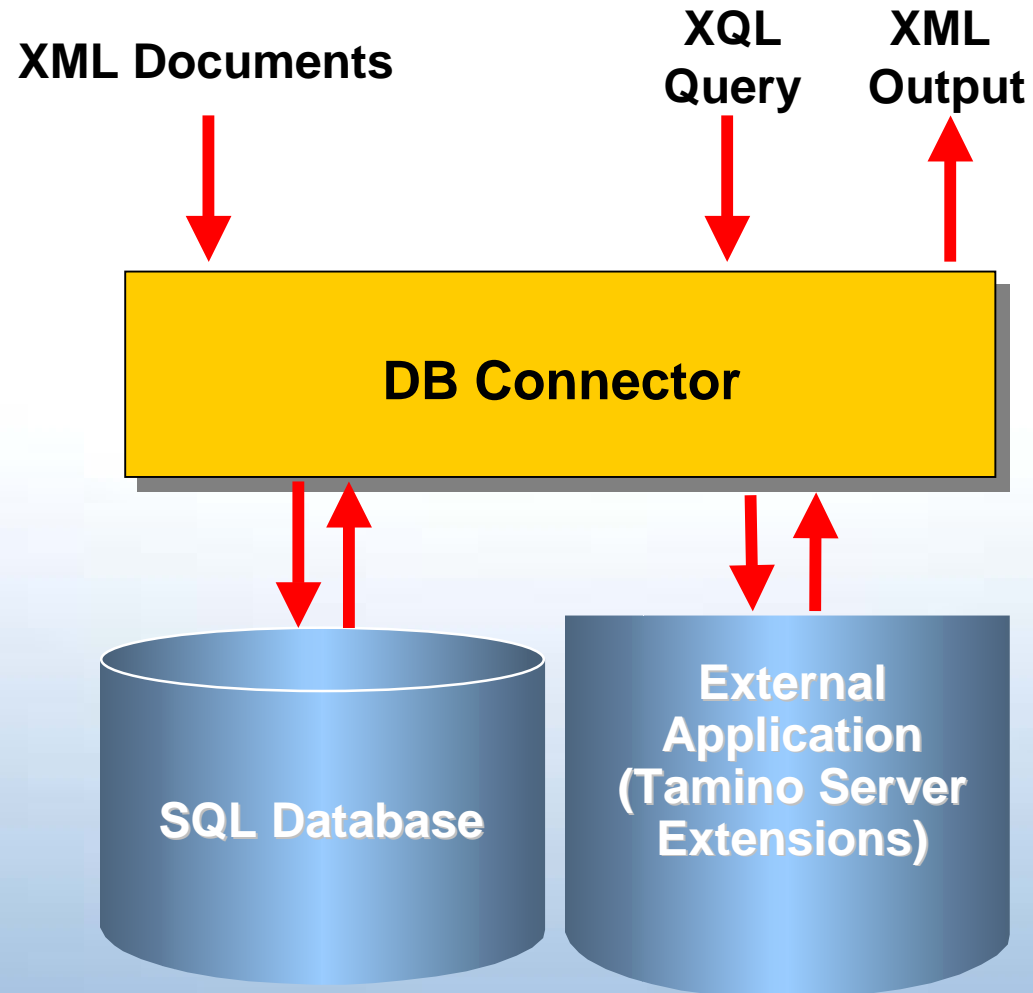
## ■ **Connecting XML-based Businesses To The Web**

- Highest performance with unlimited native XML data storage
- No conversion ! “Valid” and “well-formed” XML accepted
- XML database structure easily changeable
- Full-text search (Indexing for Full Text and Standard search)

# Native XML Server



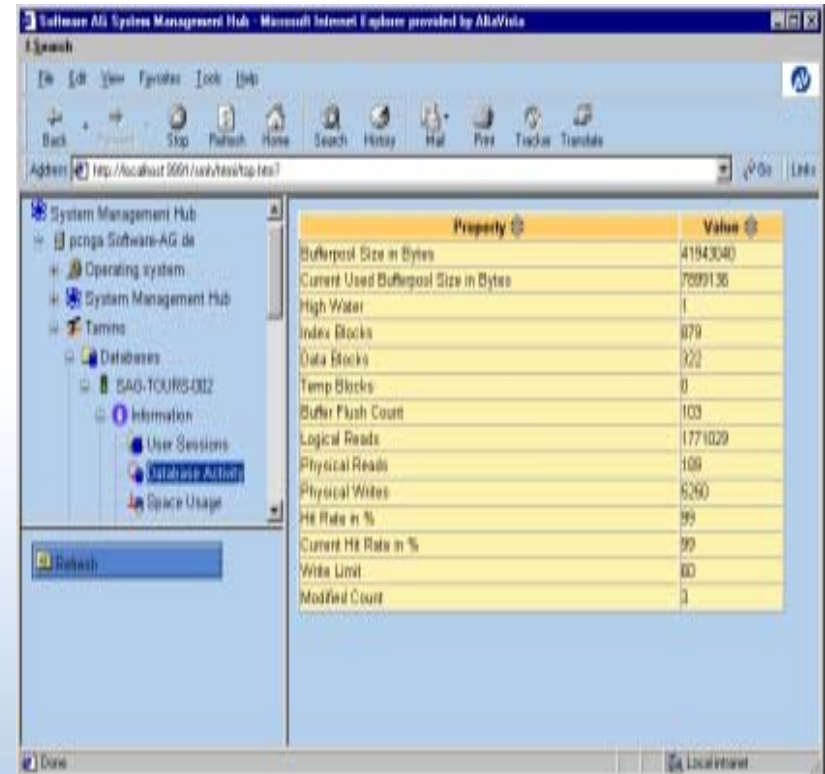
# Tamino DB Connector: data sources ==> XML



# Tamino Manager

## ■ Keeping Maintenance & Cost of Ownership Down

- Internet-based “remote control” administration
- Single point of administration for 1 or many Tamino systems
- Access from any remote location via Browser
- Create, Delete, Backup and Recover, Maintenance of servers



The screenshot shows the Software AG System Management Hub interface in a Microsoft Internet Explorer browser. The address bar shows the URL: http://localhost:8001/web/hst/top.html. The interface is divided into a left-hand navigation pane and a main content area.

The left-hand navigation pane shows a tree view of the system structure:

- System Management Hub
  - pcnaga Software-AG.de
    - Operating system
    - System Management Hub
    - Tamino
      - Databases
        - SAO-TOURS-002
          - Information
            - User Sessions
            - Database Activity** (highlighted)
            - Space Usage

The main content area displays a table of system properties:

Property	Value
Bufferpool Size in Bytes	41943040
Current Used Bufferpool Size in Bytes	7689136
High Water	1
Index Blocks	879
Data Blocks	322
Temp Blocks	0
Buffer Flush Count	103
Logical Reads	1771020
Physical Reads	168
Physical Writes	5260
Ht Rate in %	99
Current Ht Rate in %	99
Write Limit	80
Modified Count	3

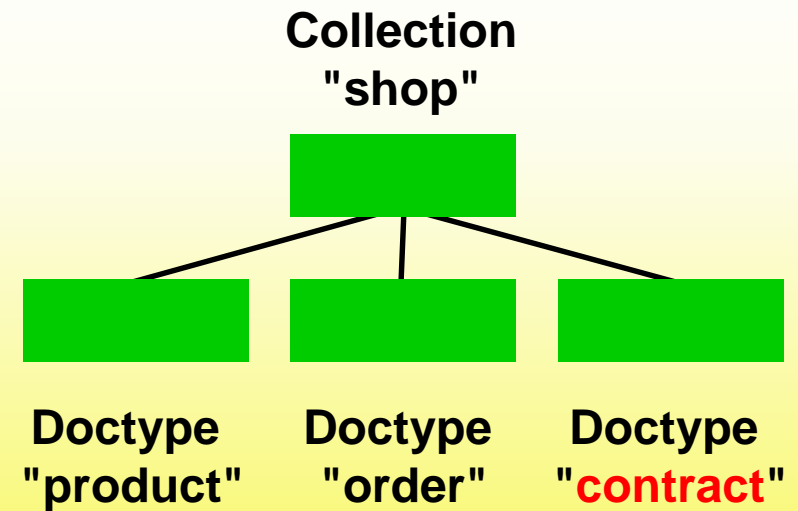
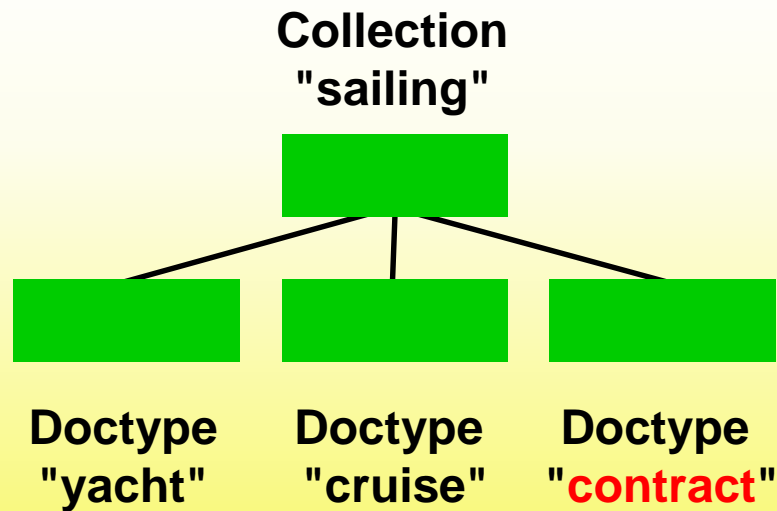
# Tamino

XML Database

## Tamino XML Schema

- Contains all the information needed for storage, indexing and processing of XML objects, especially for
  - storage of XML structures and properties
  - Integration of external databases / applications
  - Construction of standard and text indices

# Schema - Collections



# Generating a Schema

DTD/  
XSD

open

The screenshot shows the Tamino Schema Editor interface. On the left is a tree view of the 'Telephone' schema, including elements like 'Choice', 'LoginName', 'Password', 'Lastname', 'Firstname', 'Date\_of\_Birth', 'Company\_Name', 'Salutation', 'Email', 'Address', and 'EntryID'. The 'Password' element is selected. On the right, the 'Logical Properties' table is visible:

Property	Value
id	
name	Password
minOccurs	1
maxOccurs	1
mixed	true
default	
fixed	
form	unqualified
Collation	no

Below the logical properties is the 'Physical Properties' section, showing 'Storage Type' set to 'Native' and an 'Advanced' button. At the bottom, a status bar shows a task log: 'Get Tamino Schema (TBD3) <@SCED0002>', 'started at January 30, 2002 4:50:32 PM CST <@SCED0003>', and 'finished at January 30, 2002 4:50:32 PM CST <@SCED0004>'.

define

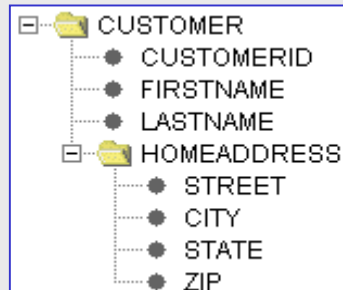
Tamino

Tamino Schema Editor

# I have the following simple XML in my customer.xml file. How to store this in Tamino?

```

<CUSTOMER>
  CUSTOMERID>1044</CUSTOMERID>
  <FIRSTNAME>Paul</FIRSTNAME>
  <LASTNAME>Astoria</LASTNAME>
  <HOMEADDRESS>
    <STREET>123 Cherry Lane</STREET>
    <CITY>Best</CITY>
    <STATE>CA</STATE>
    <ZIP>94132</ZIP>
  </HOMEADDRESS>
</CUSTOMER>
  
```



Store XML document into Schema using URL with command **\_Process** in default collection

**Ready!**  
**OR USE**

DTD

Tamino XML Schema

Read DTD or XML Schema into Tamino Schema Editor

Mark check boxes for (text) indexing

Save XML Schema with a collection name

Store XML document using URL with command **\_Process** in named collection

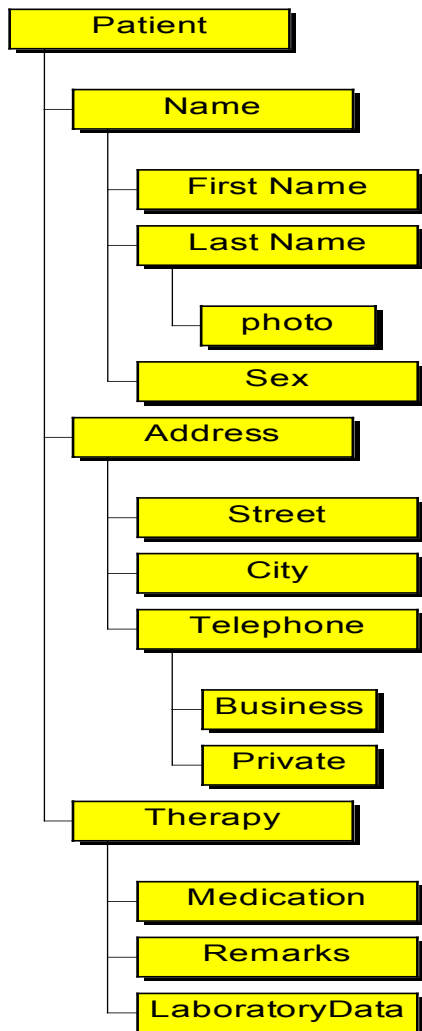
# Why is XML difficult to handle by RDBMS?

- An RDBMS is primarily designed to handle table oriented data
  - structure and format of data is known prior to storage
  - names and types of data components is fixed and unchanging
  - columns descriptions are used for indexing and retrieval
  - without column descriptions we do not know the meaning of the content

<b>Name</b> <i>A30</i>	<b>Address</b> <i>A30</i>	<b>City</b> <i>A15</i>	<b>Phone</b> <i>N10</i>
Anders	Mainstreet 2	New York	43828181
Berger	Lowlane 6	Washington	40298381

<b>Code</b> <i>A8</i>	<b>Product Description</b> <i>A50</i>
A345	Wheel 15"
C702	Steam Engine 2000 kW

# RDBMS has different storage methods with different problems



## LOB Storage

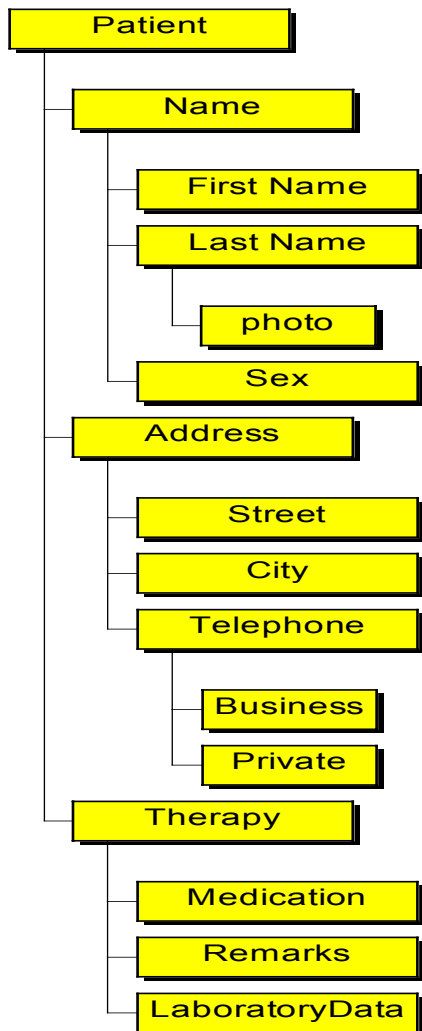
- XML documents stored in CLOB (Character Large Objects)
- Metadata of document stored in object-relational tables
- Text search in XML elements with Add-on Service (e.g. *Oracle InterMedia*)

### Problems:

**Much of the SQL functionality on object-relational columns cannot be exploited**

# RDBMS has different storage methods with different problems

see Oracle 9i  
Application Developer's Guide - XML



## DataType: XMLtype Storage (9i only)

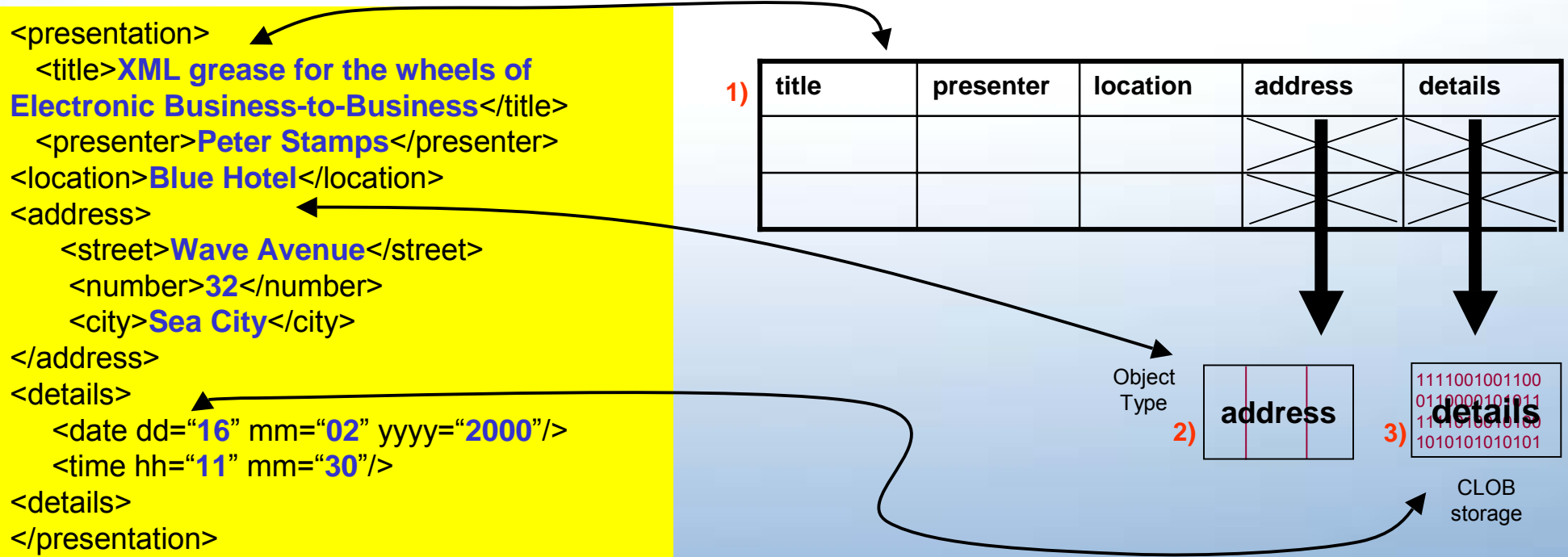
- XML document stored in a CLOB
- Extract and ExistsNode functions can be used in SQL
- No new indices or further optimization compared with CLOB storage

## Problems:

- Support **limited set of XPath** expressions,
- **No sibling or parent axes** are supported
- **No predicates** are supported (predicates filters a node-set according to the predicate expression)
- The **result** of the XPath **cannot be a boolean expression**
- Extract and ExistsNode functions **do not support multi-byte character sets**

# RDBMS's mapping rules for XML to tables

- ❑ Root can be mapped to table name
- ❑ Tag names (top elements) are mapped to columns <sup>1)</sup>
- ❑ Text-only data (elements) are mapped to scalar columns
- ❑ Sub-elements are mapped to object types <sup>2)</sup>
- ❑ Element lists are mapped to collections (table of object types)
- ❑ **Attributes in XML cannot be mapped into tables** (need a CLOB) <sup>3)</sup>

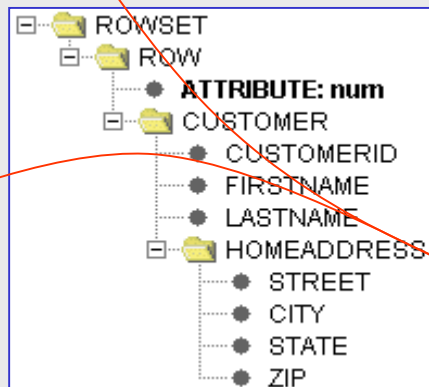


# I have the following simple XML in my customer.xml file. How to store this in Oracle?

```

<ROWSET>
  <ROW num="1">
    <CUSTOMER>
      <CUSTOMERID>1044</CUSTOMERID>
      <FIRSTNAME>Paul</FIRSTNAME>
      <LASTNAME>Astoria</LASTNAME>
      <HOMEADDRESS>
        <STREET>123 Cherry Lane</STREET>
        <CITY>Best</CITY>
        <STATE>CA</STATE>
        <ZIP>94132</ZIP>
      </HOMEADDRESS>
    </CUSTOMER>
  </ROW>
</ROWSET>

```



```

create type address_type as object (
  street varchar2(40),
  city varchar2(20),
  state varchar2(10),
  zip varchar2(10) );

```

```

create type customer_type as object (
  customerid number(10),
  firstname varchar2(20),
  lastname varchar2(20),
  homeaddress address_type );

```

```

create table customer_tab
(customer customer_type);

```

**Add <rowset> and <row> tags to document  
Insert XML document in table using utility XSU**

# How can I use full text retrieval on XML documents in Tamino?

Put some XML data into a collection:

```
_PROCESS=<XMLdocument>.....</XMLDocument>
```

```
<author>Fred Smith</author><document>I had a nice weekend in the mountains.</document>  
<author>Jack Jones</author><document>My month at the coast was relaxing.</document>  
<publisher>Dog House</publisher><document>His year in Provence was fulfilling.</document><junk>banana</junk>
```

Start querying immediately:

```
http://server-name/tamino/db-name/collection-name?_XQL=/XMLdocument[author~="Smith"]
```

# How can I use full text retrieval on XML documents in Oracle 8i? (1 of 2)

## Put some XML data into a simple table:

```

CREATE TABLE my_table (id number(5) primary key, my_column clob );
INSERT INTO my_table VALUES ( 1, '<author>Fred Smith</author>' ||
'<document>I had a nice weekend in the mountains.</document>' );
INSERT INTO my_table VALUES ( 2, '<author>Jack Jones</author>' ||
'<document>My month at the coast was relaxing.</document>' );
INSERT INTO my_table VALUES ( 3, '<publisher>Dog House</publisher>' ||
'<document>His year in Provence was fulfilling.</document>' ||
'<junk>banana</junk>' );
COMMIT;
  
```

**We want to search for 'Smith' in these simple XML documents**

id number primary key	my_column clob
1	<author>Fred Smith</author><document>I had a nice weekend in the mountains.</document>
2	<author>Jack Jones</author><document>My month at the coast was relaxing.</document>
3	<publisher>Dog House</publisher><document>His year in Provence was fulfilling.</document><junk>banana</junk>

# How can I use full text retrieval on XML documents in Oracle 8i? (2 of 2)

**FIRST create an interMedia Text Index Definition as follows:**

```

BEGIN
  Ctx_Ddl.Create_Section_Group /* We're dealing here with XML, not say HTML */
  ( group_name => 'my_section_group', group_type => 'xml_section_group' );

  Ctx_Ddl.Add_Field_Section /* THIS IS KEY */
  ( group_name =>'my_section_group', section_name =>'author', /* do this for EVERY tag used after ""
  tag =>'author' );

  Ctx_Ddl.Add_Field_Section /* THIS IS KEY */
  ( group_name =>'my_section_group', section_name =>'document', /*do this for EVERY tag after "WITHIN" */
  tag =>'document' );
END;
```

**We want to search for 'Smith' in simple XML documents**

**OR (only > Oracle 8.1.6)**

```

BEGIN
  ctx_ddl.create_section_group('my_section_group', 'auto_section_group'); /* automatic EVERY TAG after "WITHIN" */
END;
```

**THEN create the Index**

```

CREATE INDEX my_index ON my_table ( my_column ) INDEXTYPE IS Ctxsys.Context
PARAMETERS ( 'SECTION GROUP my_section_group' );
```

**THEN select Smith within Author**

```

SELECT my_column FROM my_table WHERE CONTAINS ( my_column, 'smith WITHIN author' ) > 0;
```

# How can I query XML in a column of data type XMLType (9i)?

- Oracle Text (intermedia Text) indexes a CLOB/BLOB without knowing anything about XML specifically

- Searches based on structural conditions are **not available**

For example this is not possible: select name from person where hair.color = "BROWN"

```
<person>
```

```
....
```

```
  <hair>
```

```
    <color>Brown</color>
```

```
  </hair>
```

```
</person>
```

- **Selecting a XML subtree is not possible**

You will need to subsequently parse the whole CLOB to extract it.

# How do I do a range search in a column of data type XMLType (9i)?

- Example a Curriculum Vitae of Employee stored as XML document in a CLOB has tags like  
...<SALARY>4680</SALARY>...
- How to select all the table records that have salary  $\geq 159980$  and  $\leq 165000$ ?  
I need the value between the tags as a floating point number for my calculation, as this is a salary.
- **In Oracle Text this is not possible.**  
You need to extract the content of the tag <SALARY> into an NUMBER column and use SQL to do the ranging

In Tamino: `_xql=Employee[//Salary between 159980, 165000]`

# How do we store multi-lingual XML documents in the same table-column?

- Separate out all the fields that could potentially be in different languages into different columns in the same table
- Create a corresponding language column for each of those columns
- Use the multi-plexer functionality to build separate indexes
- Use multiple CONTAINS clauses when searching across columns which can adversely affect performance

In Tamino use a language attribute in the XML document and query

Document: `<document language="CH FR"> ...</document>`

Query : `?_xql=/document[@language="CH FR"]`



# Compare insert of data: O9i and Tamino

**ORACLE 9i**

Using SQL tool or Programming language

```

CREATE TABLE warehouses (
  warehouse_id      NUMBER(3),
  warehouse_spec    SYS.XMLTYPE,
  warehouse_name    VARCHAR2(35),
  location_id       NUMBER(4) );

INSERT INTO warehouses (
  warehouse_id, warehouse_spec) VALUES (1001, sys.XMLType.createXML
  ('<Warehouse whsid="1001"><Building>Owned</Building></Warehouse>') );
  
```

**Tamino**

Using Browser or Programming language

```

http://hostname/tamino/dbname/collection/doctype?_process=
  <Warehouse whsid="1001"><Building>Owned</Building></Warehouse>
  
```

# Compare query of data and response

looking-up string value of node Warehouse/Building

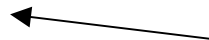
**ORACLE 9i**

```
SELECT w.warehouse_spec.extract('/Warehouse/Building/text()').getStringVal()
       "Building" FROM warehouses w
```

Building

-----

Owned

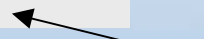


Response is always a row  
(with column heading)

**Tamino**

```
http://hostname/tamino/db/collection/doctype?_xql=/Warehouse/Building/text()
```

```
<xql:result>Owned</xql:result>
```



Response is always XML

`text()` selects all text node children of the context node

# Compare query of data and response

looking for the last section in the 5<sup>th</sup> chapter

ORACLE *9i*

```
SELECT w.doc_spec.extract(' /doc/chapter[5]/section[last()]).getStringVal() "Section" FROM
doc w
```

Error as no predicates are allowed



*Tamino*

```
http://hostname/tamino/dbname/collection/doctype?_xql=/doc/chapter[5]/section[last()]
```

```
<xql:result>
```

```
<section ino:id="6">
```

A predicate filters a node-set with respect to an axis to produce a new node-set. For each node in the node-set to be filtered, the PredicateExpr is evaluated with that node as the context node, with the number of nodes in the node-set as the context size, and with the proximity position of the node in the node-set with respect to the axis as the context position; if PredicateExpr evaluates to true for that node, the node is included in the new node-set; otherwise, it is not included.

```
</section>
```

```
</xql:result>
```

Extract from XPATH specification

`/doc/chapter[5]/section[last()]` selects the last section of the fifth chapter of the doc

# No support of predicates means no direct support for this type of questions

- I am looking for the last paragraph of a news paper article!
- What was the last job description of this person?
- What is the current price (last) of this product history?
- Select all data of a patient except his or her name and any remarks at the end of the XML document
- Predicates are useful for any query where a "location or positional" aspect is involved

## From Oracle Developers Guide

### Predicates are Not Supported

**XMLType does not support any predicates. If you need predicate support, you can rewrite the function into multiple functions with the predicates expressed in SQL, when possible.**

# Combining both Worlds

## ■ RDBMS

- SQL Server
- For storage and retrieval of Transaction and Table based data
- Topics:
  - Transactional data for internal usage
  - Risk of performance decrease when opened to Internet, conversion XML↔SQL needed
  - Data needs to be "reworked" for external usage
  - XML support is complex and cumbersome
    - relative long development cycle
    - high costs
    - time-to-market ↓

## ■ Tamino

- XML Server
- For storage and exchange of XML based Business Documents
- Topics:
  - Medium neutral data storage for in - and external multi-channel publishing for internet enabled devices
  - High performance for Internet e.g. Caching services
  - Data prepared for outside world commercial, technical, economical,...
  - XML support is key, no conversion
  - Time-to-market is key ↑
    - less effort needed

# Summary of Functional Benchmark

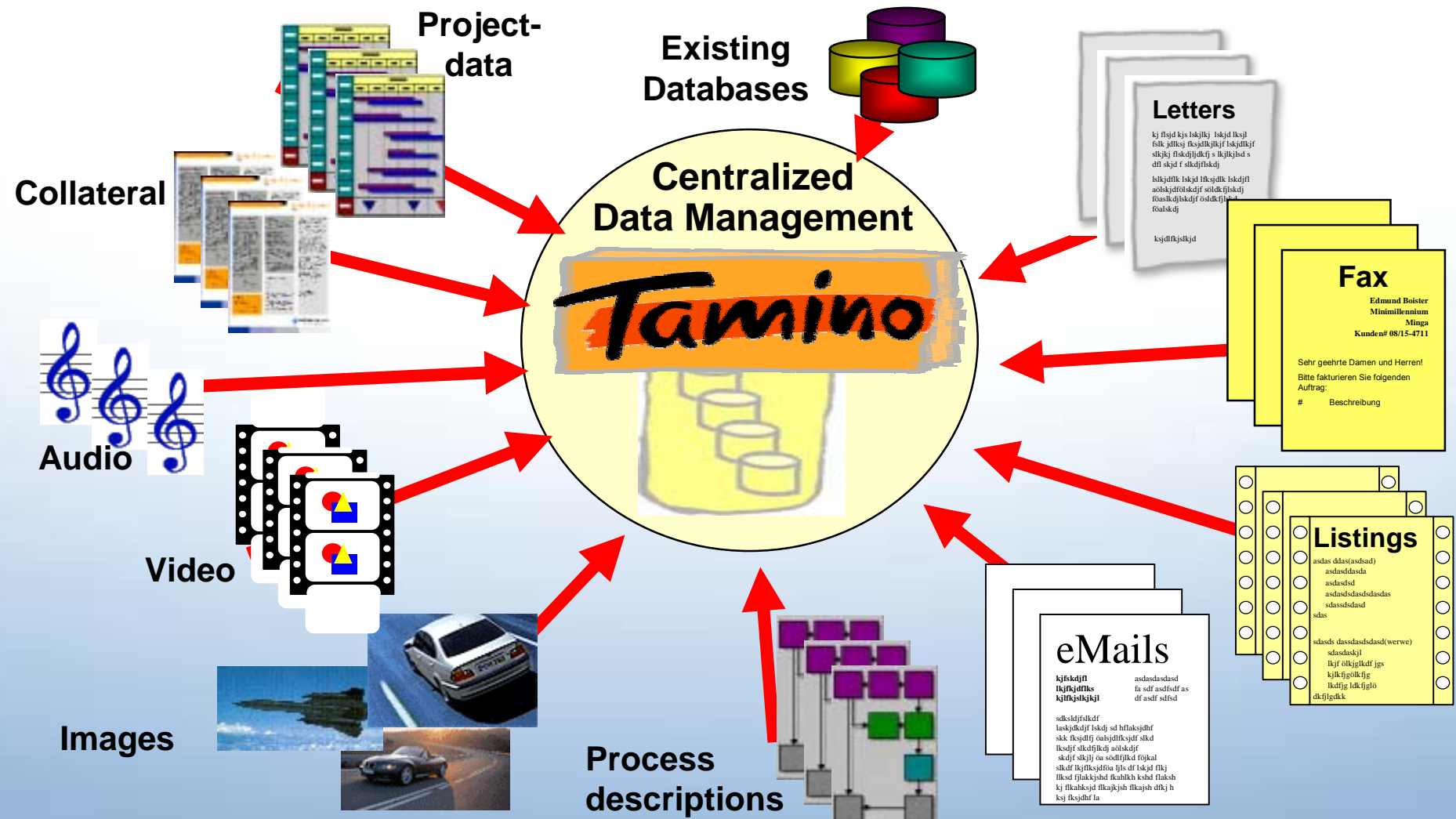
- **Benchmark with leading RDBMS systems shows that RDBMSs**
  - ... have restrictions in XML data modelling
  - ... offer limited XML query functionality
  - ... require time-consuming and costly administration
- **Only a limited subset of XML queries are comparable**
  - ... as RDBMSs simply cannot perform them
- **When it comes to XML, Tamino has major advantages**
  - ... in handling and administration
  - ... shows up a better performance in most cases
  - ... because it is the only system for direct XML storage and full/partial retrieval, with maximum XML functionality

# Summary of Performance Benchmark

## ■ The Message:

- Tamino Query in comparison with leading RDBMS
  - Overall Performance about ~ 4 times faster
- Data-centric approach
  - Comparable Queries factor 2 – 4 times faster
  - In a specific case factor 21 (e.g. missing Index)
- Document-centric approach (only Oracle)
  - Limited comparable Queries factor 1,5 – 2,5 times faster
  - Most queries 2-5 times faster
  - In specific case factor 50

# Electronic Business and Tamino: Easier management of any data



# XQuery

## ■ Examples :

### Filter Expression

```
/cruise [ @cruiseid = 5152 ] /skipper
```

```
/cruise [ skipper//address/city = 'DERBY' ] /skipper
```

### Relational Operators ( = != > >= < <= ~= )

```
/cruise[ @status != 'available']
```

```
/cruise/route[ harbor = "Athen" ]
```

```
/cruise[route/ harbor ~= "at*"]
```

### Arithmetic Operators ( + - \* div mod )

```
/cruise[ (price -200) div 5 < (2000 * 1.16) mod 1000 ]/price
```

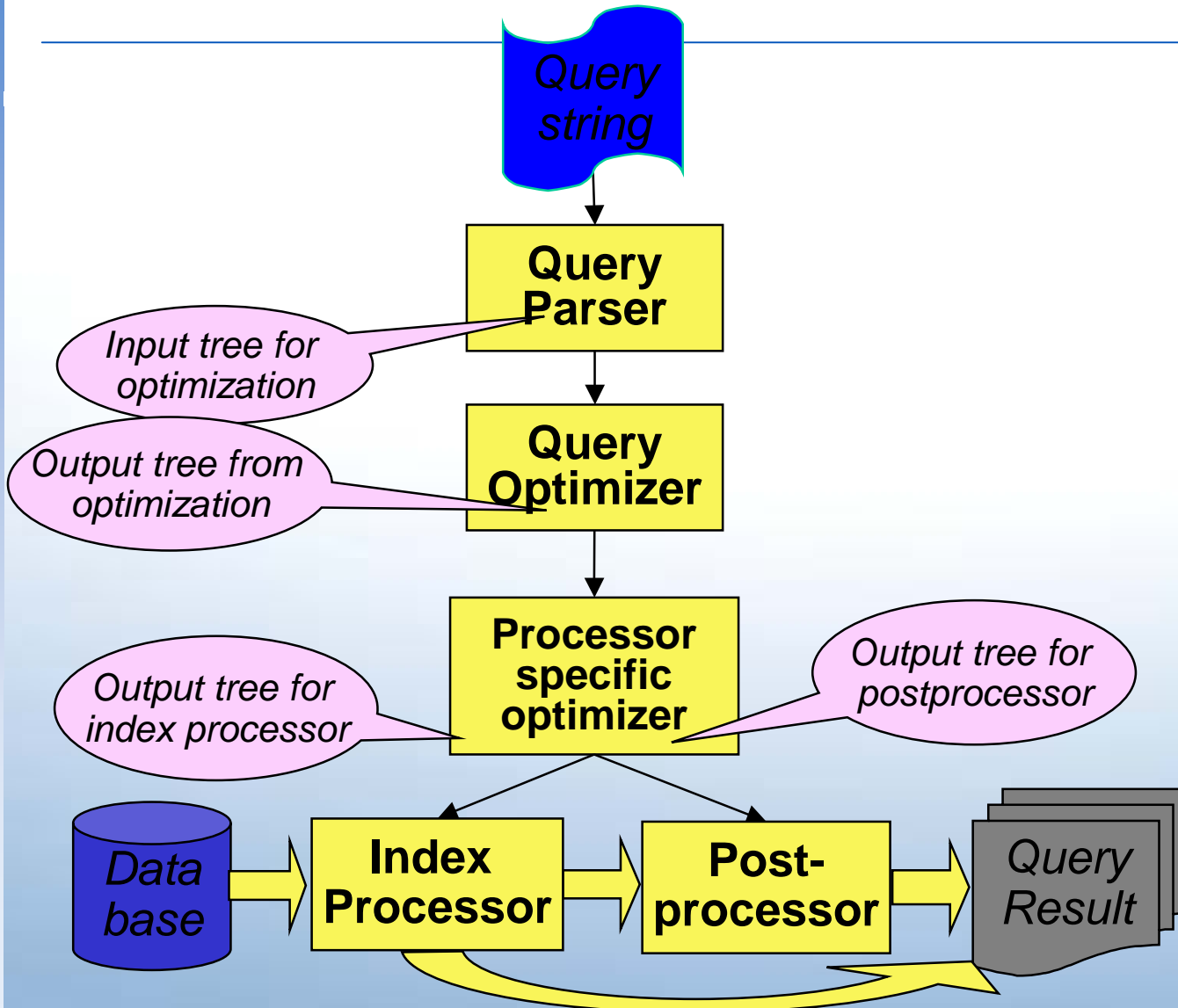
### Logical Operators (and or)

```
/cruise[price < 1000 and skipper/name ~= "abellan"]
```

# XQuery Syntax

- Relational Operators      = != > >= < <= ~=
- Filter Expression      []
- Attribute Operator      @
- Step/Path/Axes/Position operators    / // . .. []
- Sorting                    sortby(asc/desc)
- Arithmetic Operators    + - \* div mod
- Logical Operators      and or
- Between Operator      between betw
- Proximity Operators    near adj
- Sequence Operators    before after
- .....

# Query Optimizer



## Components:

- index processor
- postprocessor
- query parser
- query optimizer
- processor specific optimizer

# ino:explain(*query*, "path")

- shows elements and attributes used by *query* together with matype and searchtype as retrieved from repository
- result is a set of nested xop:path elements

**Sample query: /patient[.//surname~="Atkins"]/address**

```
<xql:result>
  <ino:explanation ino:preselection="TRUE" ino:postprocessing="FALSE">
    xmlns:xop="http://namespaces.softwareag.com/tamino/xop">
      <xop:path xop:name="patient" xop:matype="native">
        <xop:path xop:name="name" xop:matype="no">
          <xop:path xop:name="surname" xop:searchtype="text"
            xop:matype="no" />
        </xop:path>
        <xop:path xop:name="nextofkin" xop:matype="no">
          <xop:path xop:name="name" xop:matype="no">
            <xop:path xop:name="surname" xop:searchtype="text"
              xop:matype="no" />
          </xop:path>
        </xop:path>
        <xop:path xop:name="submitted" xop:matype="no">
          <xop:path xop:name="doctor" xop:matype="no">
            <xop:path xop:name="name" xop:matype="no">
              <xop:path xop:name="surname" xop:searchtype="text"
                xop:matype="no" />
            </xop:path>
          </xop:path>
        </xop:path>
      </xop:path>
    </ino:explanation>
  </xql:result>
```

...

# Powerful Free Text Search

## ■ Standard / Text Search

### ■ Standard search

Search for 'Bank of China'

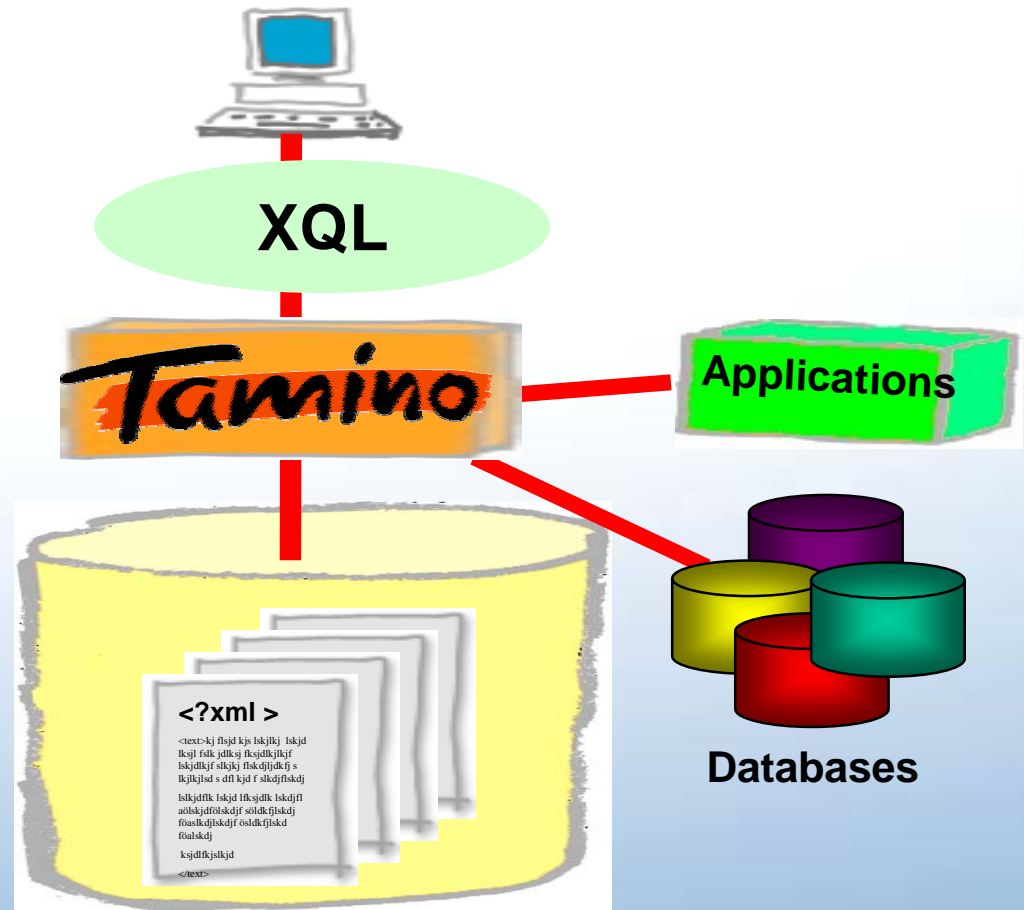
**Bank of China**  
Standard Chartered Bank  
Hong Kong & Shanghai Bank Co.

### ■ Free-text retrieval

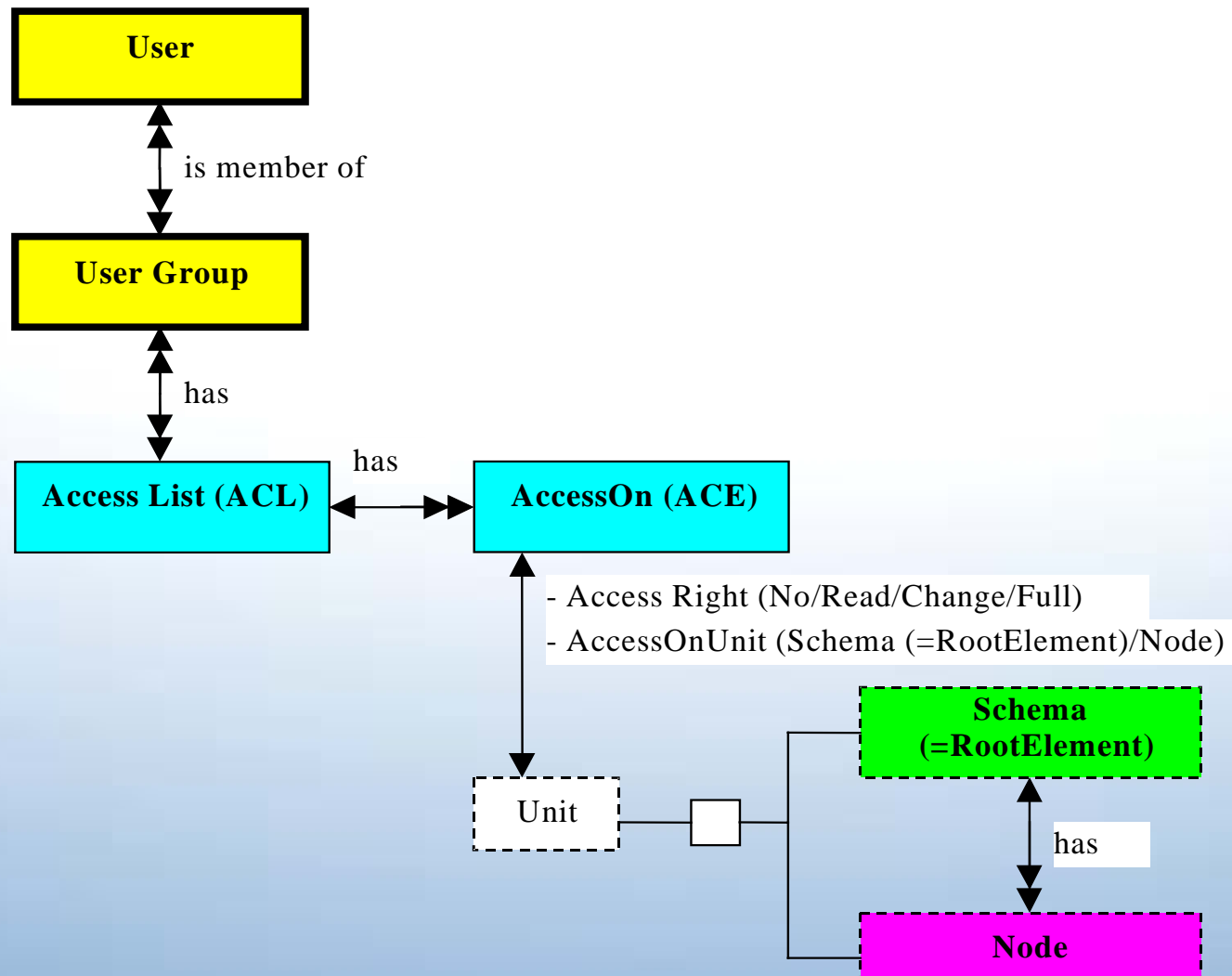
Search for '\*bank\*'

**Bank of China**  
Standard Chartered Bank  
Hong Kong & Shanghai Bank Co

### ■ Adjacent / Near search

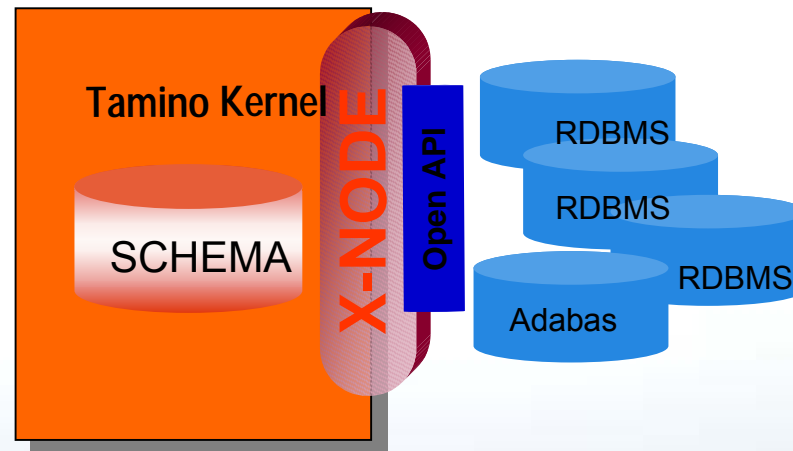


# Tamino Security



## DB Connector

### ■ Easy Integration with Existing DBMS Systems



- **“One Server View” on integrated heterogeneous databases**
  - non-native objects stored in other database types (SQL, Adabas,...)
- **Connection to existing data storage in other DB types**  
(e.g. RDBMS, Excel ... )

# XML View of SQL Table

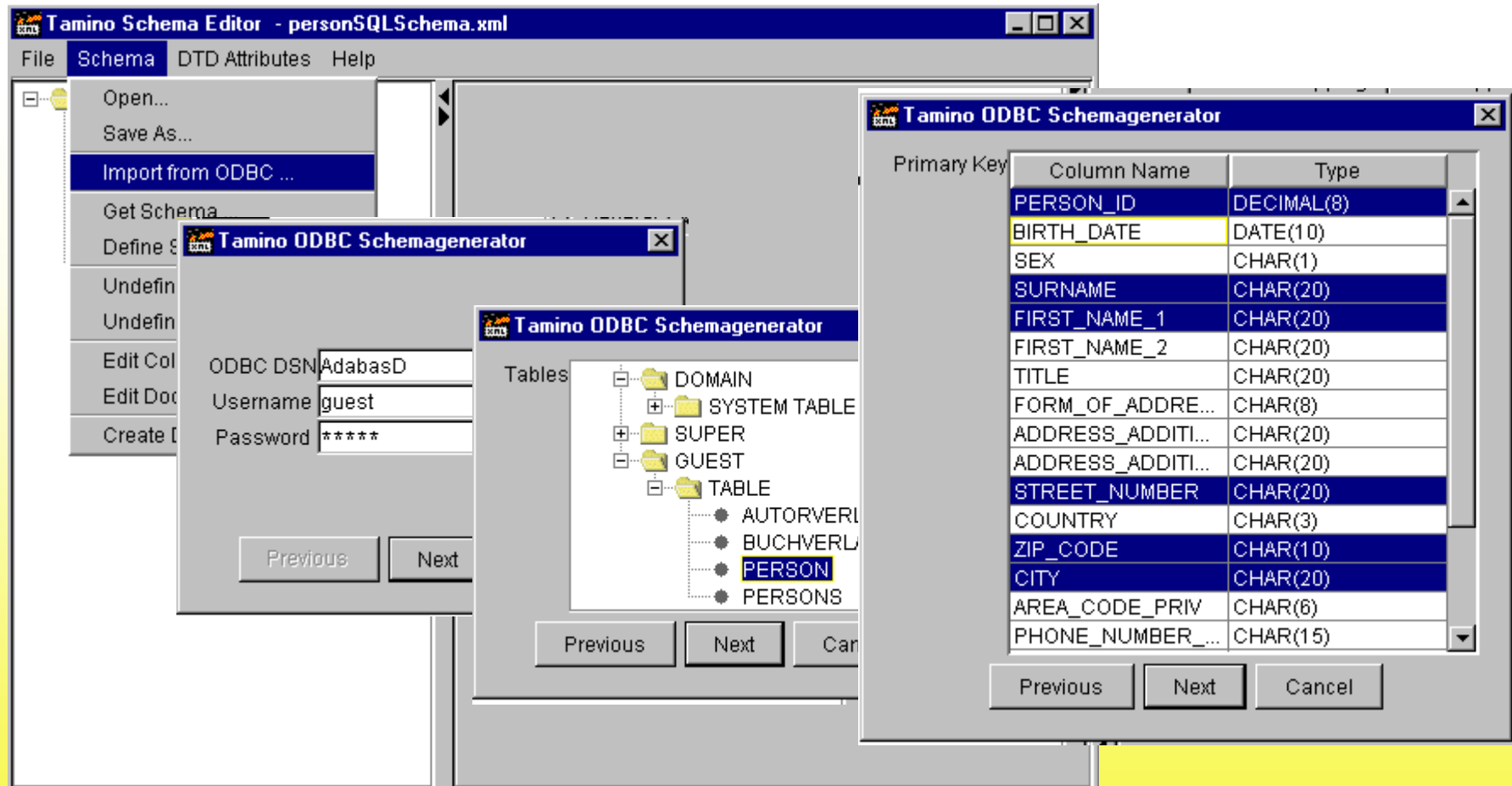
- owner
  - owner\_id
  - title
  - first\_name
  - surname
  - street\_nr
  - zip
  - city

Map-Type = **SqlTable**  
 Data Source = **Oracle**  
 Sql Table = **PERSON**  
 Sql PrimaryKeys = **&quot;PERSON\_ID&quot;**

Map-Type = **SqlColumn**  
 Data Type = **Char**  
 Sql Column=**Street\_Number**  
 Precision = **20**

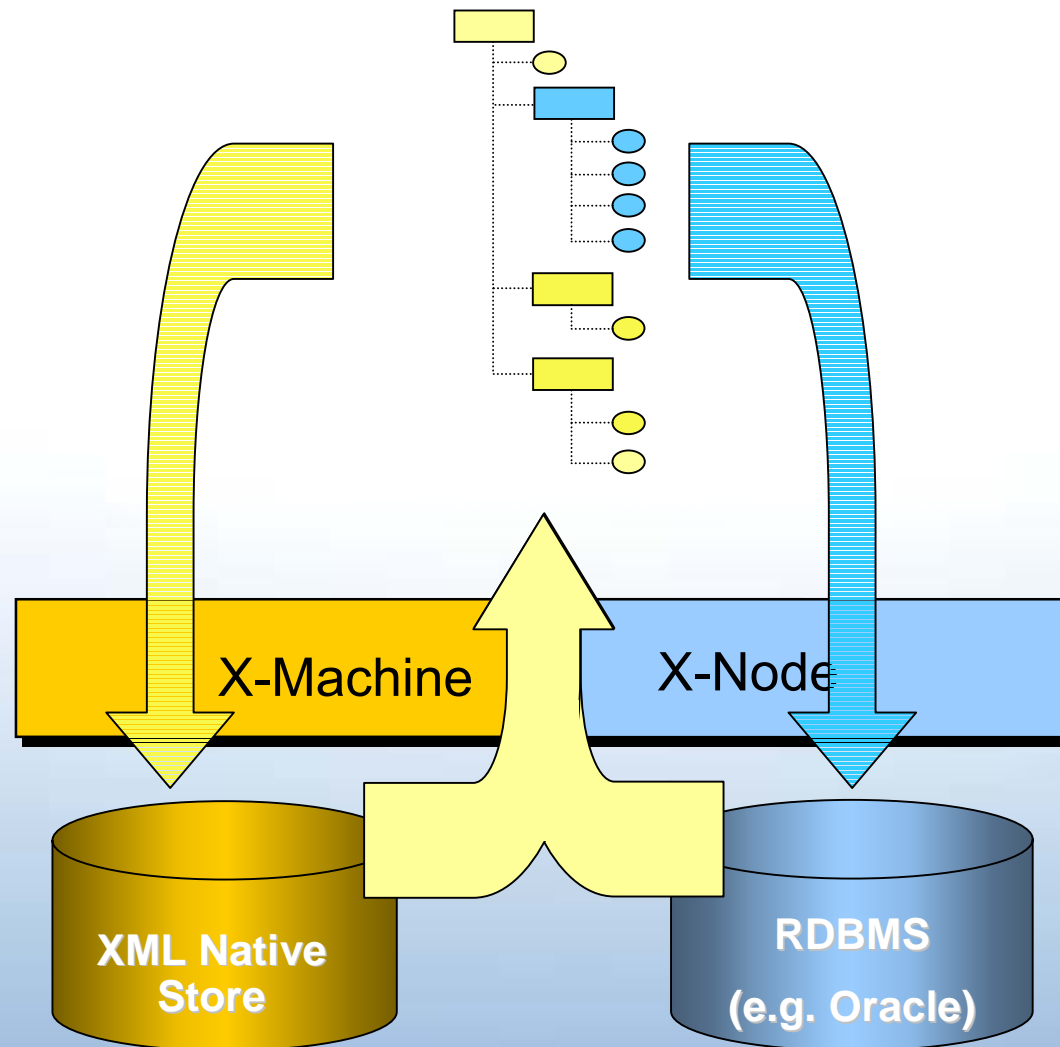
show tabledef person			
	COLUMNNAME	MOD	DATATYPE
1	PERSON_ID	KEY	FIXED
2	BIRTH_DATE	OPT	DATE
3	SEX	OPT	CHAR
4	SURNAME	OPT	CHAR
5	FIRST_NAME_1	OPT	CHAR
6	FIRST_NAME_2	OPT	CHAR
7	TITLE	OPT	CHAR
8	FORM_OF_ADDRESS	OPT	CHAR
9	ADDRESS_ADDITION_1	OPT	CHAR
10	ADDRESS_ADDITION_2	OPT	CHAR
11	STREET_NUMBER	OPT	CHAR
12	COUNTRY	OPT	CHAR
13	ZIP_CODE	OPT	CHAR
14	CITY	OPT	CHAR
15	AREA_CODE_PRIV	OPT	CHAR

# Import from ODBC



Tamino Schema Editor

# Writing and Reading Everything via Tamino



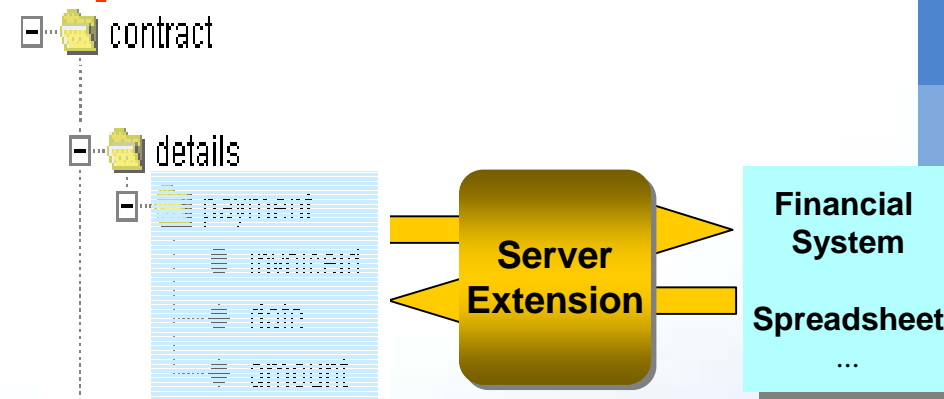
# Tamino

## X-Tension

XML Database

### Flexibility To Fulfill Special Requirements

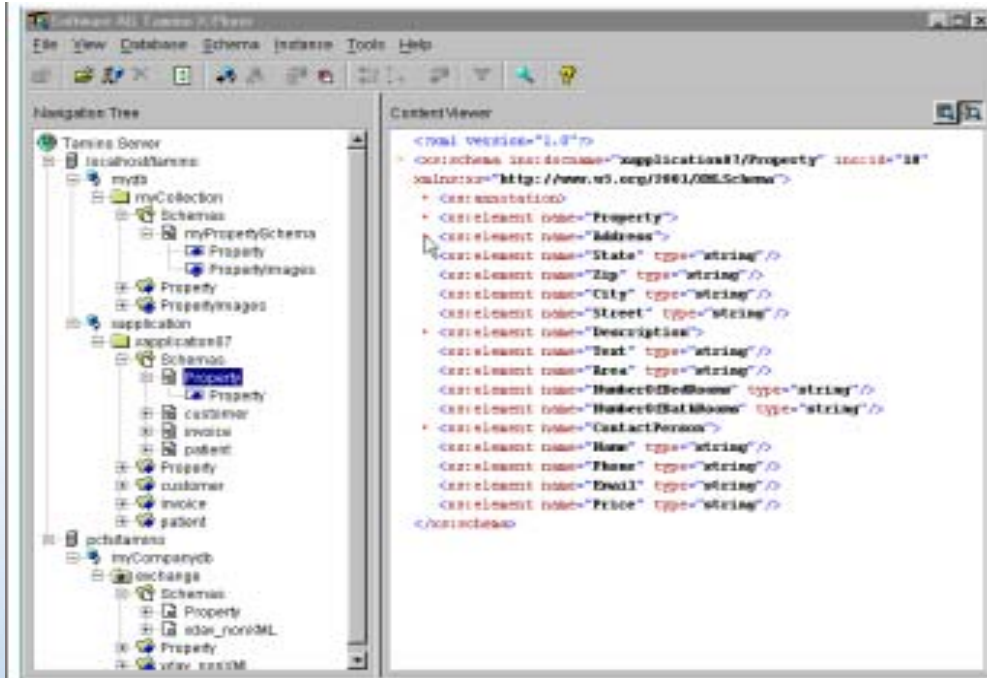
- XML-based application
- logic on the Tamino server
  - Event functions, query functions, content dependent mapping ...
  - Message forwarding on events
  - User programmable server functionality, customization
  - Allows integration with ext. Applications
  - Technology: COM Objects (C++) & Java



# Tamino Tools & Services



# Tamino X-Plorer

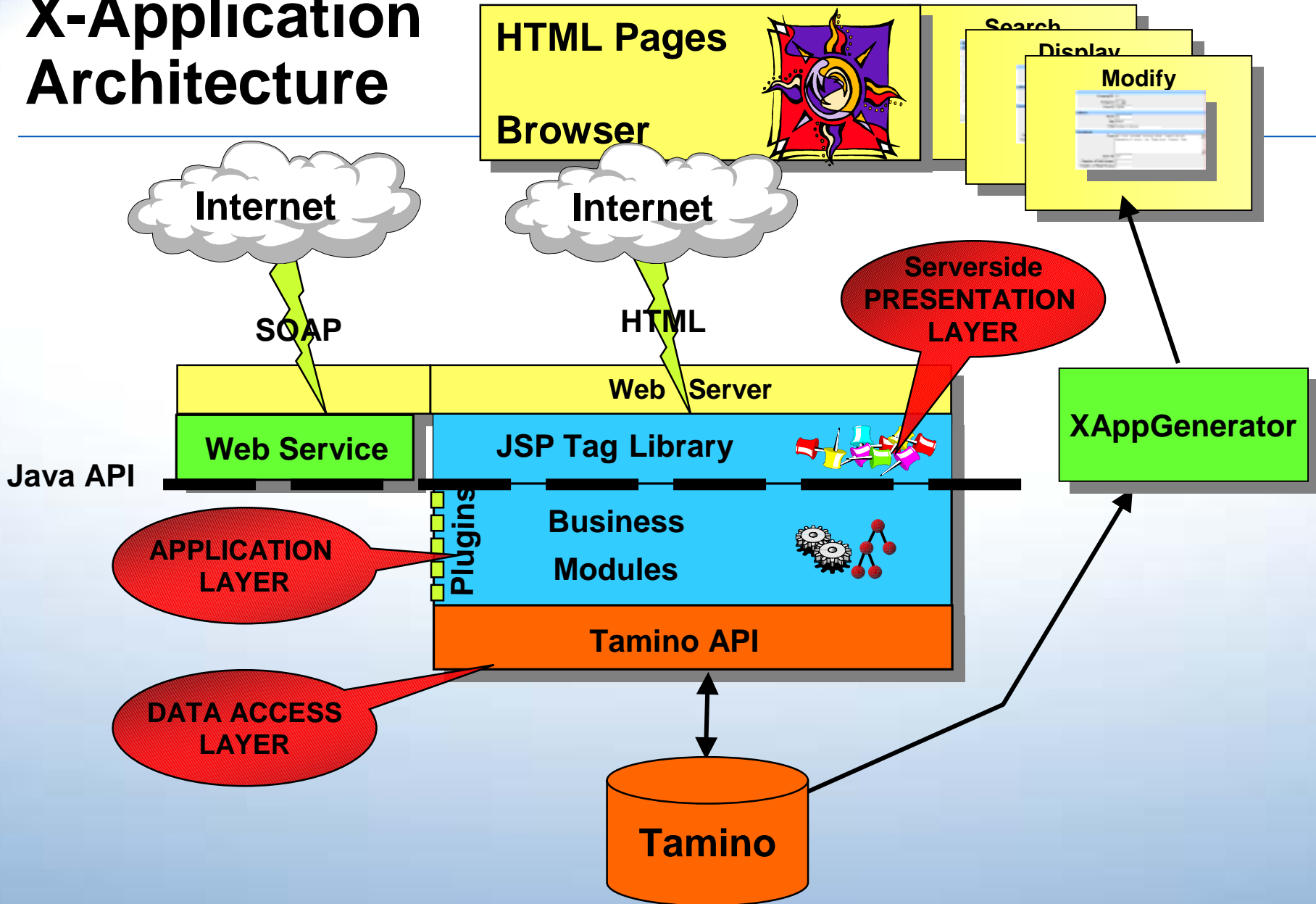


## ■ Central tool for application developer

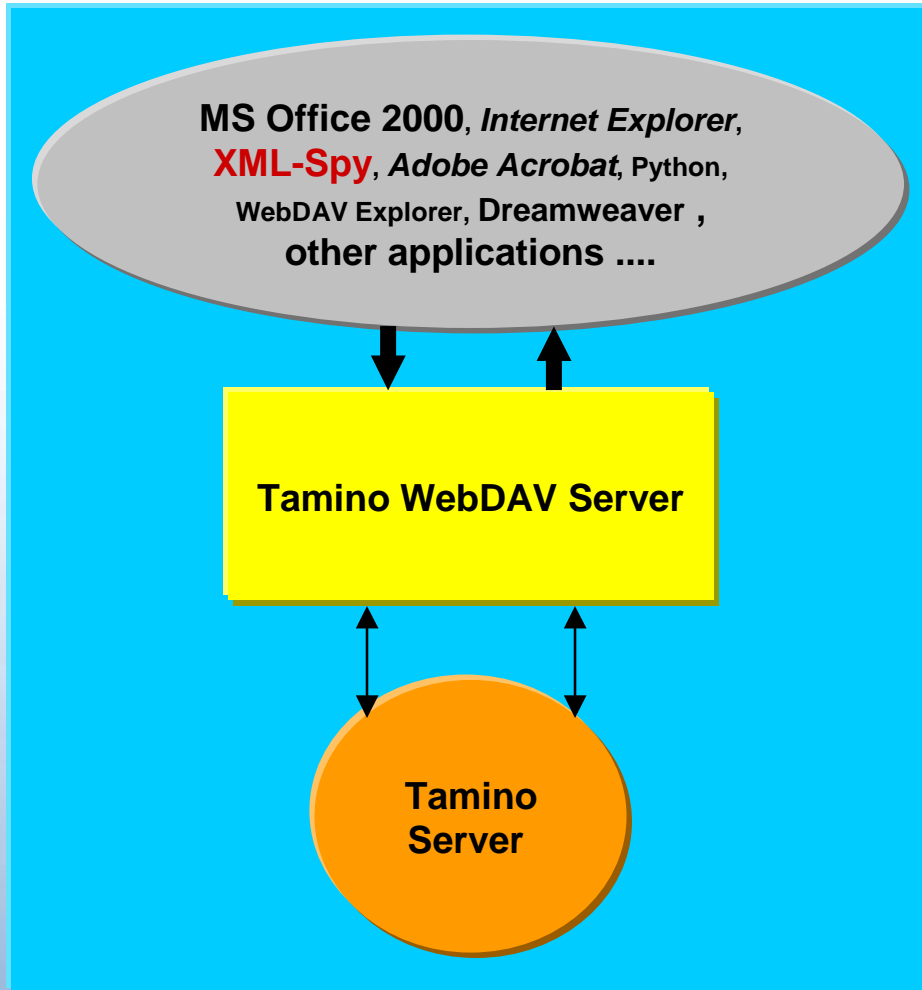
- Navigate/browse/query
- View content
- Invoke tools



# X-Application Architecture

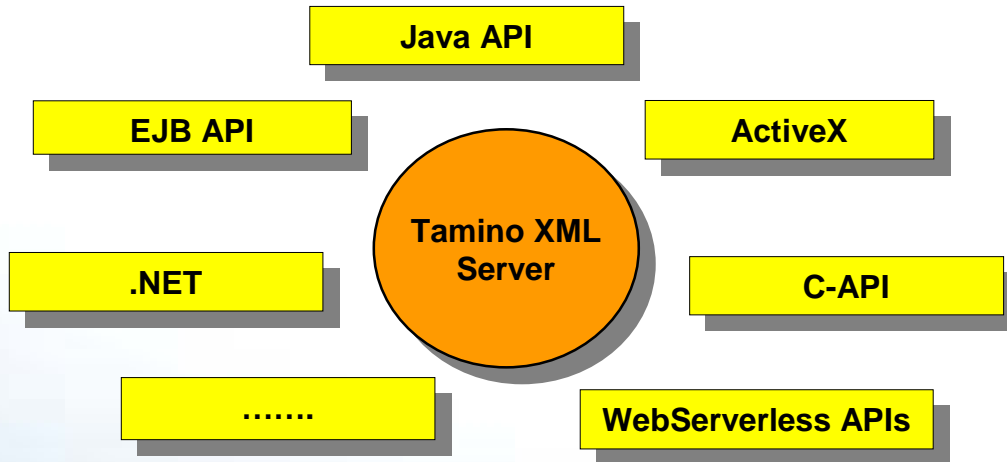


# Tamino WebDAV Server



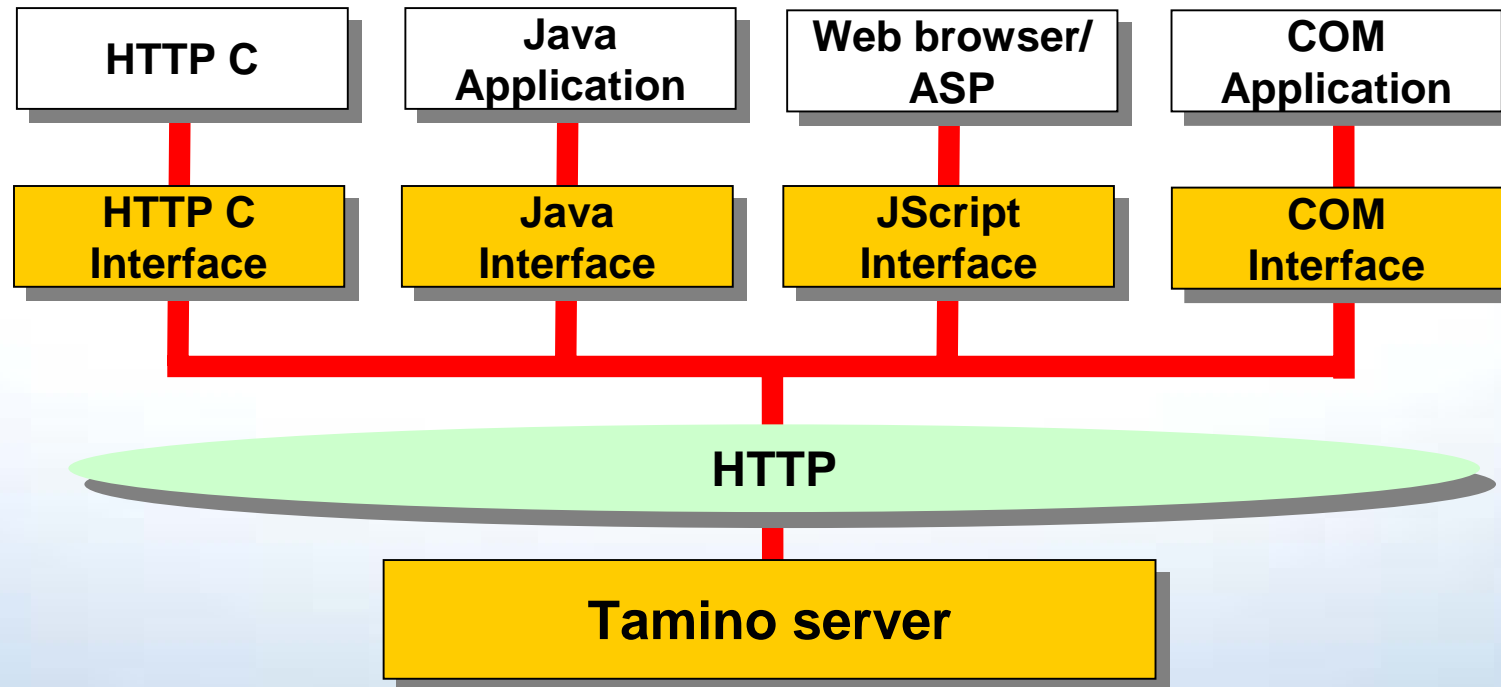
- **WebDAV** = Web-based Distributed Authoring and Versioning
- **WebDAV is a standard**
- **Future: framework for CMS (with check-in/-out, versioning, query)**
- **Community downloads**

# Tamino APIs



- **Various APIs to Tamino Server**
- **Released with Tamino 3.1**
  - Java APIs
  - EJB-API (Application Server support – BEA, IBM, HP ...)
  - ActiveX
- **Future Releases**
  - Community downloads

# Tamino DOM APIs



# Tamino Developer Community



The screenshot shows the Tamino Developer Community website. At the top, there is a navigation bar with links for Home, Contact, Search, and Legal. Below this is a header for the community. The main content area is divided into several sections:

- Releases:** A list of recent releases, including 'Tamino WebDAV Server for Solaris' (12. Dec. 01), 'Tamino X-Application 3.1.1' (04. Dec. 01), and 'Quip version 1.6.1' (04. Dec. 01).
- News:** A section with news items, such as 'Special service for our customers' (12. Dec. 01) and 'Tamino XML Server 3.1 released' (04. Dec. 01).
- Discussion Forum Statistic:** A table showing statistics for the last 7 days:
 

Discussion Forum Statistic	
Statistics for the last 7 days	
New Posts	84
Total Members	1128
New Members	18
Page Views	8022
Visits	1548
- Links:** A section with various links, including 'Software AG', 'The XML Academy', and 'XML: the XML industry portal'.

At the bottom of the page, there is a logo for 'THE TAMINO' and a copyright notice: 'Copyright 2001, Software AG. Copyright reserved.' The 'THE XML STARTER KIT' logo is also visible on the left side.

- Discussion forums and download areas
- External & internal community
- Community traffic increased
- High interest in downloads (e.g. WebDAV and Quip)
- New processes for handling “community source“
- Infrastructure for release of product components

# Usage of Tamino

- **E-Journal Library System – HKUST**
- **CRM – Wong’s Int. Company**
- **Portal – Swiss YellowWorld Portal**
- **Archival System – MemIQ AG**
- **E-Government – California Board of Equalization**
- **E-Learning – TCL, Learning Digital Ltd.**
- **.....**



**The XML Company**

**Thank you!**