Universität Zürich[UZH]

XML and Databases

Spring Semester 2018

Date: Thursday, 08:15-9:45, BIN-2.A.10

Lecturer: Dr. Can Türker (Functional Genomics Center Zurich)

Email: tuerker@fgcz.ethz.ch

WWW: http://www.ifi.uzh.ch/dbtg/teaching/courses/xmlanddatabases.html

Google Indicator (February 21, 2018)

Database	866'000'000
XML	460'000'000
FIFA	322'000'000
SQL	211'000'000
UEFA	144'000'000
Zürich	89’200'000

W3C	53'000'000
ETH	70’700'000
UBS	34’200'000
DB2	18’200'000
FGCZ	45’900
Can Türker	24’200

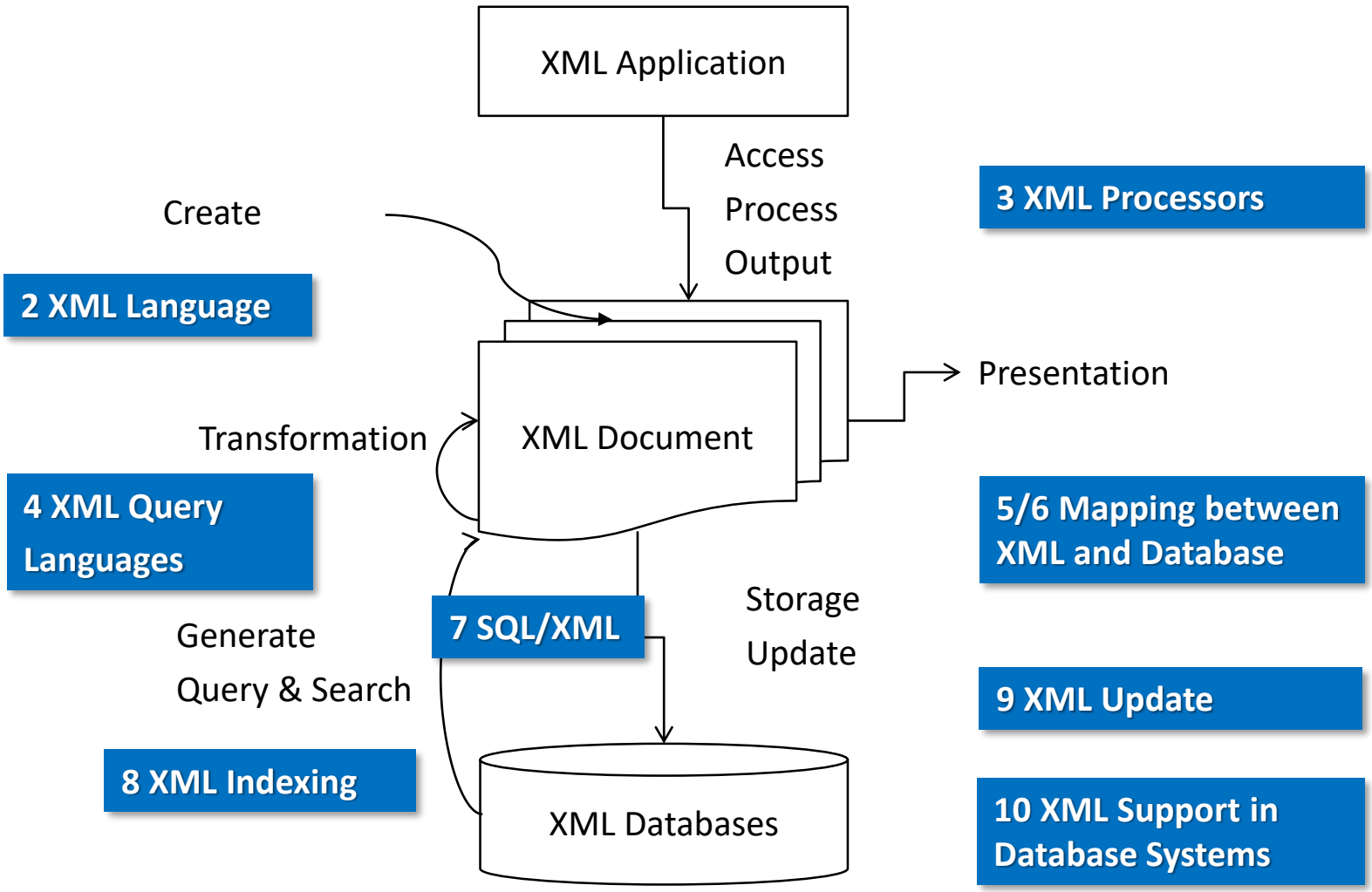
⇒ **XML and Databases: Both terms seam to be highly relevant!**

In this lecture you will learn more about XML and its interplay with databases!

Lecture Summary

- **Description:** Today, the W3C standard XML is widely used as document format for exchanging data over the Internet. While the generation of XML data is easy, the management of XML data requires systems that are able to efficiently store, query, and process XML data. With other words, database technology is required for handling XML data. The goal of this lecture is to teach the interplay between XML and databases. The following aspects are studied in detail: semi-structured data model of XML, query languages (XPath, XQuery) for declarative access to XML data, XML processor technologies, mapping between XML and databases including efficient storage and index structures for XML data. A further central concern of this lecture is to show the practical relevance of all presented concepts by demonstrating how they are realized in major database management systems such as Oracle, IBM DB2, Microsoft SQL Server, and PostgreSQL.
- **Goal:** Achieve deep understanding of XML and its interplay with database technology
- **Prerequisite:** Databases (Bachelor level), i.e., basic knowledge in databases
- **Recommended:** MSc Studies
- **Documents:** Before each lesson, presentations and demo scripts are provided for download via the lecture's home page http://www.ifi.uzh.ch/dbtg/teaching/courses/xmlanddatabases.html

Topics


XML Application
Access
Process
Output
3 XML Processors
Create
2 XML Language
Presentation
Transformation
XML Document
4 XML Query Languages
5/6 Mapping between XML and Database
Storage
Update
7 SQL/XML
Generate
Query & Search
9 XML Update
8 XML Indexing
XML Databases
10 XML Support in Database Systems

Schedule

Week	Date	Chapter
8	22.02.2018	Introduction and Motivation
9	01.03.2018	XML & XML-DTD
10	02.03.2018	XML Schema & XML Processors
11	15.03.2018	XML Processors
12	22.03.2018	XML Query Languages
13	29.03.2018	XML Query Languages
14	05.04.2018	No lecture!
15	12.04.2018	Mapping between XML and Databases
16	19.04.2018	Mapping between XML and Databases
17	26.04.2018	SQL/XML
18	03.05.2018	No lecture!
19	10.05.2018	No lecture!
20	17.05.2018	No lecture!
21	24.05.2018	XML Indexing & XML Update
22	31.05.2018	XML Support in Databases Systems
24	14.06.2018	Examination

Literature

- M. Klettke, H. Meyer: XML & Datenbanken. dpunkt Verlag, 2002
- H. Schöning: XML und Datenbanken - Konzepte and Systeme. Carl Hanser Verlag, 2002
- M. Seemann: Native XML-Datenbanken im Praxiseinsatz. Software & Support Verlag, 2003
- A. B. Chaudhri, A. Rashid, R. Zicari: XML Data Management: Native XML and XML-Enabled Database Systems. Addison-Wesley, 2003
- S. Abiteboul, P. Buneman, D. Suciu: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann, 1999
- G. Powell: Beginning XML Databases. Wiley / Wrox, 2007

Chapter 1

Introduction and Motivation

Motivation of XML via

1) HTML

2) Databases

3) Information Retrieval

HTML vs. XML

```
<html>
<head>
<title>The Tragedy of Hamlet, Prince of Denmark</title>
</head>
<body>
<h1>The Tragedy of Hamlet, Prince of Denmark</h1>
<h2>Dramatis Personae</h2>
<p>CLAUDIUS, king of Denmark. </p>
<p>HAMLET, son to the late, and nephew to the present
      king. </p>
<h2>Scene Description</h2>
<p>SCENE Denmark.</p>
<h2>Subtitle</h2>
<p>HAMLET</p>
<h2>ACT I</h2>
<h3>SCENE I. Elsinore. A platform before the
      castle.</h3>
<p><i>FRANCISCO at his post. Enter to him
      BERNARDO</i><br>
BERNARDO: Who'sthere?</p>
…
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<play>
 <title>The Tragedy of Hamlet, Prince of
      Denmark</title>
 <personae>
  <title>Dramatis Personae</title>
  <persona>CLAUDIUS, king of Denmark.</persona>
  <persona>HAMLET, son to the late, and nephew to the
      present king.</persona>
 </personae>
 <scndescr>SCENE  Denmark.</scndescr>
 <playsubt>HAMLET</playsubt>
 <act>
  <title>ACT I</title>
  <scene>
   <title>SCENE I.  Elsinore. A platform before the
      castle.</title>
   <stagedir>FRANCISCO at his post. Enter to him
      BERNARDO</stagedir>
   <speech> <speaker>BERNARDO</speaker>
   <line>Who'sthere?</line></speech>
   …
  </scene>
  …
 </act>
 …
</play>
```

Motivation via HTML

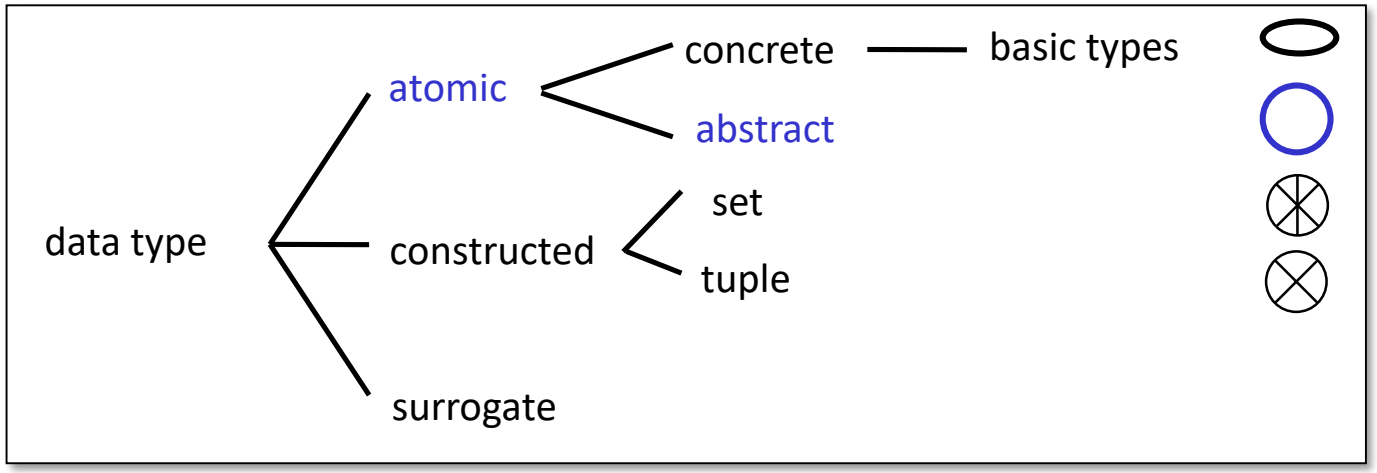
- **Web provides information embedded in HTML documents**
  - **Positive:**
    - Platform-independent, producer-independent
    - Always accessible from anywhere
    - Easy access/download via HTTP
    - Applications can easily generate HTML documents
  - **Negative:**
    - HTML document processing is cumbersome within applications since HTML primarily describes layout but not structure of documents
    - HTML provides only a fixed set of structuring elements, which can neither be updated nor extended by users
    - Only simple, text-based queries supported
- **XML tackles the disadvantages of HTML without giving up it advantages**
  - Data transfer is based on Internet protocols such as HTTP
  - Structuring elements can be user-defined
  - Structuring elements determine not only layout but also structure
  - User-defined structuring elements support arbitrary complex queries

XML as Web Standard

- **XML documents**
  - Generated by applications
  - Consumed by applications
  - Used for data exchange
- **Data description and exchange as main scenarios for XML usage**
  - Weakly structured data such as Shakepears dramas available as text documents
  - Strongly structured data such customer/order data coming from a relational database
- **Power of XML goes far beyond HTML**
  - XML vs. classic data models vs. information retrieval models: How they relate?
  - From semantic to semi-structured data models: A schema capable of describing all objects?

Motivation via Databases and Semantic Data Models

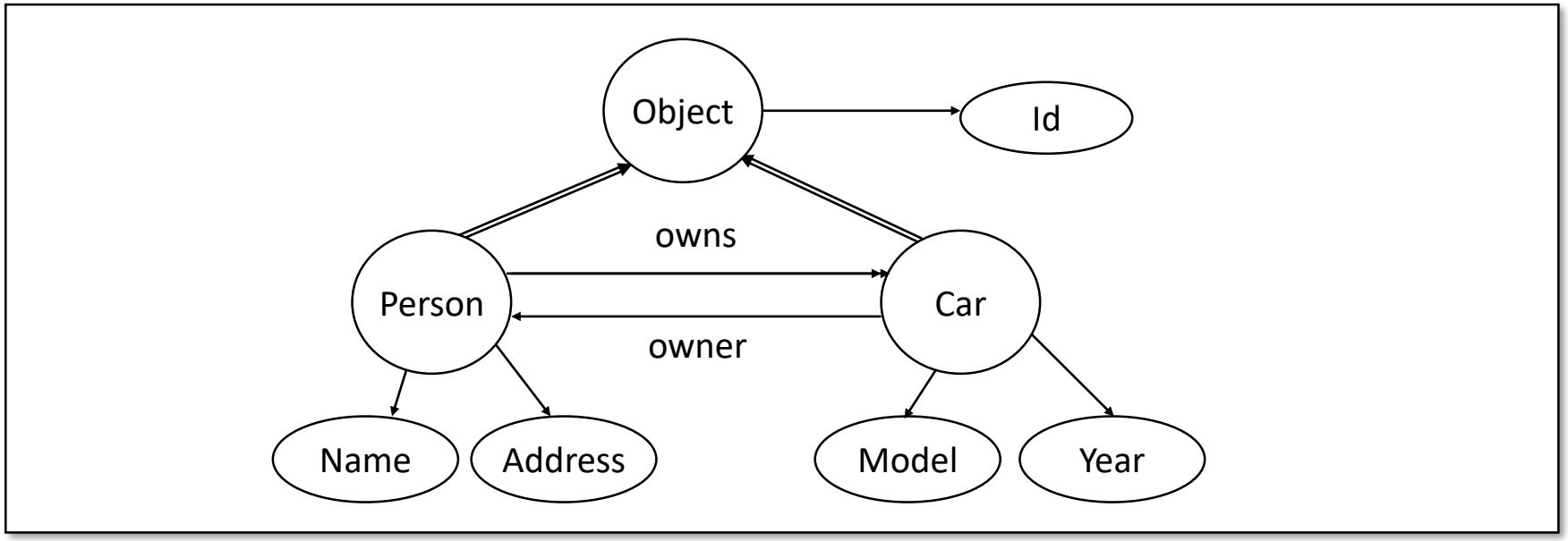
- **Semantic data models distinguish between**
  - Schema and instance
  - Type and variable



- **Object-oriented databases focus on atomic, abstract data types**
- **Relational databases build upon constructed data types**
- **Schema definition implicitly determines elementary integrity constraints**

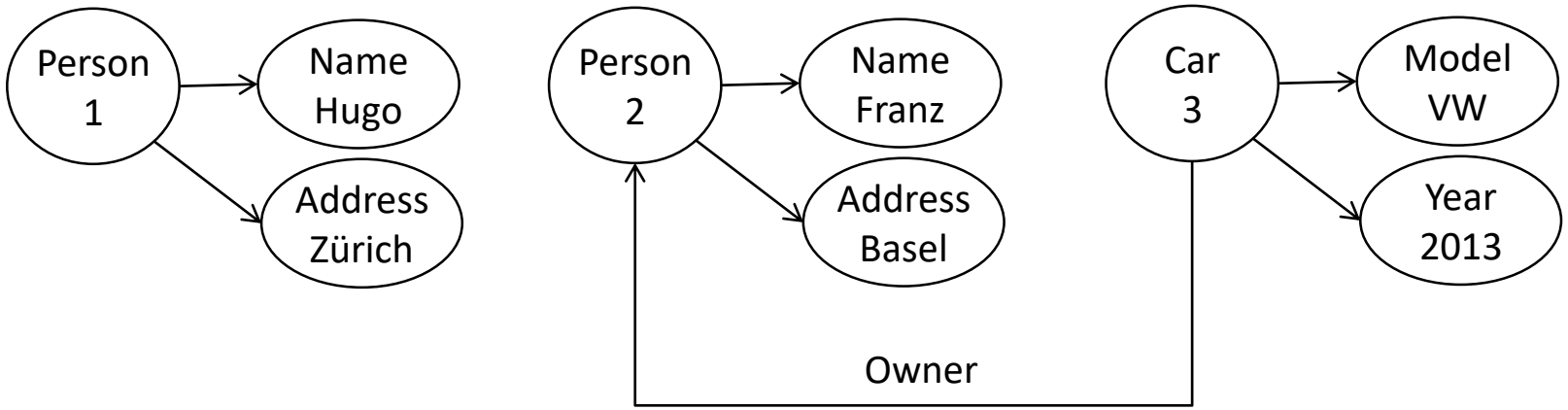
Simple Data Modeling Example (1)

In our mini world, called PC, we distinguish persons and cars. Cars are described by their model, fabrication year, and owner. Persons are characterized by their name and address. Every car belongs to exactly one person while a person can own several cars.



Simple Data Modeling Example (2)

Suppose there are following instances respresenting two persons and a car



The relational representation would looks as follows:

Person	Id	Name	Address
	1	Hugo	Zürich
	2	Franz	Basel

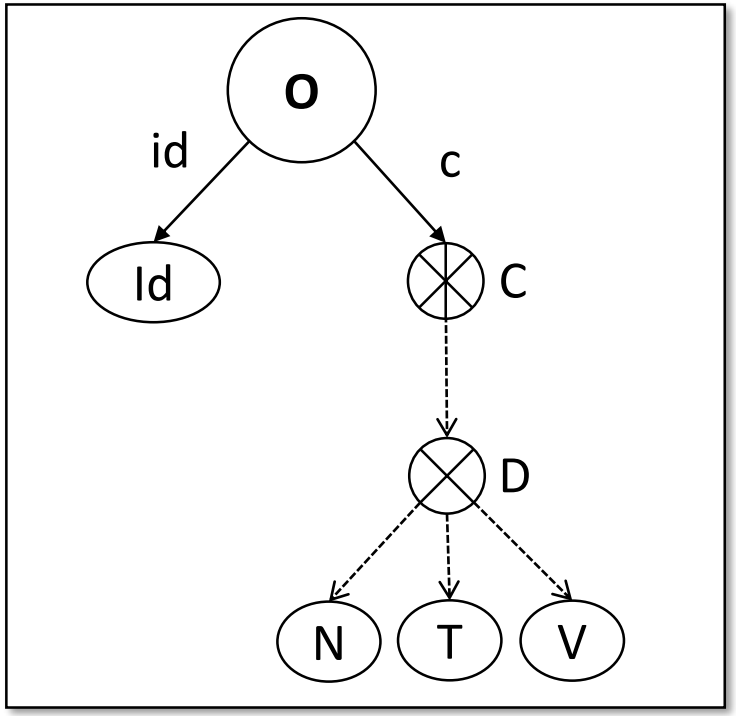
Car	Id	Model	Year	Owner
	3	VW	2013	2

Self-Describing Data

- **Refers to data with integrated meta-data which annotates (describes) the data**
- **Remember: databases strictly separate between meta-data (schema) and data (instance)**
  - Advantage: logical independence as basis for optimization
  - Disadvantage: instance without schema complicates data exchange
- **Data should be self-describing! But how?**
- **Idea: Introduce a „semi-structured“ data model**
  - Use a descriptor-oriented database schema
  - Uniform for all applications ⇨ hence often named as „meta schema“

Simple Descriptor-Oriented Data Model

- **Each object has an id and a set C of descriptors D describing the characteristics of the object**
- **Each descriptor D has a name N, a type T, and a value V**



Fixed schema („meta schema“):
- fixed names Id, C, D, N, T, V
- application-independent

- **Each object can be described by its characteristics**
- **Name and type information are part of the instance: Self-Description!**

Simple Descriptor-Oriented Data Model: Relational Representation

Entity	Id	Name	Type	Value
	1	Name	String	Hugo
	1	Address	String	Zürich
	2	Name	String	Franz
	2	Address	String	Basel
	3	Model	String	VW
	3	Year	Integer	2013
	3	Owner	Integer	2

Person

Person

Car

Entity	Id	Name	Type	Value
	1	Name	String	Hugo
	1	Address	String	Zürich
	1	Class	String	Person
	2	Name	String	Franz
	2	Address	String	Basel
	2	Class	String	Person
	3	Model	String	VW
	3	Year	Integer	2013
	3	Owner	Integer	2
	3	Class	String	Car

Add further descriptor for each entity to maintain the class information

Simple Descriptor-Oriented Data Model: Advantages and Limitations

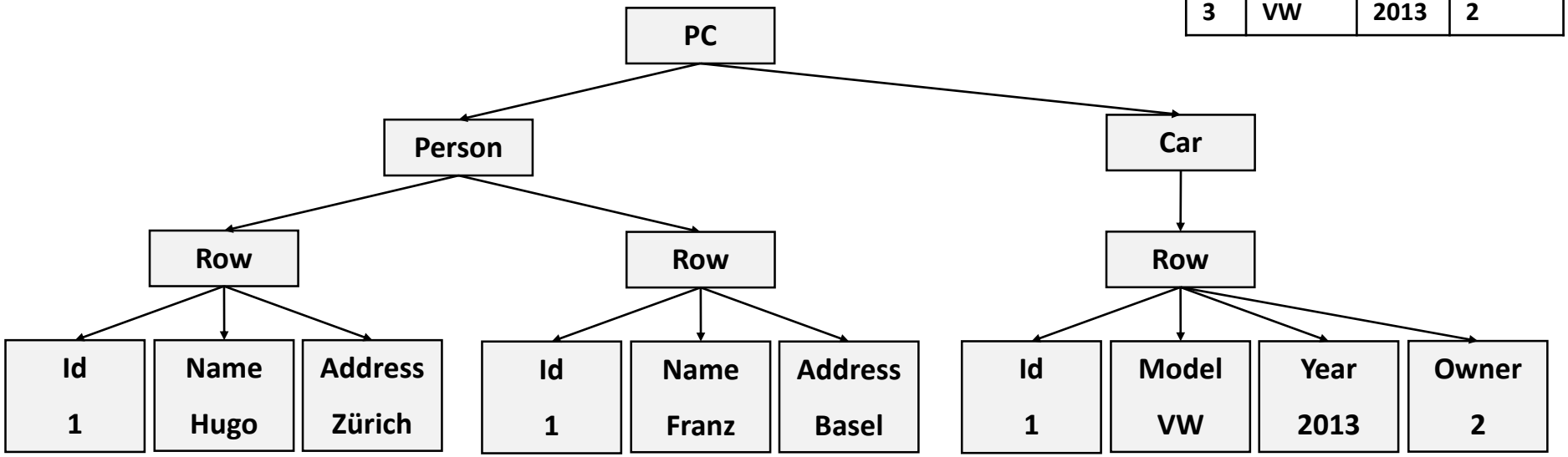
+ **Each object described by variable set of characteristics (descriptors)**
  - No null values
  - No forced introduction of sub types when more characteristics are known
  - Each individual can be described individually

- **Object names (classfication) disappear: must explicitly be defined as additional descriptor**
  - Which type of an object is the object with the id=1? Person or Car?
  - No comprising context (class)

XML Representation of Relational Data: Variant 1

Person	Id	Name	Address
	1	Hugo	Zürich
	2	Franz	Basel

Car	Id	Model	Year	Owner
	3	VW	2013	2

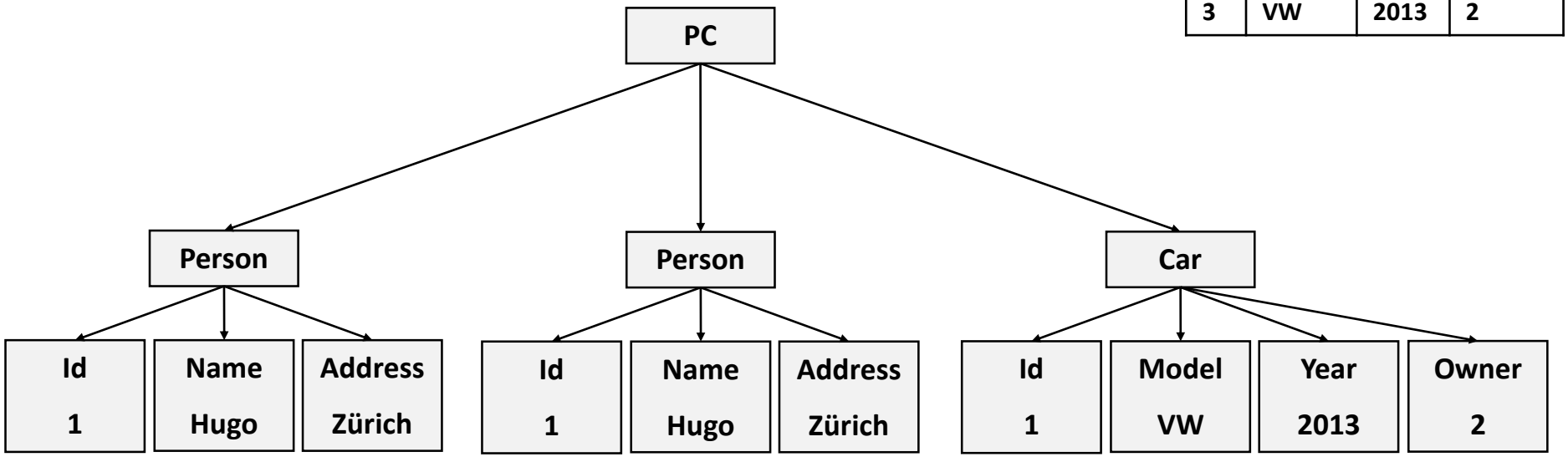



```
<PC>
   <Person>
      <Row> <Id>1</Id> <Name>Hugo</Name> <Address>Zürich</Address> </Row>
      <Row> <Id>2</Id> <Name>Franz</Name> <Address>Basel</Address> </Row>
   </Person>
   <Car>
      <Row> <Id>3</Id> <Model>VW</Model> <Year>2013</Year> <Owner>2</Owner></Row>
   </Car>
</PC>
```

XML Representation of Relational Data: Variant 2

Person	Id	Name	Address
	1	Hugo	Zürich
	2	Franz	Basel

Car	Id	Model	Year	Owner
	3	VW	2013	2

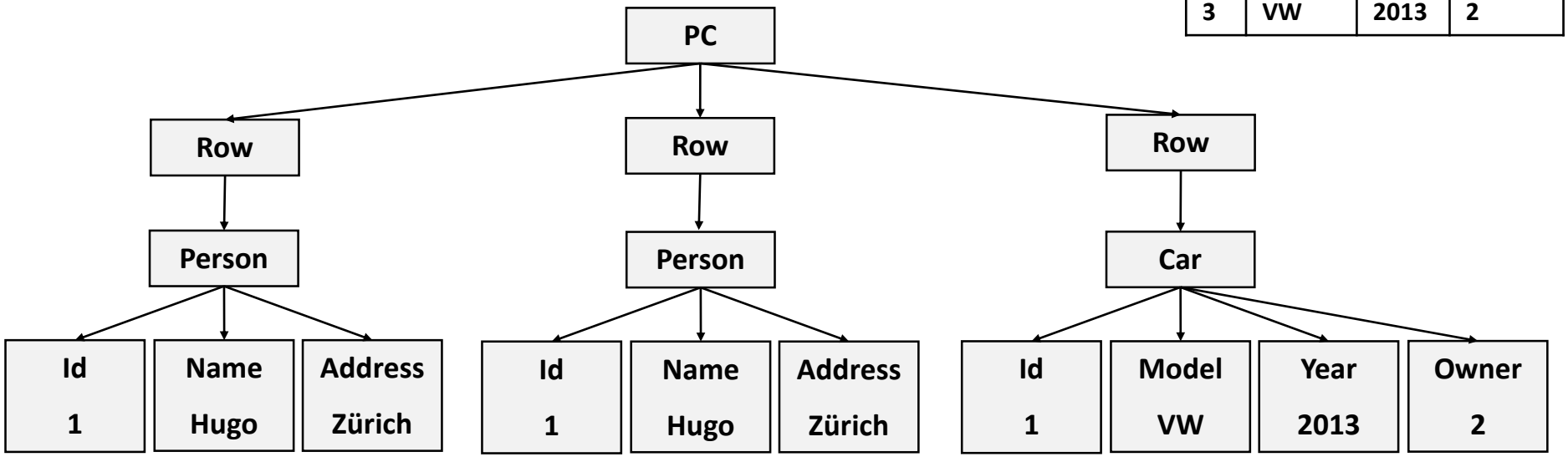



```
<PC>
   <Person> <Id>1</Id> <Name>Hugo</Name> <Address>Zürich</Address> </Person>
   <Person> <Id>2</Id> <Name>Franz</Name> <Address>Basel</Address> </Person>
   <Car> <Id>3</Id> <Model>VW</Model> <Year>2013</Year> <Owner>2</Owner></Car>
</PC>
```

XML Representation of Relational Data: Variant 3

Person	Id	Name	Address
	1	Hugo	Zürich
	2	Franz	Basel

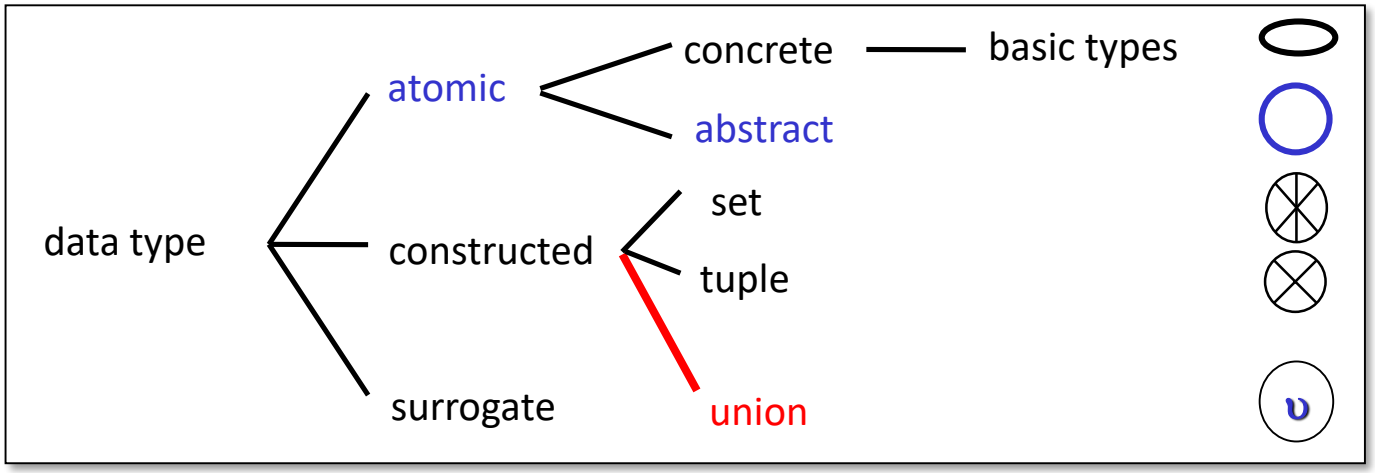
Car	Id	Model	Year	Owner
	3	VW	2013	2




```
<PC>
   <Row> <Person> <Id>1</Id> <Name>Hugo</Name> <Address>Zürich</Address> </Person></Row>
   <Row> <Person> <Id>2</Id> <Name>Franz</Name> <Address>Basel</Address> </Person></Row>
   <Row> <Car> <Id>3</Id> <Model>VW</Model> <Year>2013</Year> <Owner>2</Owner> </Car></Row>
</PC>
```

Semi-Structured Data Models

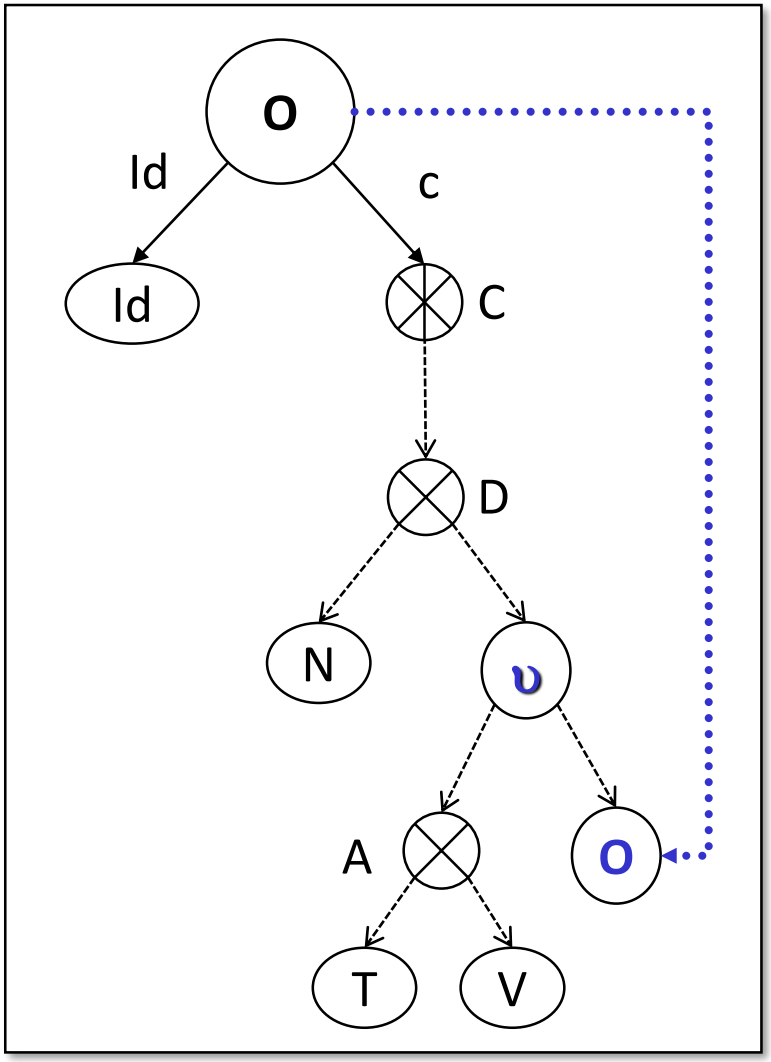
... evolve by extending the simpe descriptor-oriented model by a union data type:



Union operator ∪ introduced as additional type constructor!

In the context of XML, often also named sequence (ordered set)

Extended Descriptor-Oriented (Meta) Schema



- **An object O is described by set C of descriptors D which consist of a name N and either**
  - another object O or
  - a type-value pair A with the type T and the value V

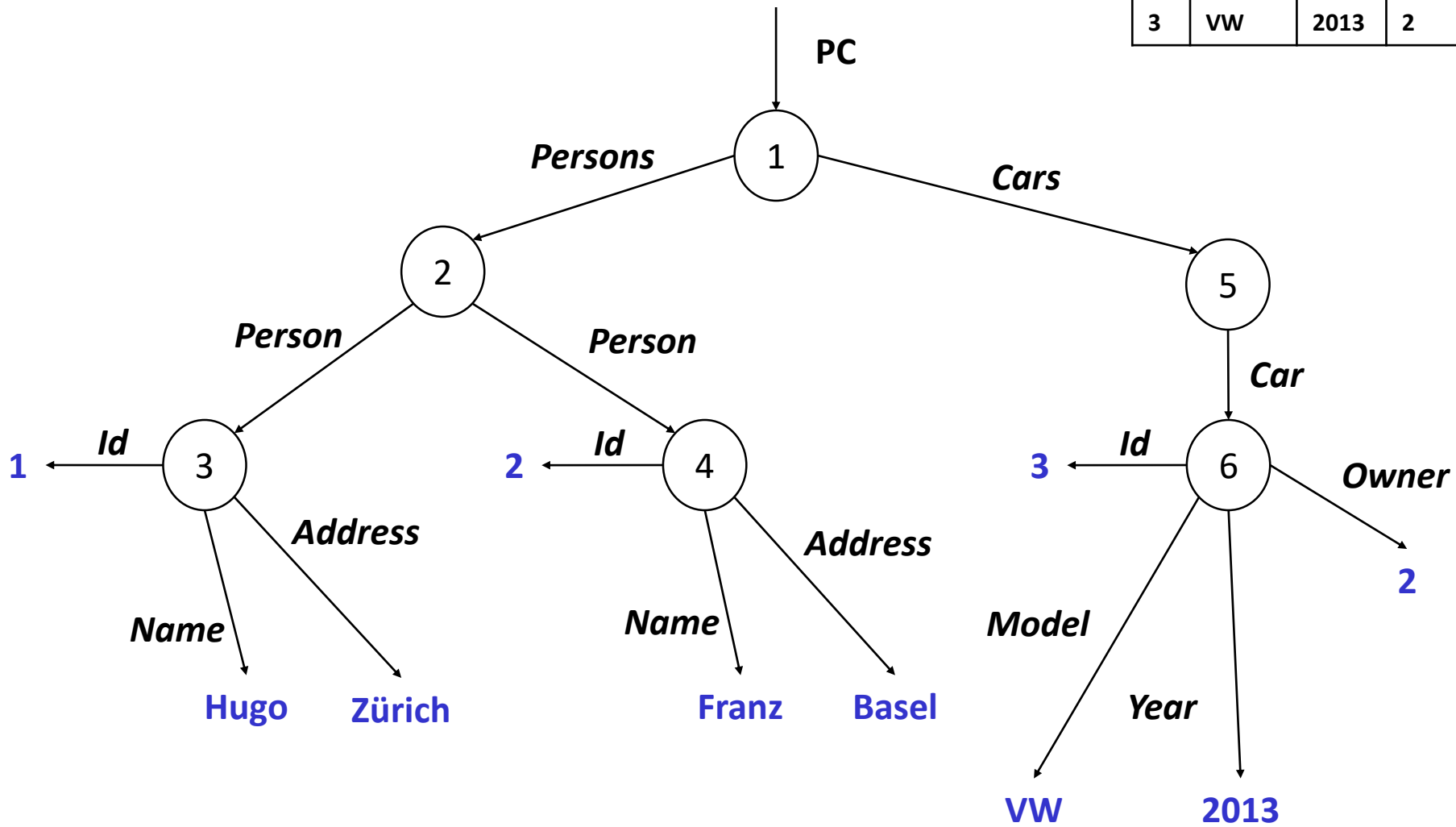
➔ **Flexibility introduced by union type!**

➔ **Semi-structured data model**

Tree Representation with Edge and Node IDs

Person	Id	Name	Address
	1	Hugo	Zürich
	2	Franz	Basel

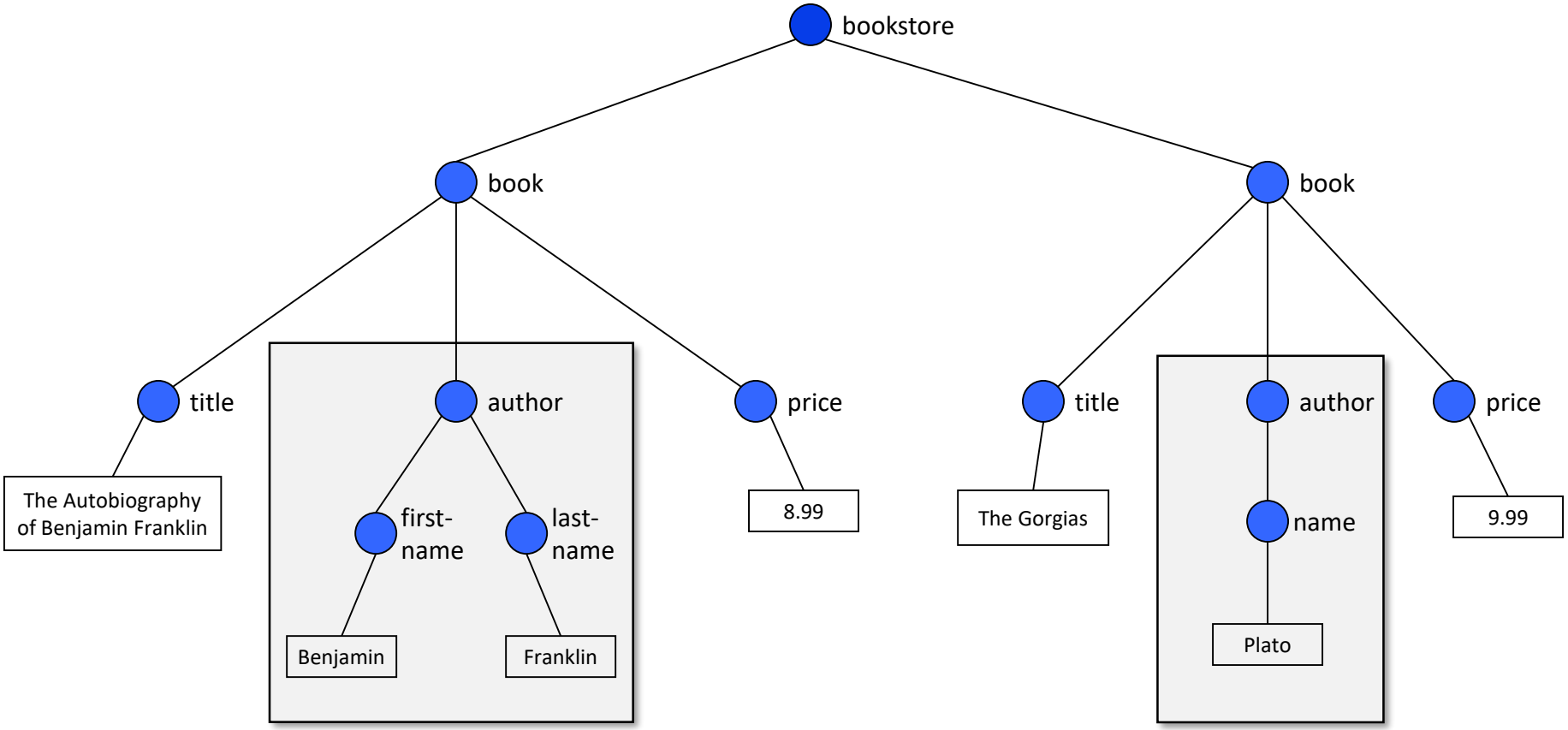
Car	Id	Model	Year	Owner
	3	VW	2013	2



Flexibility of Semi-Structured Data Models

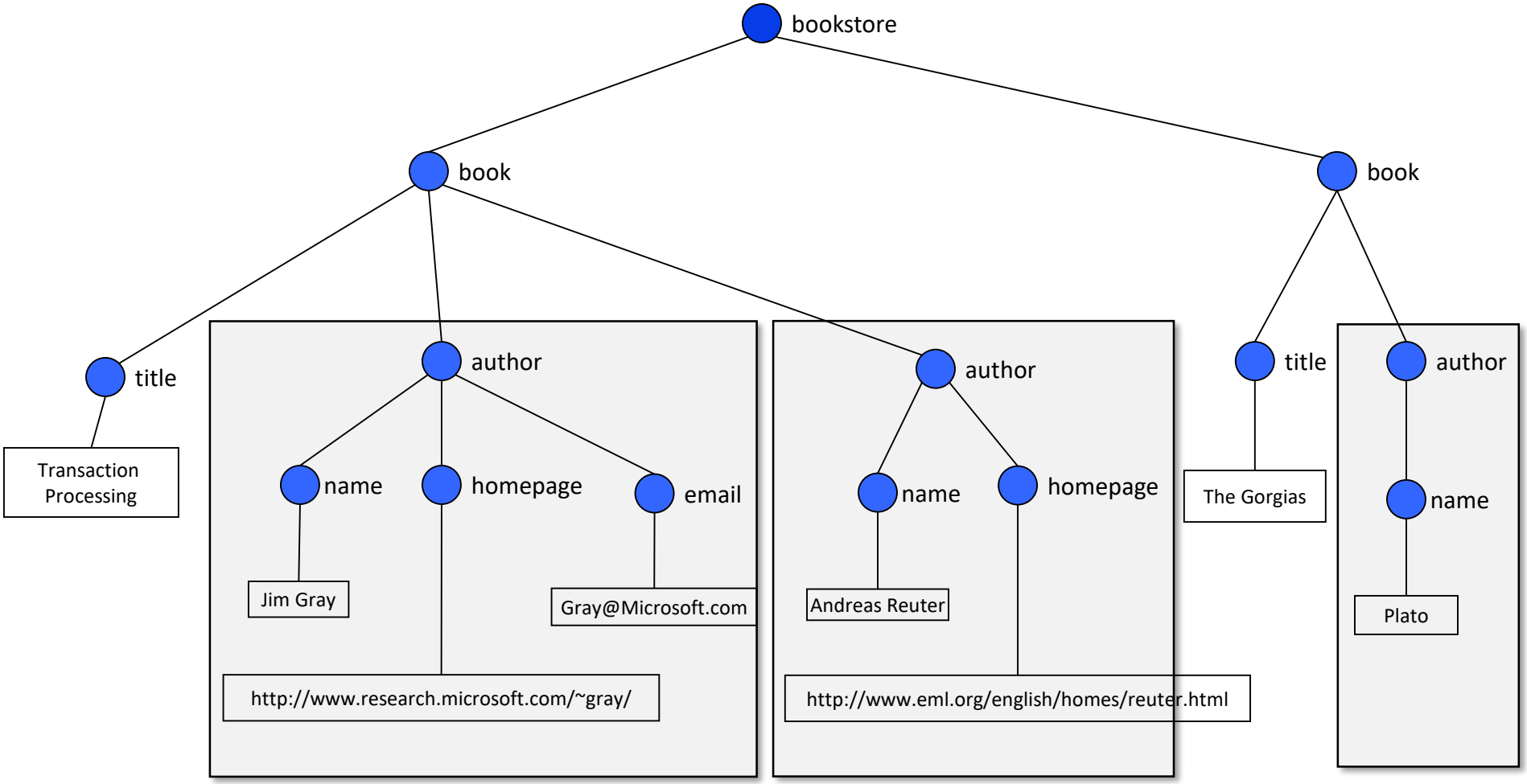
- **Data must not be compliant to a fixed schema a priori**
- **Reasons against explicit structuring of data**
  - Not meaningful
  - Not uniquely possible
  - Not known (completely)
  - Complete structuring to expensive
- **Data not always easily mappable to simple tables**
  - Optional characteristics
  - Repetition of elements
  - Element order potentially relevant
  - Structuring however desired, e.g. for fast data access or fine-grained search

Semi-Structured Data Models: Example (1)



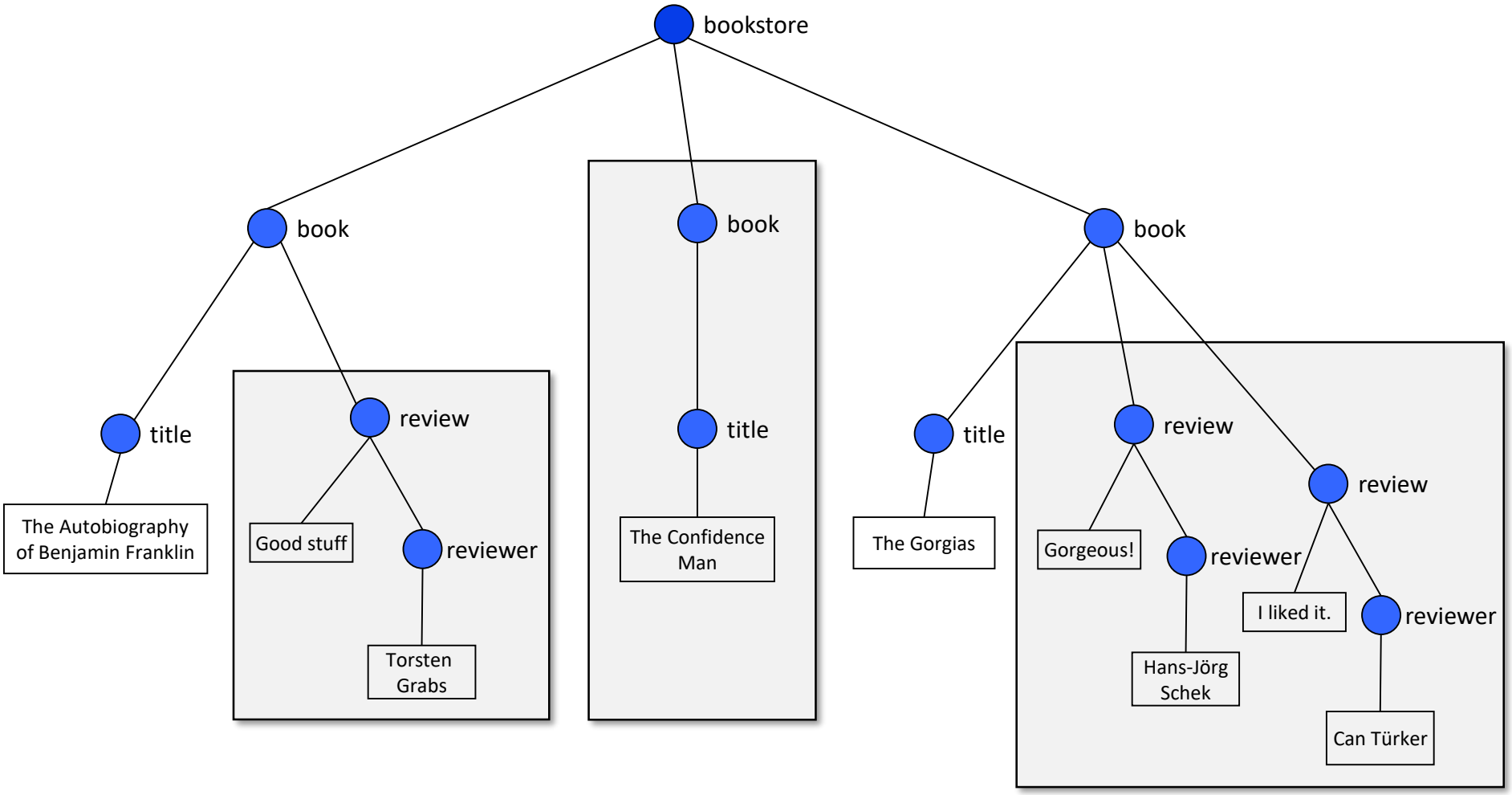
Authors do not have always a first and a last name

Semi-Structured Data Models: Example (2)



Avoid null values by allowing different author structures

Semi-Structured Data Models: Example (3)



Books can have arbitrary reviews but must not have one

Conclusions: Motivation for XML via Semantic Data Models

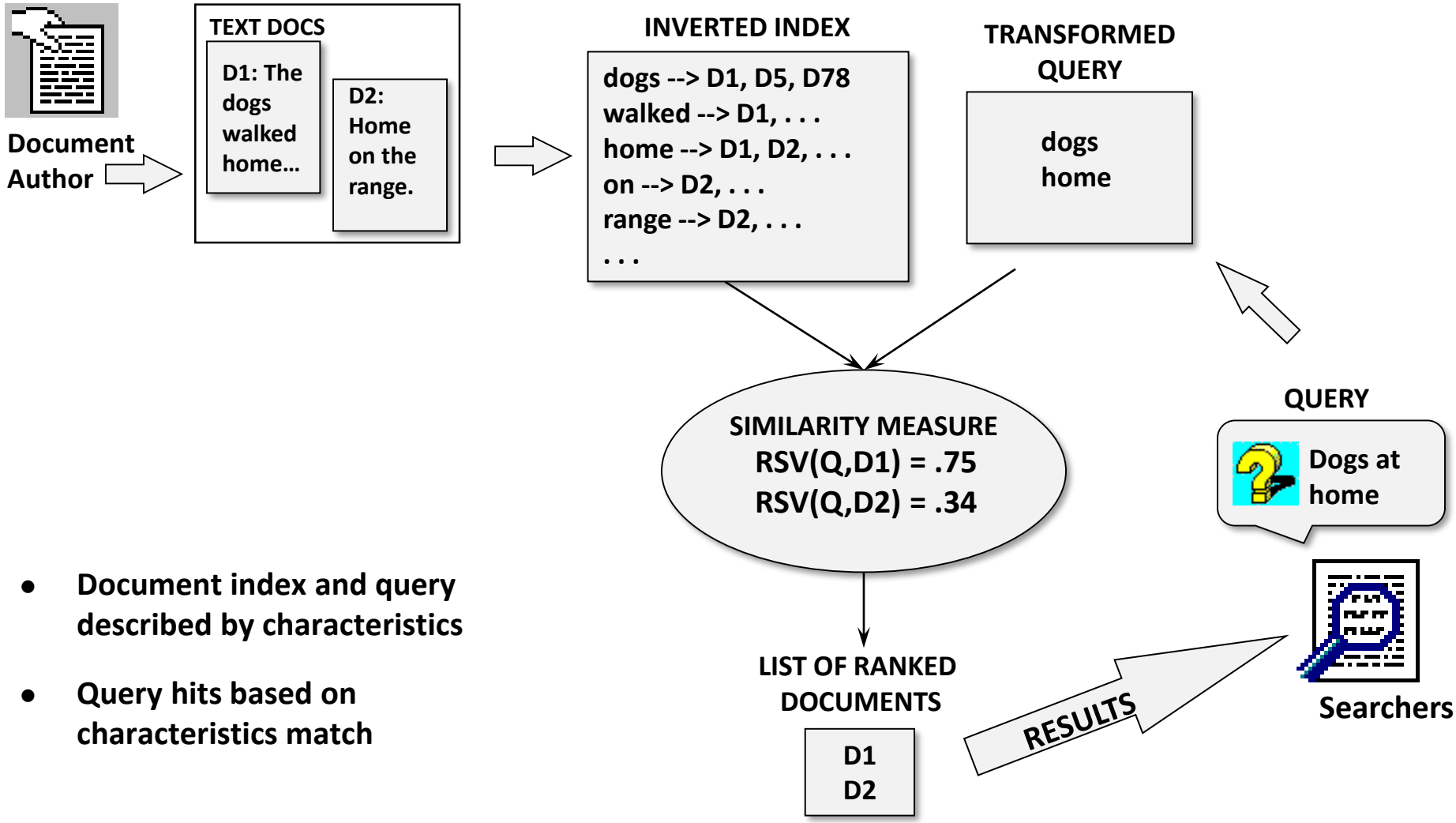
- **Semi-structured data model supports flexible structures**
  - XML documents of a given type can be structured differently

- **Self-description avoids separation of schema and instance**
  - XML documents can describe data as well as data structure

Motivation of XML via Information Retrieval

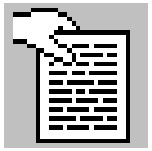
- **Database systems manage well-structured data**
- **Information-Retrieval-Systems manage less-structured documents**
  - Text documents: arbitrary content length, potentially structured in paragraphs
  - Hyper-Text documents: additionally contain links to related text positions
  - Multi-Media documents: may include images, audios, and videos
  - Documents may significantly differ in structure and size
- **Compared to flat information retrieval models, XML supports**
  - Better structuring possibilities
  - Integrated meta-data description
- **Information-Retrieval may benefit from XML**
  - Structure information is explicitly available in contrast to HTML
  - Access granularity is more fine-granule in contrast to flat document structures

Information Retrieval on Flat Documents



- Document index and query described by characteristics
- Query hits based on characteristics match

Information Retrieval on Structured Documents



Document Author

TAGGED DOCS

```
<DOC>
<DOCNO> D1
</DOCNO>
<TEXT> The
dogs walked
home. …
</TEXT>
</DOC>
```

```
<DOC>
<DOCNO>D2
</DOCNO>
<TEXT> Home
on the range.
</TEXT>
</DOC>
```

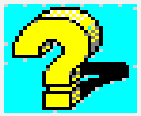
INVERTED INDEX

```
dogs --> D1, D5, D78
walked --> D1, . . .
home --> D1, D2, . . .
on --> D2, . . .
range --> D2, . . .
. . .
```

TRANSFORMED QUERY

dogs
home

QUERY



"Dogs at home" in /DOC/TEXT

SIMILARITY MEASURE

RSV(Q,D1) = .75

RSV(Q,D2) = .34

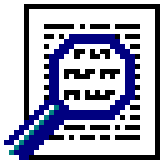
Can we use the same ranking function as for flat documents?

LIST OF RANKED ELEMENTS

D1: /DOC/TEXT

D2: /DOC/TEXT

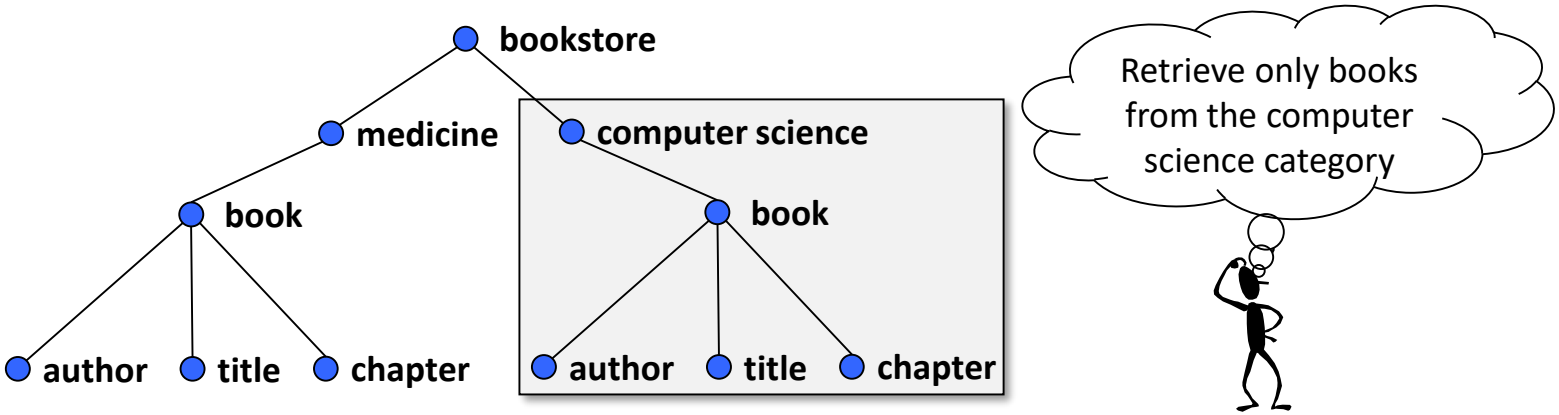
RESULTS



Searchers

Conclusions: Motivation of XML via IR

- **Information-Retrieval-Systems**
  - Support retrieval of relevant text fragments
  - Extract words and word sequences
  - Abstract from document structure
- **Information Retrieval with XML**
  - XML makes structure of certain text elements explicit
  - Finer access and retrieval granularity, e.g., restriction to certain XML elements
  - Increases document search hit potential by exploiting XML document structure
  - Better retrieval results by improved ranking based on integrated structure information
- **Tradional IR ranking approaches such as the vector-space model deliver better quality when document structure is included**



Conclusions: Motivation for XML

General advantages

- Platform-independent
- Producer-independent
- Language-independent
- Unicode-based („international“)
- Extensible
- Usable at all application levels
  - Storage
  - Application logic
  - Presentation

Advantages from business perspective

- Support extended search possibilites on distributed, heterogenous data
- Allow new functionality on existing data
- Ease data transformation
- Connect applications
- Reduce application development time
- No (little) license costs since many XML tools are freely available

Motivation for XML in Databases

- **XML as data exchange format**
  - XML documents describing business relevant data
  - Archiving original documents
  - Transformation into other formats
  - Storing XML data in pipeline steps
- **Applications on XML data**
  - XML as internal data structure, e.g., to describe configuration files, application data, or style sheets
  - Requirements for adequat XML storage?
- **Advantages of storing XML in databases**
  - Powerful and efficient search functions
  - Transactional, persistent storage
  - Concurrency control
  - Access control