Databases and data model

Lecturer: Jiaheng Lu

Autumn 2016

Outline

- History of databases (Why we use databases)
- Data models
  - Three features of data models
  - Relational model
  - Semi-structure model
    - XML model
    - JSON model
  - Graph model

Data storage and history

Before-1950s Data was stored as paper records

Lot of time was wasted. e.g. when searching. Therefore inefficient.

Magnetic tapes and hard disk

- 1950s and early 1960s: Data processing using magnetic tapes for storage

Magnetic tapes and hard disk

- 1950s and early 1960s: Data processing using magnetic tapes for storage
- Late 1960s and 1970s: Hard disks allow direct access to data
- Data stored in files in the above two devices.

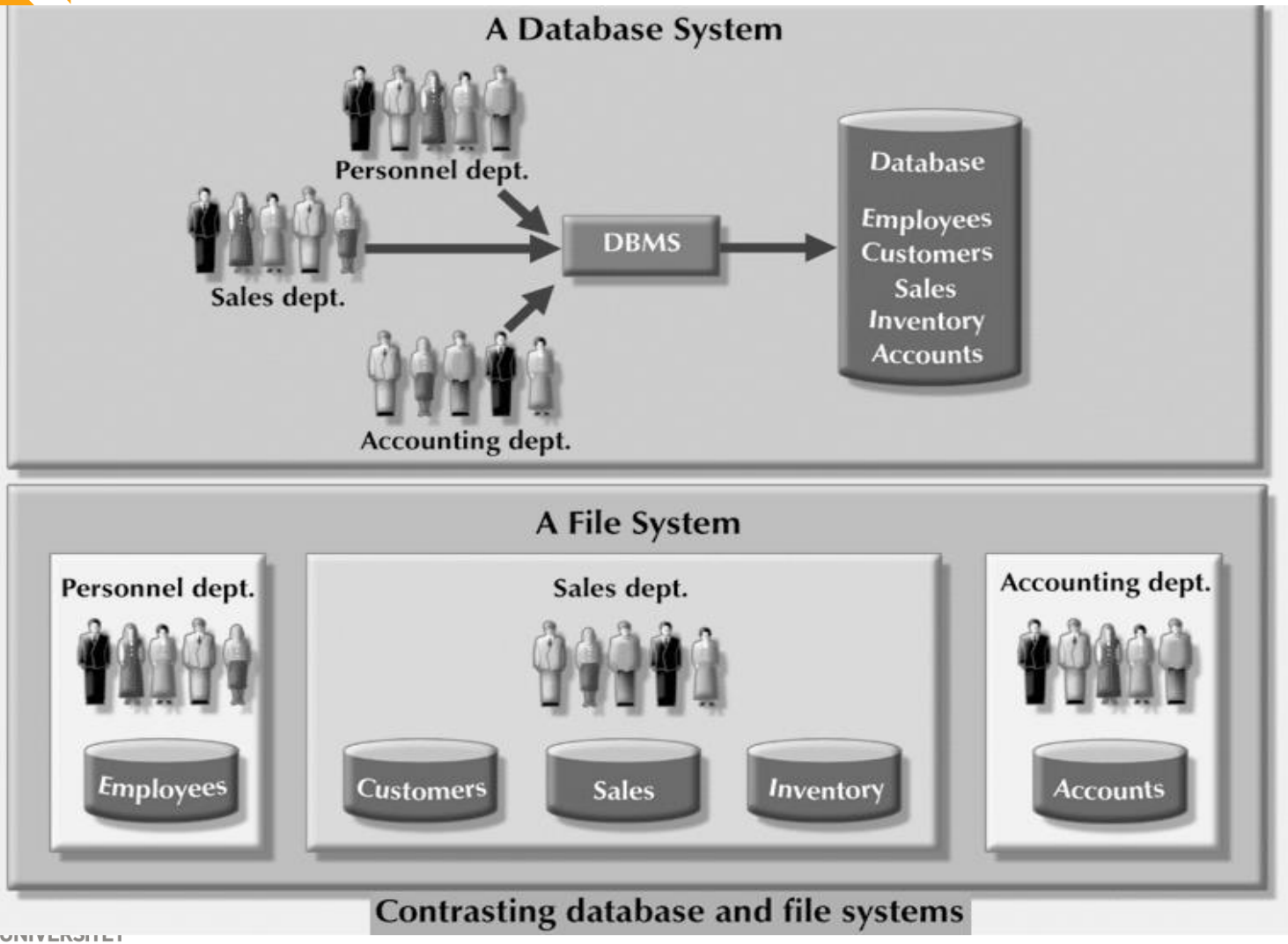
Drawbacks of file system

- Data sharing: Each program has its own data format. Data cannot be easily accessed by other programs.
- Data duplication
- Data independence on programs

Database Systems

- Problems inherent in file systems make using a database system desirable
- File system
  - Many separate and unrelated files
- Database
  - Logically related data stored in a single logical data repository

Database Systems and File System



Contrasting database and file systems

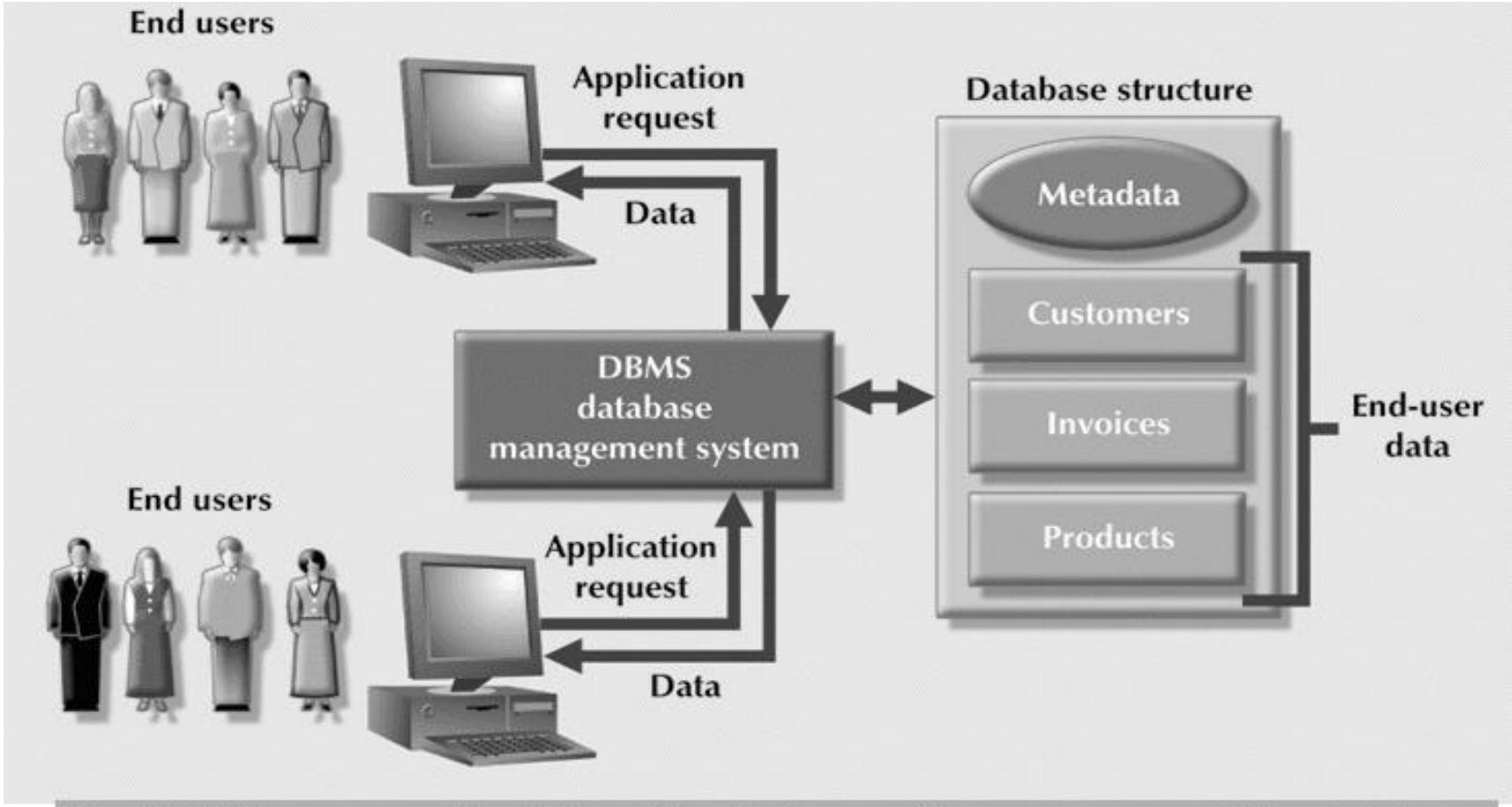
Introducing the Database and the DBMS

- Database—shared, integrated computer structure that stores:
  - End user data (raw facts)
  - Metadata (data about data)

Introducing the Database and the DBMS (continued)

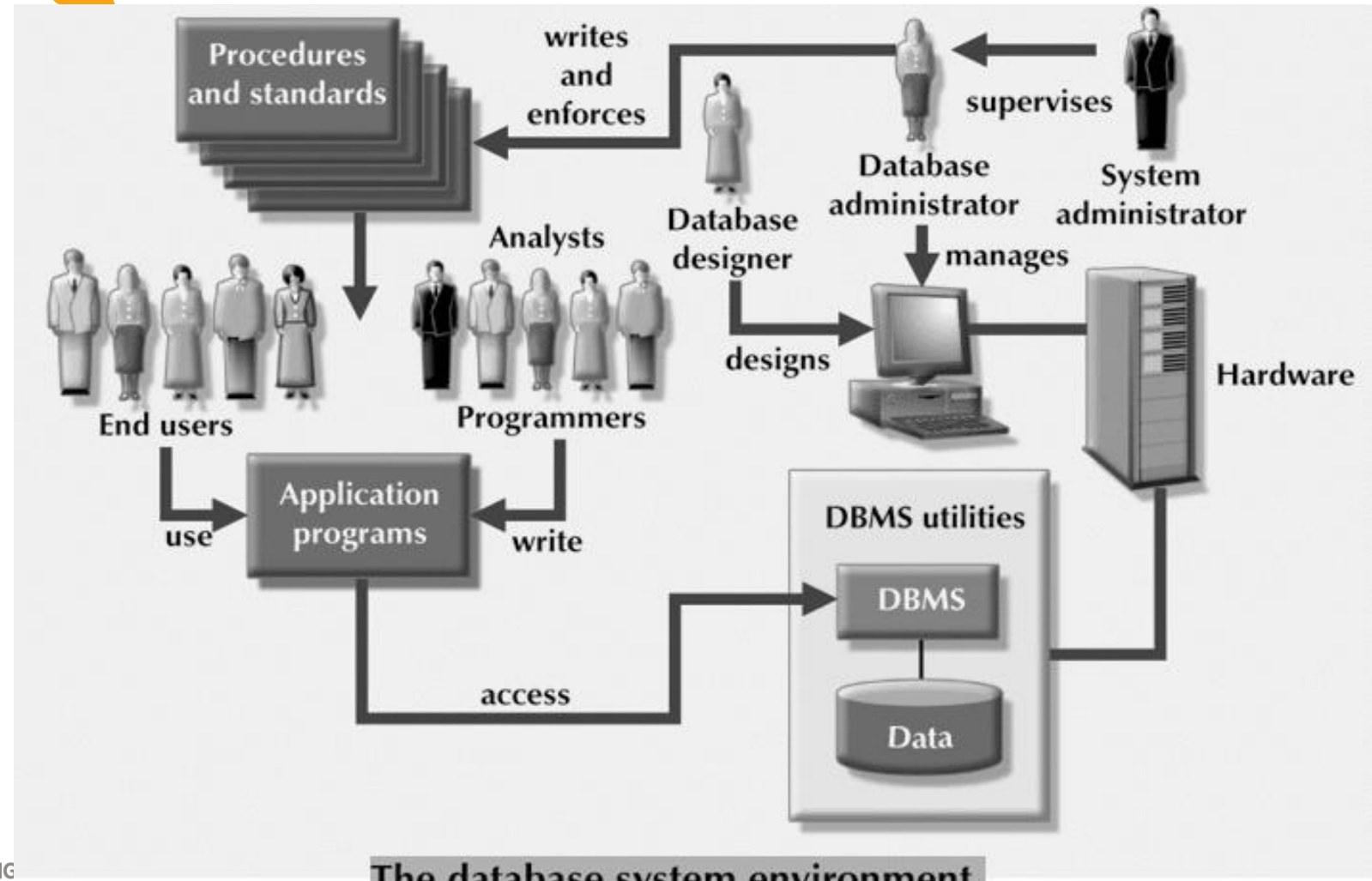
- DBMS (database management system):
  - Collection of programs that manages database structure and controls access to data
  - Possible to share data among multiple applications or users
  - Makes data management more efficient and effective

DBMS



The DBMS manages the interaction between the end user and the database

The Database System Environment (continued)



The database system environment

- DBMS’s were developed to address file systems’ inherent weaknesses

- Data is independent of the program
- Data can shared between two programs
- Data duplication can be removed by database design

- Watch a video on the young history of database systems.
- Youtube link:
- https://www.youtube.com/watch?v=nowcBsX_4kc

Outline

- History of databases (Why we use databases)

- Data models
  - Three features of data models
  - Relational model
  - Semi-structure model
    - XML model
    - JSON model
  - Graph model

Data Model: What's a model?

- A data model is a representation of reality
- It's used to define the storage and manipulation of a data base.

A Map Is a Model of Reality

Model features

- A model is a means of communication
- Users of a model must have a certain amount of knowledge in common
- A model on emphasized selected aspects
- A model is described in some language
- A model can be erroneous

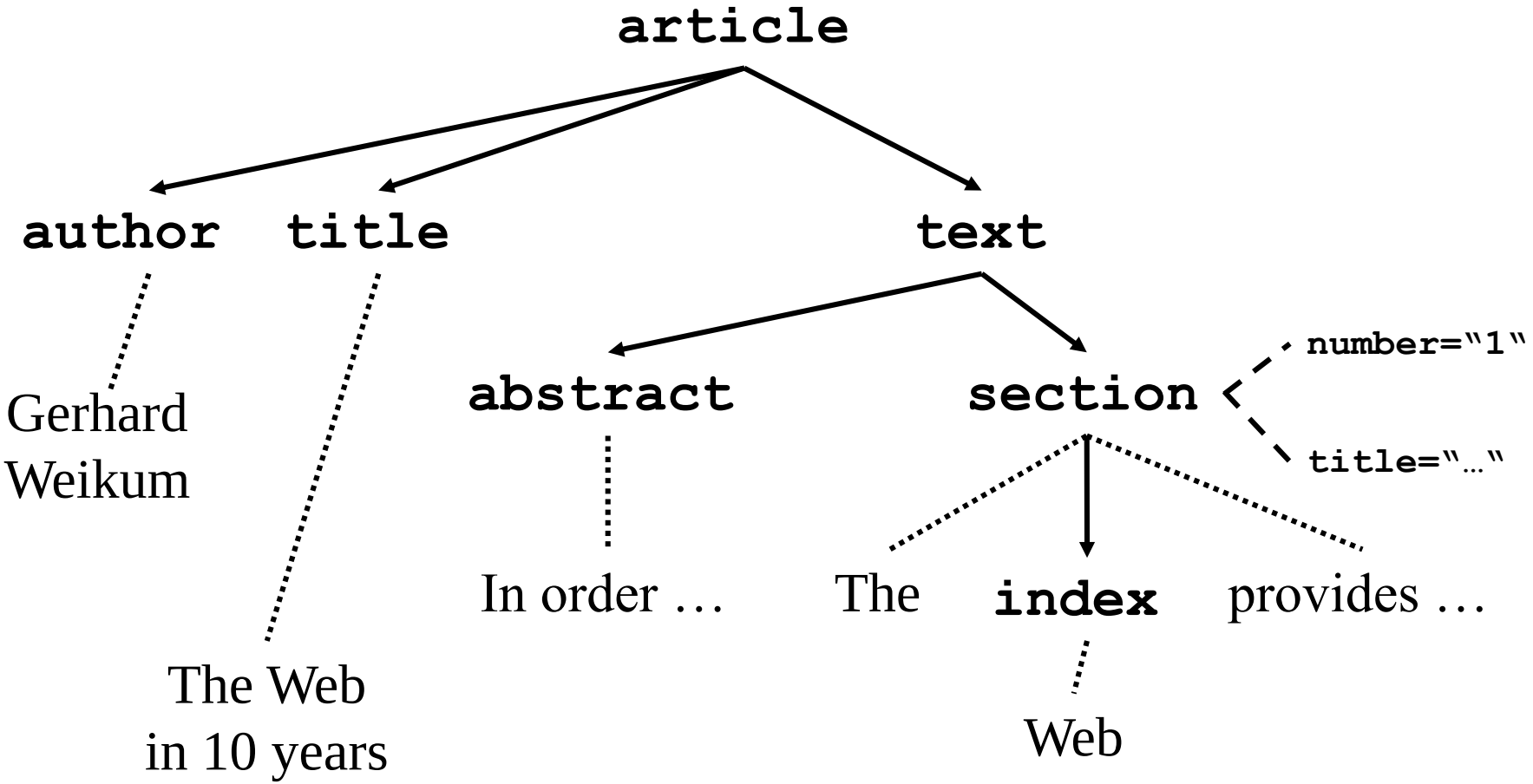
Data model describes three features of data

- Structure: the structure of the data stored within
- Operations: facilities for manipulation of the data.
- Constraints: the constraints of data values

A table structures of relational model

Name	Age	Gender	Job
John	32	Male	Engineer
Mary	27	Female	Doctor
Anna	57	Female	Teacher

A tree structures of XML model



Examples of operations

- Subsetting
  - Given a condition and a set of data, find a subset of data which satisfy the condition

Name	Age	Gender	Job
John	32	Male	Engineer
Mary	27	Female	Doctor
Anna	57	Female	Teacher

  - Given a condition Age < 40

Examples of operations

- Subsetting
  - Given a condition and a set of data, find a subset of data which satisfy the condition
- Substructure extracting
  - Extract from each data item a part of structure as specified by a condition

Name	Age	Gender	Job
John	32	Male	Engineer
Mary	27	Female	Doctor
Anna	57	Female	Teacher

Examples of operations

- Subsetting
  - Given a condition and a set of data, find a subset of data which satisfy the condition
- Substructure extracting
  - Extract from each data item a part of structure as specified by a condition
- Union and Join

Types of constraints

- Value constraints
  - E.g. age is never negative
- Uniqueness constraints
  - E.g. any course can have only one ID
- Cardinality constraints
  - E.g. each person can have at most three blood pressure records in the system

Types of constraints (cont'd)

- Type constraint
  - E.g. age is an integer
- Domain constraint
  - E.g. Month is between 1 to 12

Outline

- History of databases

- Data models
  - Three features of data model
  - Relational model
  - Semi-structure model
    - XML model
    - JSON model
  - Graph model

Relational data models

- Relational model is an approach to managing data using tables:
  - No duplicate tuples
  - Dissimilar tuples disallowed

Name	Age	Gender	Job
John	32	Male	Engineer
Mary	27	Female	Doctor
Anna	57	Female	Teacher
23.	87	Computer science	Teacher

Relational models
Operation: Join

- Join operation for two tables based on the names

Name	Age	Job
John	32	Engineer
Mary	27	Doctor
Anna	57	Teacher

Name	Hobby
John	Golf
John	Reading

Name	Age	Job	Hobby
John	32	Engineer	Golf
John	32	Engineer	Reading

Relational models
constraints

- Uniqueness constraints: key
  - E.g. any course can have only one ID
- Type constraint
  - E.g. age is an integer

Outline

- History of databases (Why we use databases)
- Data models
  - Three features of data models
  - Relational model
  - Semi-structure model
    - XML model
    - JSON model
  - Graph model

Semi-structured model

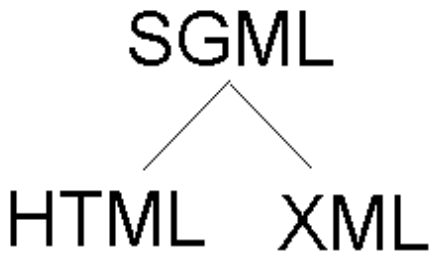
- The **semi-structured** model is a database model where there is no separation between the data and the schema, and the amount of structure used depends on the purpose.

Advantage:

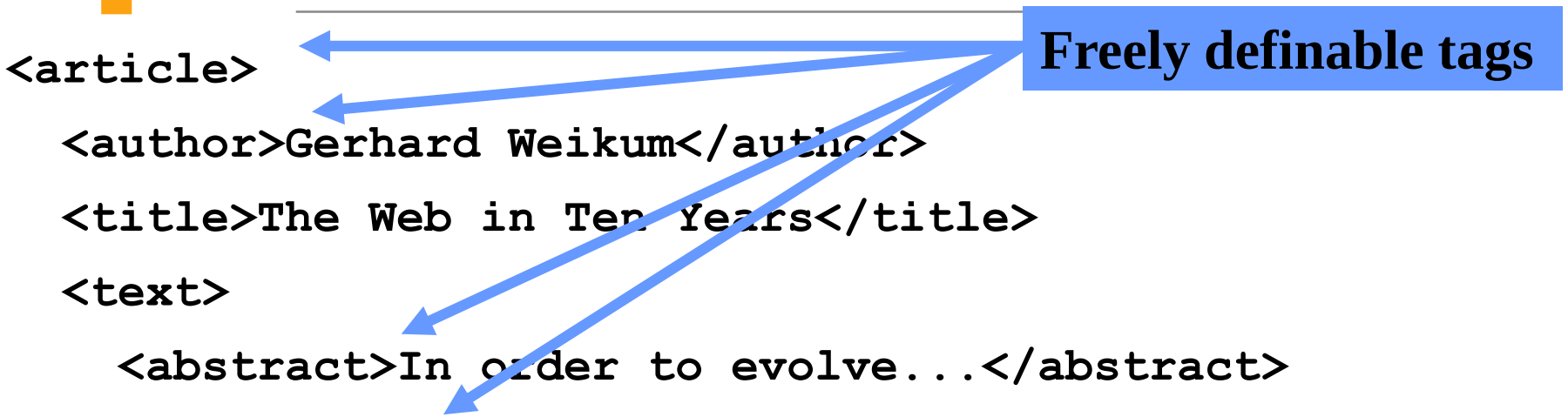
- The schema (with flexible formats) can easily be changed.

XML….

- *e***X***tensible* **M***arkup* **L***anguage*
- Based on Standard Generalized Markup Language (SGML)
- Version 1.0 introduced by World Wide Web Consortium (W3C) in 1998
- Bridge for data exchange on the Web

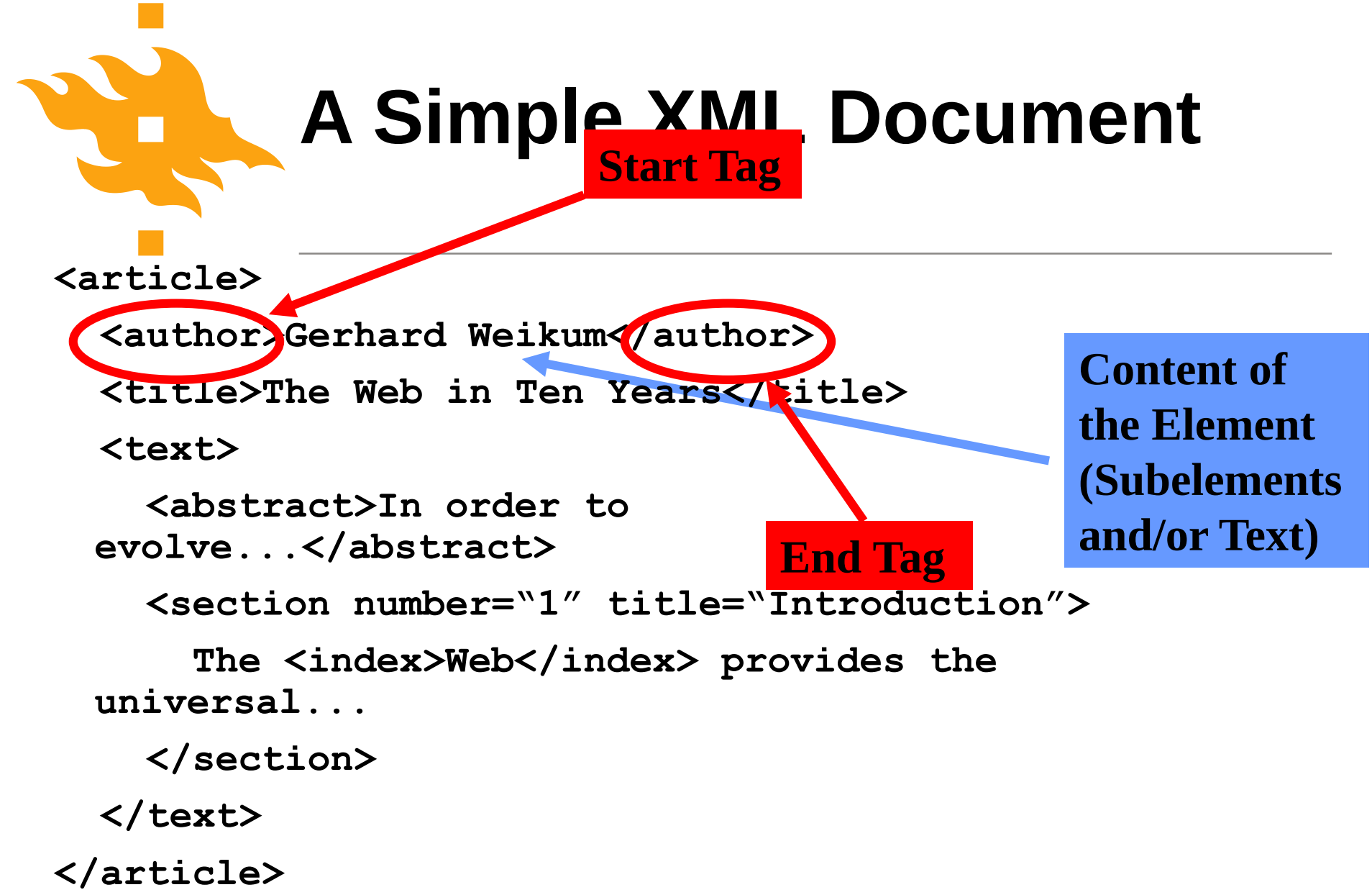


A Simple XML Document




```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

A Simple XML Document



A Simple XML Document

```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

Attributes with **name** and **value**

Elements in XML Documents

- (Freely definable) **tags**: `article`, `title`, `author`
  - with start tag: `<article>` etc.
  - and end tag: `</article>` etc.
- **Elements**: `<article> ... </article>`
- Elements have a **name** (`article`) and a **content** (. . .)
- Elements may be nested.
- Elements may be empty: `<this_is_empty/>`

Attributes

- Only one attribute with a given name per element
- Attributes have no structure, simply strings

Example:

```
<person born="1912-06-23" died="1954-06-
  07">
Alan Turing</person> proved that…
```

Document Type Definitions (DTD)

- An XML document may have an optional DTD.
- DTD serves as grammar for the underlying XML document, and it is part of XML language.

XML and DTD examples

- Consider an XML document:

```
<db><person><name>Alan</name>
                    <age>42</age>
                    <email>agb@usa.net </email>
        </person>
        <person>……</person>
        ……
</db>
```

XML and DTD examples

- DTD for it might be:

```
<!DOCTYPE db [
    <!ELEMENT db (person*)>
    <!ELEMENT person (name, age, email)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT age (#PCDATA)>
    <!ELEMENT email (#PCDATA)>
     <! AttributeList email (#PCDATA)>
 ]>
```

DTD (cont'd)

Occurrence Indicator:

Indicator	Occurrence	
(no indicator)	Required	One and only one
?	Optional	None or one
*	Optional, repeatable	None, one, or more
+	Required, repeatable	One or more

Important attribute types in DTD

- There are ten attribute types
- These are the most important ones:
  - PCDATA — The value is parse-able text data
  - ID — The value is a unique identifier
    - ID values must be legal XML names and must be unique within the document

Attribute list in DTD

- Recall that an attribute has the form
  <!ATTLIST *element-name name type requirement*>
- The *requirement* is one of:
  - A default value, enclosed in quotes
    - Example: <!ATTLIST degree CDATA "PhD">
  - #REQUIRED
    - The attribute must be present
  - #IMPLIED
    - The attribute is optional
  - #FIXED "value"
    - The attribute always has the given value
    - If specified in the XML, the same value must be used

Another example: XML

```
<?xml version="1.0"?>
<!DOCTYPE weatherReport SYSTEM
      "http://www.mysite.com/mydoc.dtd">
<weatherReport>
  <date>05/29/2002</date>
  <location>
     <city>Philadelphia</city>, <state>PA</state>
     <country>USA</country>
  </location>
  <temperature-range>
     <high scale="F">84</high>
     <low scale="F">51</low>
  </temperature-range>
</weatherReport>
```

The DTD for this example

```
<!ELEMENT weatherReport (date, location,
                         temperature-range)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT location (city, state, country)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT temperature-range
          ((low, high)|(high, low))>
<!ELEMENT low (#PCDATA)>
<!ELEMENT high (#PCDATA)>
<!ATTLIST low scale (C|F) #REQUIRED>
<!ATTLIST high scale (C|F) #REQUIRED>
```

Operations on XML documents

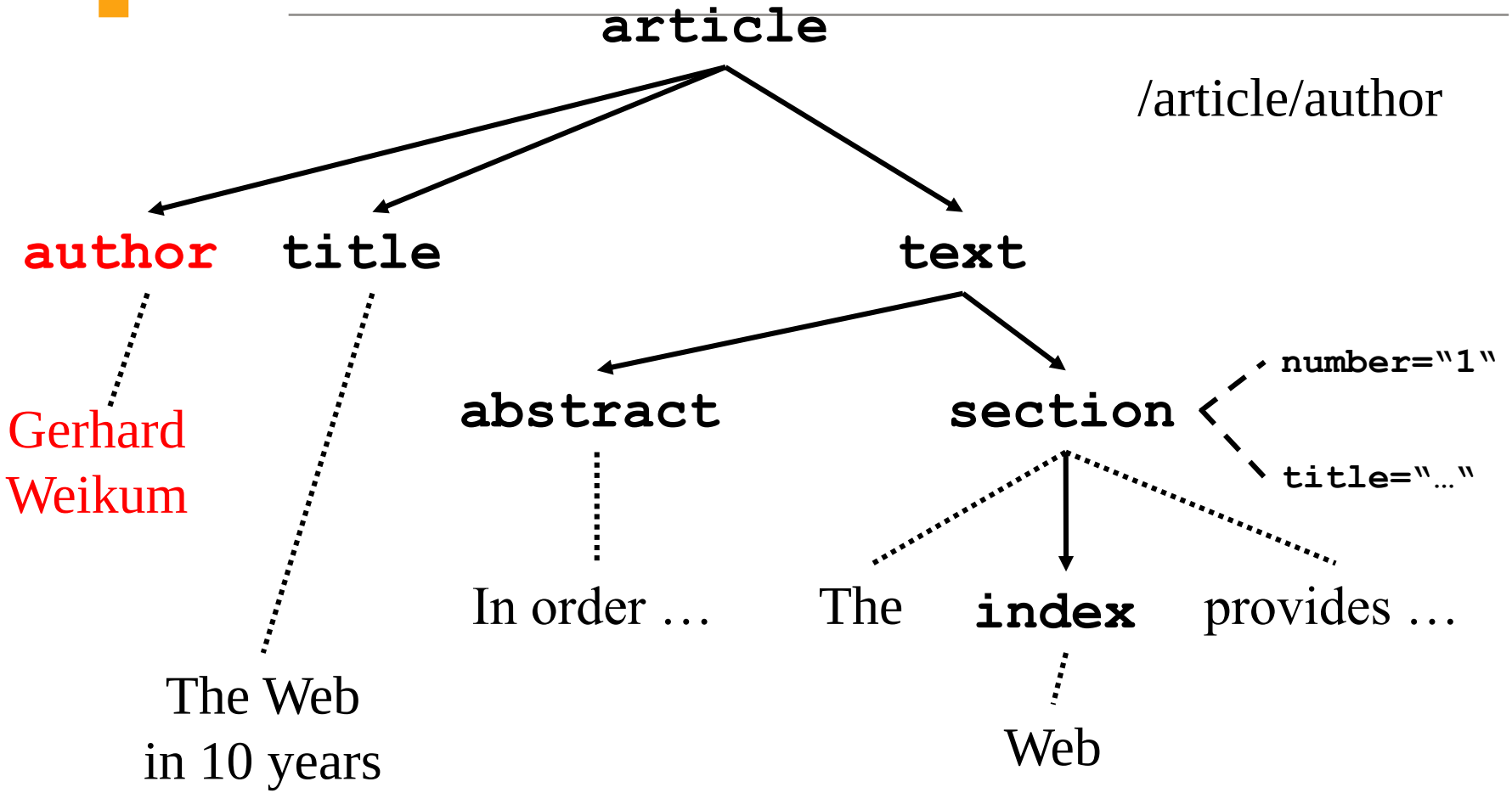
- GetParent
- GetChildren
- GetSibling
- Root-node path
- Query needs the tree traversal

Querying XML with XPath

XPath is query languages for XML data, both
standardized by the W3C and supported by various database products.

A **query result** is a set of qualifying nodes, paths, subtrees,
or a set of XML documents constructed from this raw result.

XPath by Example



XPath

- XPath is a simple language to identify parts of the XML document (for further processing)
- XPath operates on the tree representation of the document
- Result of an XPath expression is a set of elements or attributes

Elements of XPath

- An XPath expression usually is a **location path** that consists of **location steps**, separated by ***/***:

  `/article/text/abstract`: selects all **`abstract`** elements

- Possible location steps:
  - child element **`x`**: select all child elements with name **`x`**
  - Attribute **`@x`**: select all attributes with name **`x`**
  - Wildcards **`*`** (any child), **`@*`** (any attribute)
  - Multiple matches, separated by **`|`**: **`x|y|z`**

Location Steps

- Standard: `/` (context node is the result of the preceding location step)

  `article/text/abstract` (all the abstract nodes of articles)

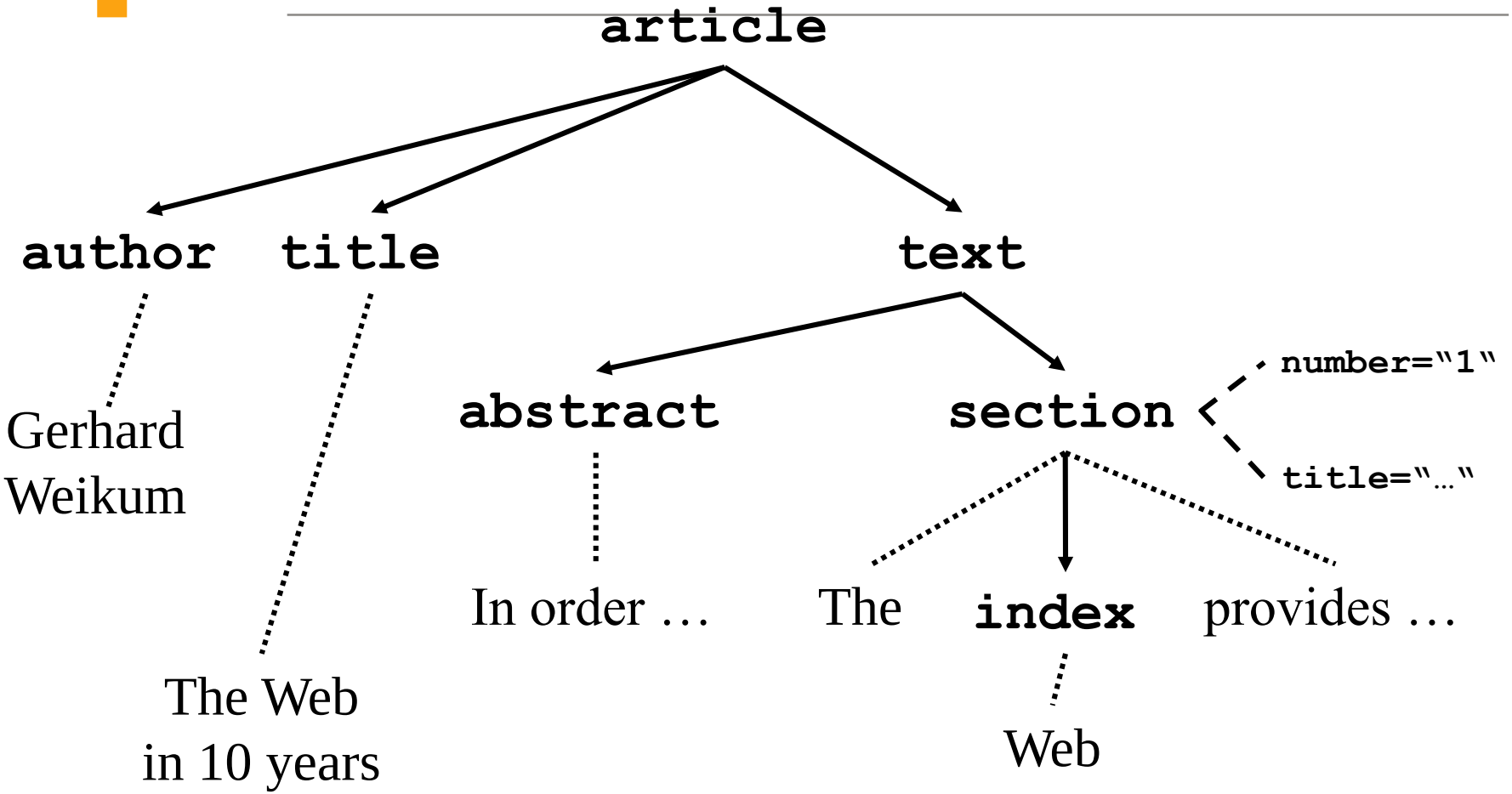
- Select any descendant, not only children: `//`

  `article//index` (any index element in articles)

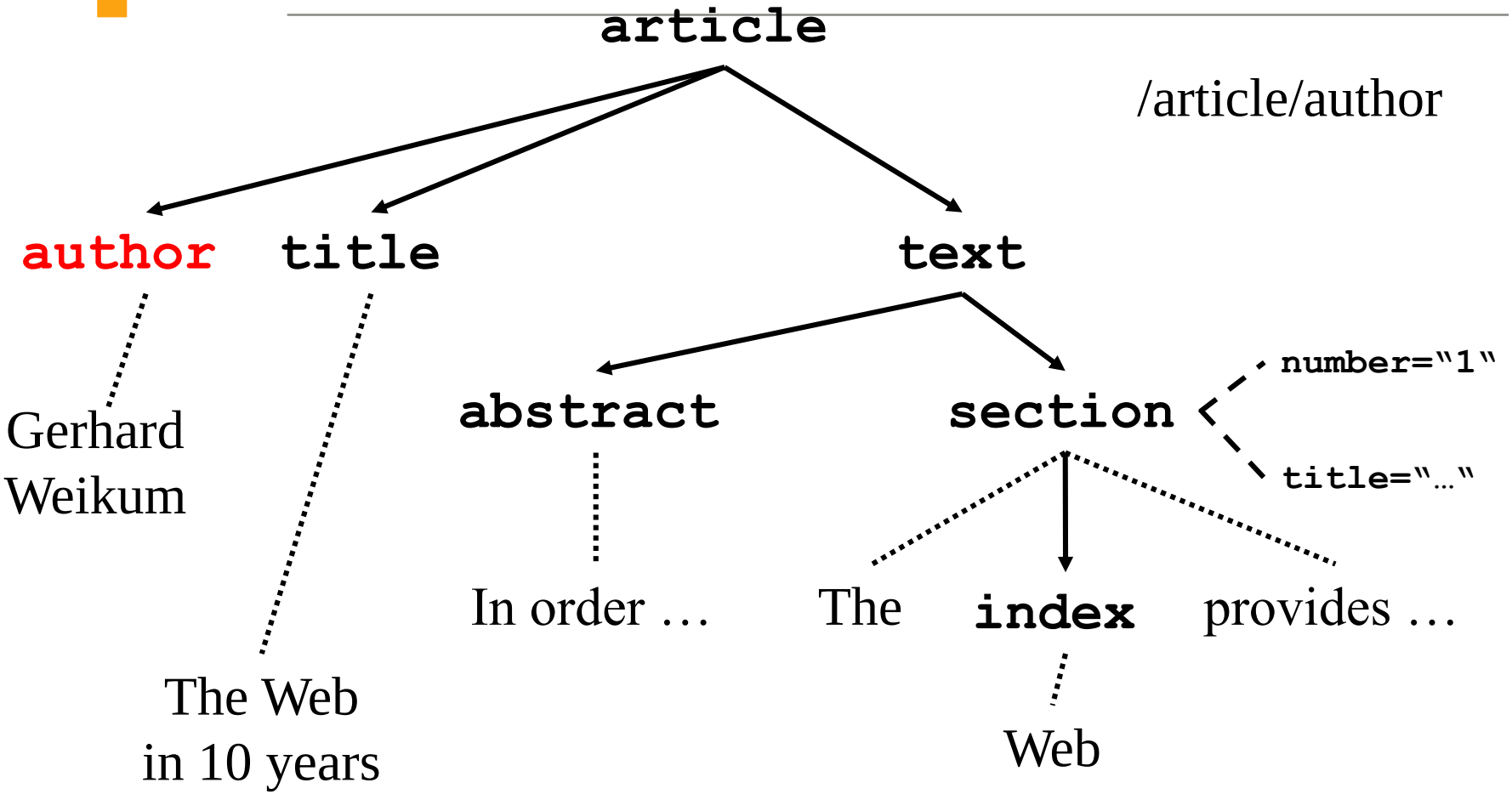
Predicates in Location Steps

- Added with `[]` to the location step
- Used to restricts elements that qualify as result of a location step to those that fulfil the predicate:
  - **`a[b]`** elements **`a`** that have a subelement **`b`**
  - **`a[@d]`** elements **`a`** that have an attribute **`d`**
  - Plus conditions on content/value:
    - **`a[b="c"]`**
    - **`A[@d>7]`**
    - **`<, <=, >=, !=, …`**

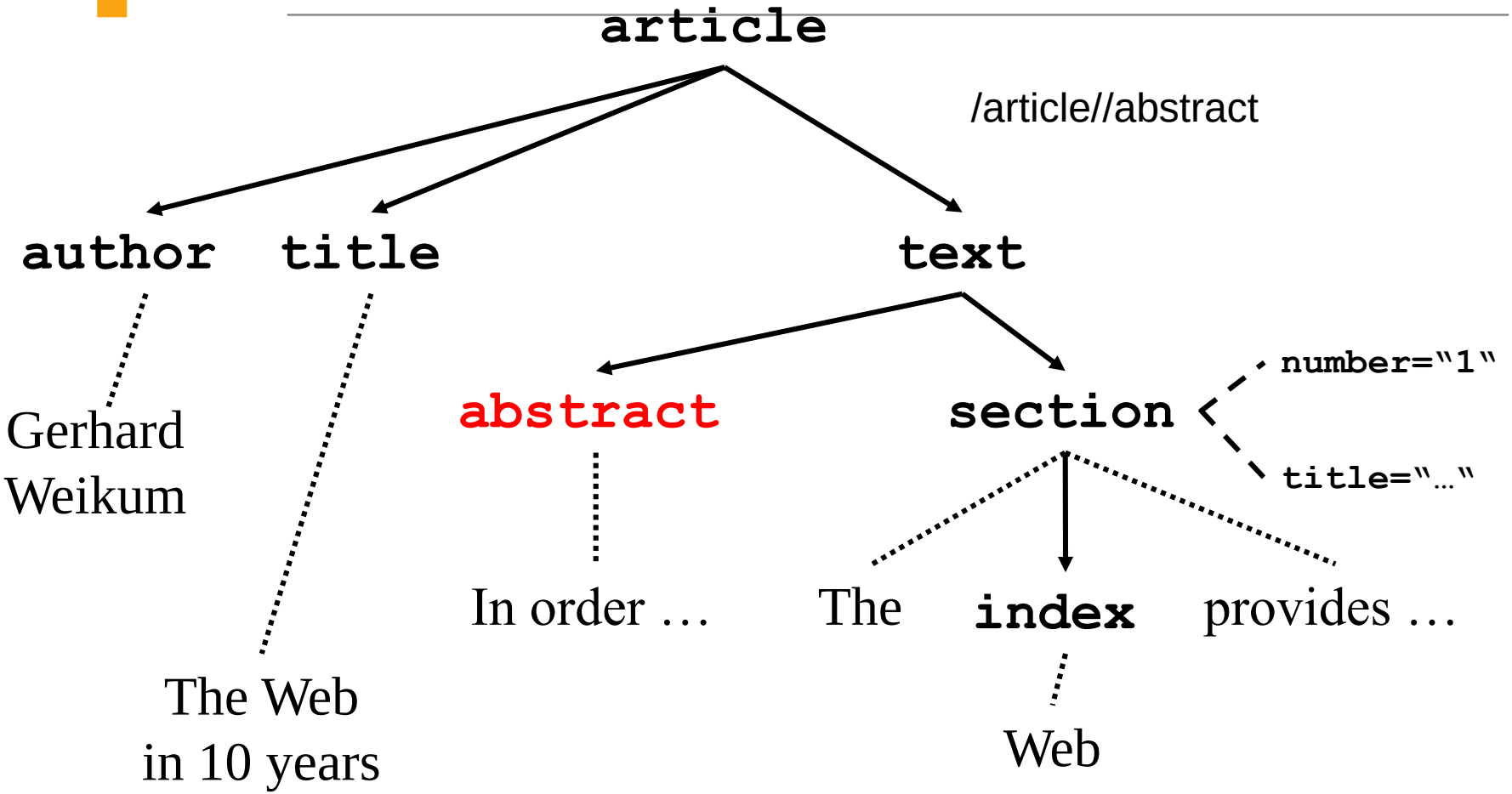
XML Documents as Ordered Trees



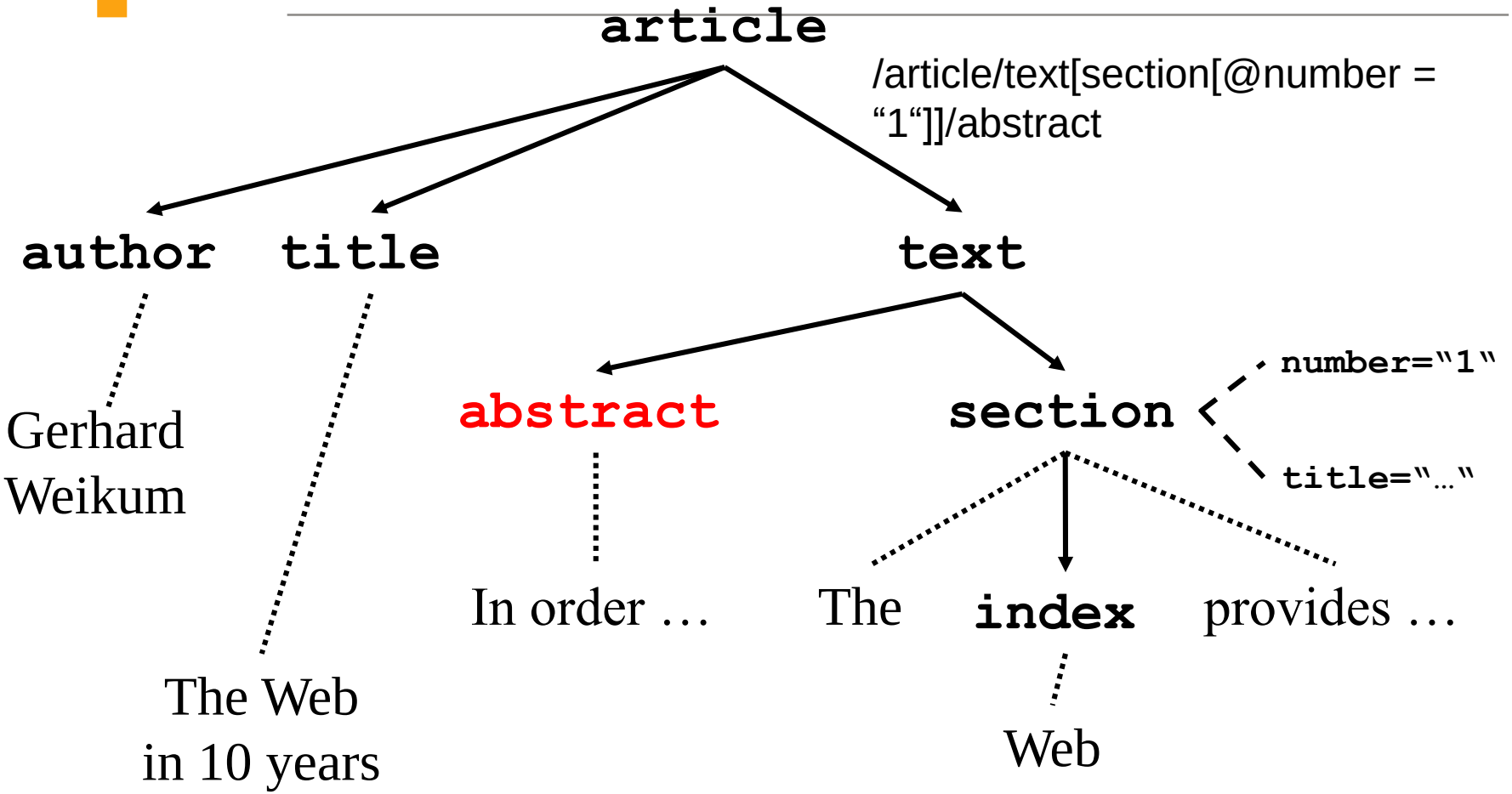
XPath by Example



XPath by Example



XPath by Example



Summary

- Database history from file system to big data system
- Data models has three characters: Structure, Operations and Constraints
- Three common data models: relational model, semi-structured model (XML and JSON) and graph model