Web Data:
Objects, Document, Document Object Model (DOM)

SIV851

Lecture 3

2024

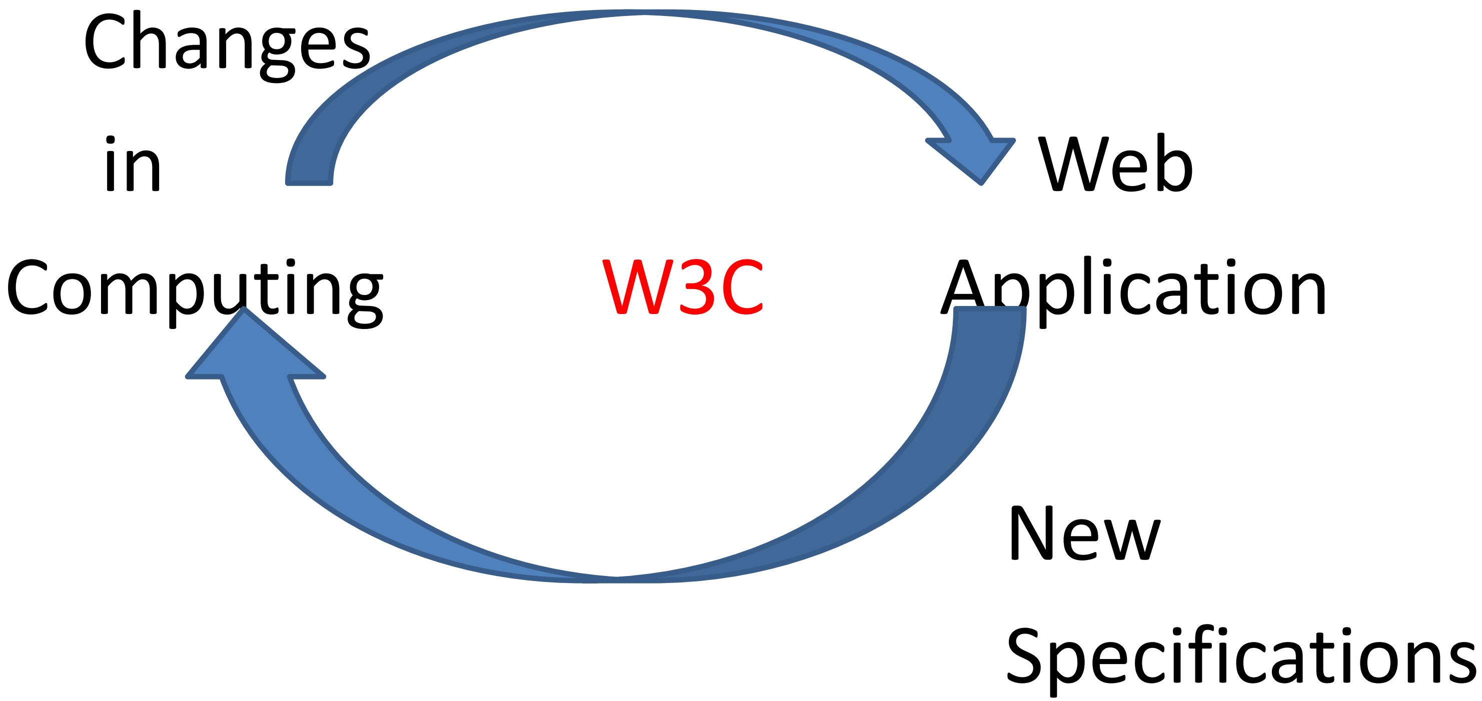
(set, graph, map, archetype)

(relations, XML, KML, ADL)

(list)

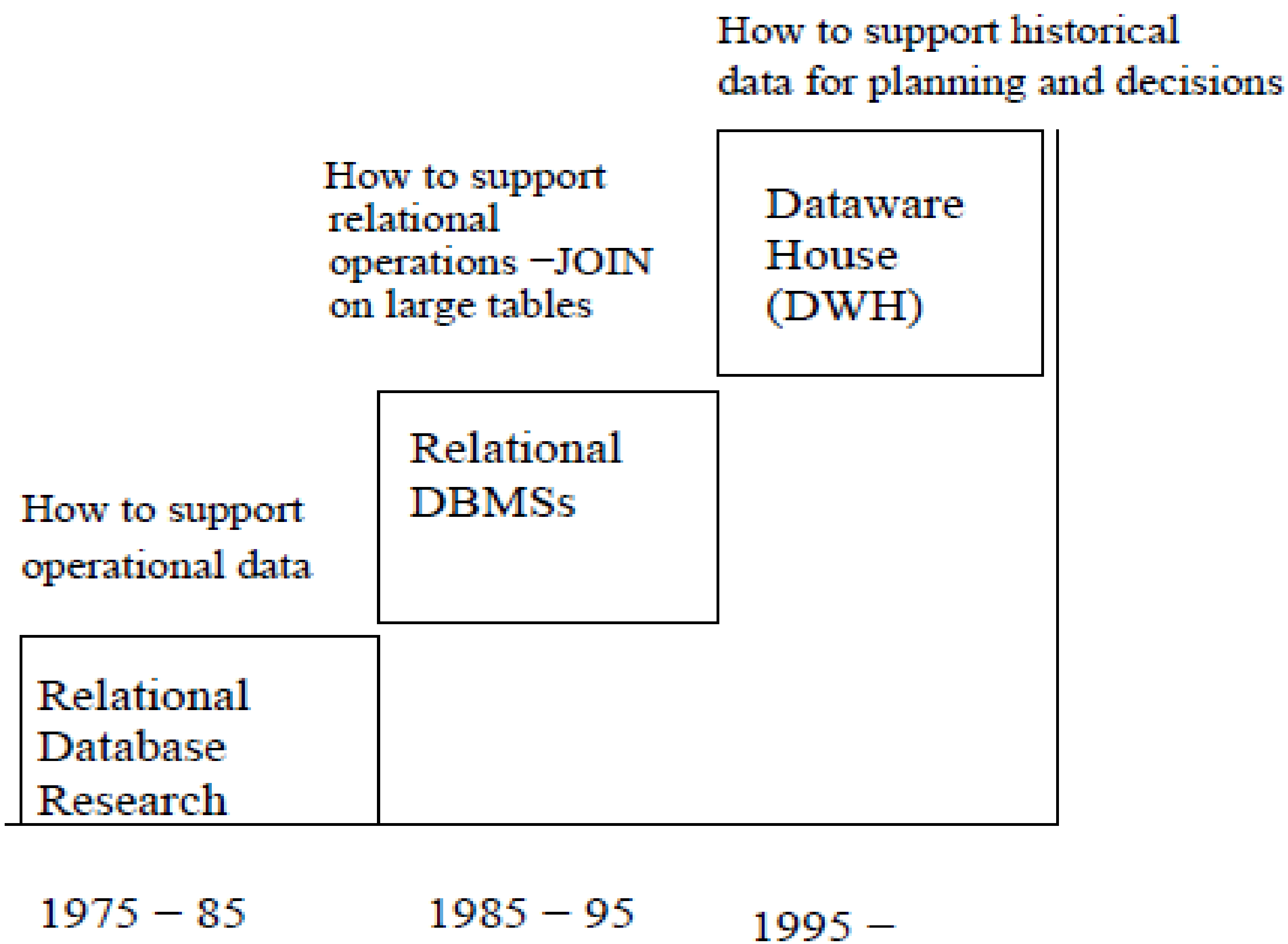
-Subhash Bhalla

Web-based Computing as enterprise (1993 - to date)

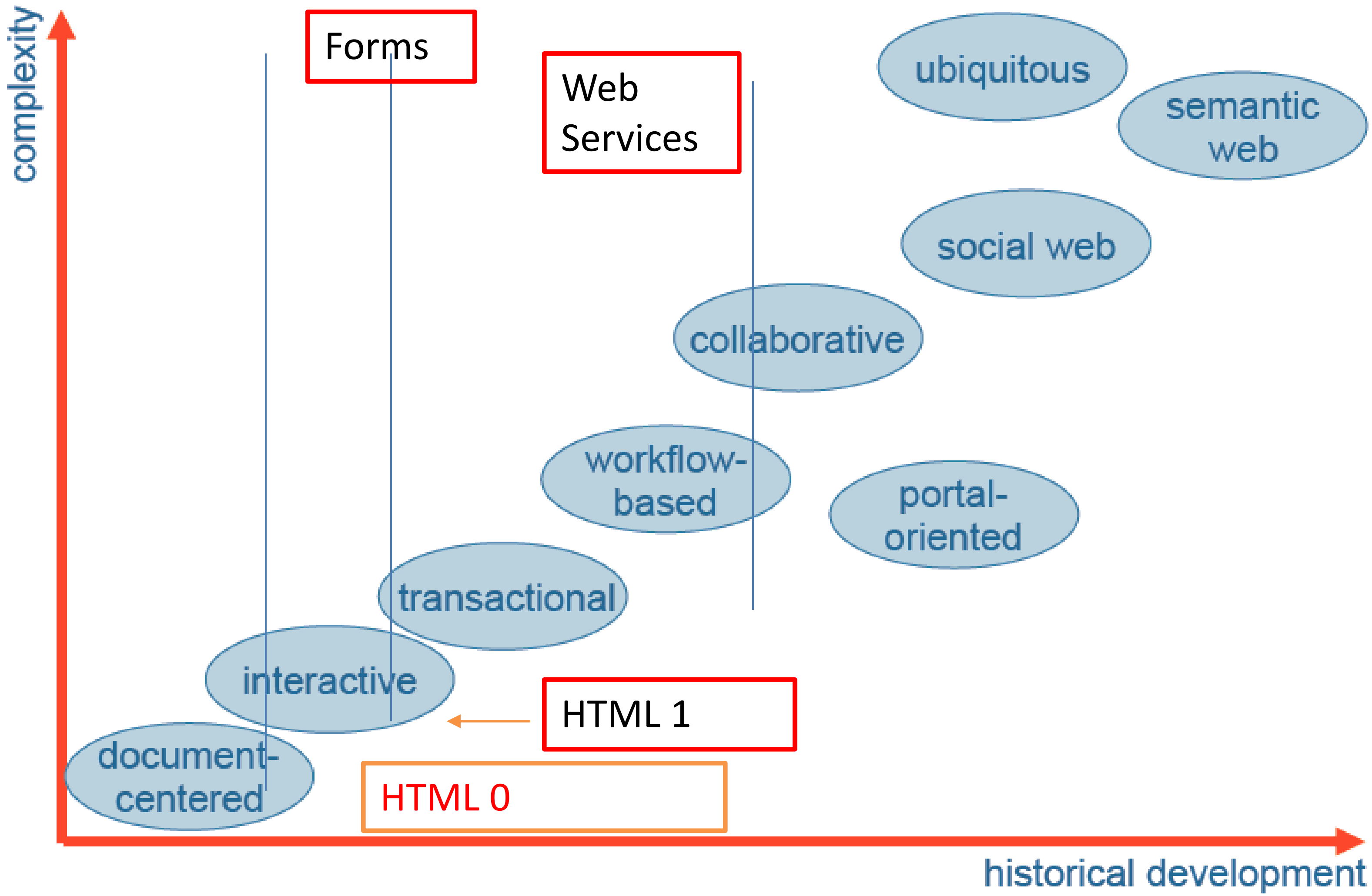


-
-

Development of Data warehouse



Categories of Web Applications



Difference between Database

- Database → Operational Data
- Data warehouse (DWH) → Cummulative data (over space , time and data for other factors)
- Web-based Databases and Web-based DWHs

  Modern TIMES
- (1) Huge Archives (DB, DWH)
- (2) Huge Volume of Information Interchange

  (Network, Distributed Systems)

Recent News

- Vivo secretly sent out Rs 62,476 crore to China to avoid taxes in India

- Read more at: https://www.deccanherald.com/national/vivo-remitted-rs-62476-crore-to-china-to-avoid-getting-taxed-in-india-says-ed-1124632.html

- DRI accuses Oppo India of Rs 4,400 crore tax and duty evasion

- Read more at:
- http://timesofindia.indiatimes.com/articleshow/92861539.cms?utm_source=contentofinterest&utm_medium=text&utm_campaign=cppst

- No Governance ? Omissions OR Commisions ?

- Evolutionary process – too little and too late

E-governance in other countries

- In China, Japan, USA, .. → It is not possible to avoid tax
- In China, a company required to use transactional software approved by Govt. of China
- → A copy of each business transaction is sent to China Govt. servers for record and audit in real-time
- YAHOO Japan news in July 2022 (after vivo and oppo):

In India, most cases, authority notice has legal flaws (governance is weak, e-governance ?

- Hence (courts, lawyers, reply quoting a rule, etc…)
- Bank frauds → Chowksi, Nirav Modi, ..ICICI Bank
- ED and CBI cases " conviction rates are poor"
- Governance →e-governance (records and monitoring) is missing ?

Governance

- 100s of activities as tasks and subtasks
- + Aadhar
- +PAN card
- +Election cards
- +employment in rural area

- Large scale

- Concern population

- Manual procedures
- Simple adoption of IT and
- computerization

- Tasks → unmanageable without IT or e-governance

Data Exchange Environment
Information Interchange

- LAN + Internet ;   on the Web
- ------------------------------
- RPC ; Web Download; Web Services
- ------------------------------
- C-2-B;   B-2-B;   P-2-P
- Cloud ←→ Client
- Cloud ←→ Peer
- Cloud ←→ Cloud

History: Programs and Data Exchanges

- [Stage 1]   Program ←→ Data

  → Direct access to the data/medium

  → (format – csv, space, Columns – data types, …variables (hardwired to data)

  Program [structured data – JAVA/C++ objects]

[Stage 2]   Database – 3 level Data Dictionary

- Structured Data (DBMS (scheme): db (data))
  - sets, relations, list, bags

[Stage 3] Web Data– Semi-structured Data (latex, HTML, …)

- Objects, object-class, sets, ….

History: Programs and Data Exchanges

- Stage 1 → Programs have built in knowledge of data (size and form), programming includes I/O programming

- Stage 2 → Programs deal with higher level contents (data as set), programming is easier; high level programs, I/O programming is at outside end.

- Stage 3 → Programs use semi-structured content as object and documents; Programming is easier by handling O-O content; example- Live Objects

Files and Documents

- Raw data in Files:
- Edit files with text
- Vi, emacs, vim,

  → create latex, html,..

- Web data as objects and Documents:
- Update documents with Objects
- MS word,
- ? WYSIWYG (What-you-see-is-what-you-get)
- WYSIWYG HTML editor

Key Question about Exchacnges

- Some data → Raw form ?
- (a) handling on both sides
- (b) Context ; Interpretation ; dynamic

- Some data → Object form

- Some data → Documents form

- Web Browzer : Uses a Document Object Model

Web Documents – Objects in a DOM (Document Object Model)

- Browzer → Display objects

- Browzer:
- Server-Clients → exchange objects

- Facebook, Gmail, google Docs, Google Maps, …

- OBJECTS are packaged with tags in HTML, XML, …

Documents, objects, DOM

- Example: Document → Book
- Objects → Title, title page, preface, index, chapter 1, ..
- Document Object Model (DOM) → FIXED
- Consider Compilers, interpreters, →

Program/instruction→ [Compile] →need a DOM

Consider a Browzer ? It uses a DOM

Input file (html) → [Browzer] → Object display

Input file (xml) → [Browzer] → Object exchange

Web Data: Program + Data; Database Application Design and Development (web client and Server programming)

- Application Programs and User Interfaces
- Tree Structured Data

  (a) Clint-side – XHTML, CSS, scripting with JavaScript

  (b) Storage side –

  RDBMS, XML, JSON, scripting with PHP, JSP…
- Web Systems – Client + web server + Database Server
- Servlets and JSP, PHP
- Application Architectures
- Rapid Application Development using web services
- Application Performance
- Application Security
- Encryption

Document Object Model

- Documents: Book Chapter, Report, Book, ..
- Example: Book: Title, Index, Chapter heading

- Web → Documents → have Object Model
- Display objects (data standard): HTML
- Data objects (data standards): XML, JSON, RDF, ….

- Web: DOM (Document Object Model)
- Programming: a) Client-side – HTML, XML, XHTML,
  CSS, JavaScript, ..
  b) Server-side- PHP, Java/JSP, Javascript,...

Web Data: Tree Structured Data

= Data in Tree form

XML → root (tree)

→{tree} ; level 1 [ ]; level 2 (); ....

→ {..[( ) ()] [()] ..} Proper Nesting

→A)rooted Tree

→B) Unique path from root

→C) No cycle (tree has no cycle)

Contents in XML: Object Model

- 1. Data (example) → Web Programming

- 2. xml → Element; Sub-element; Attribute

<name-of-book> Web Programming </name-of-book>

- Tag → <name-of-book > .. </name-of-book>

- (Opening) Tag can include an attribute →

<name-of-book Type=“text-book” >

Organization of contents in XML

<bookstore>

- <book Category = “basic” >

  <title lang = “Italian” > Everday Italian</title>

  <author> Giada De Laurentiis </author>

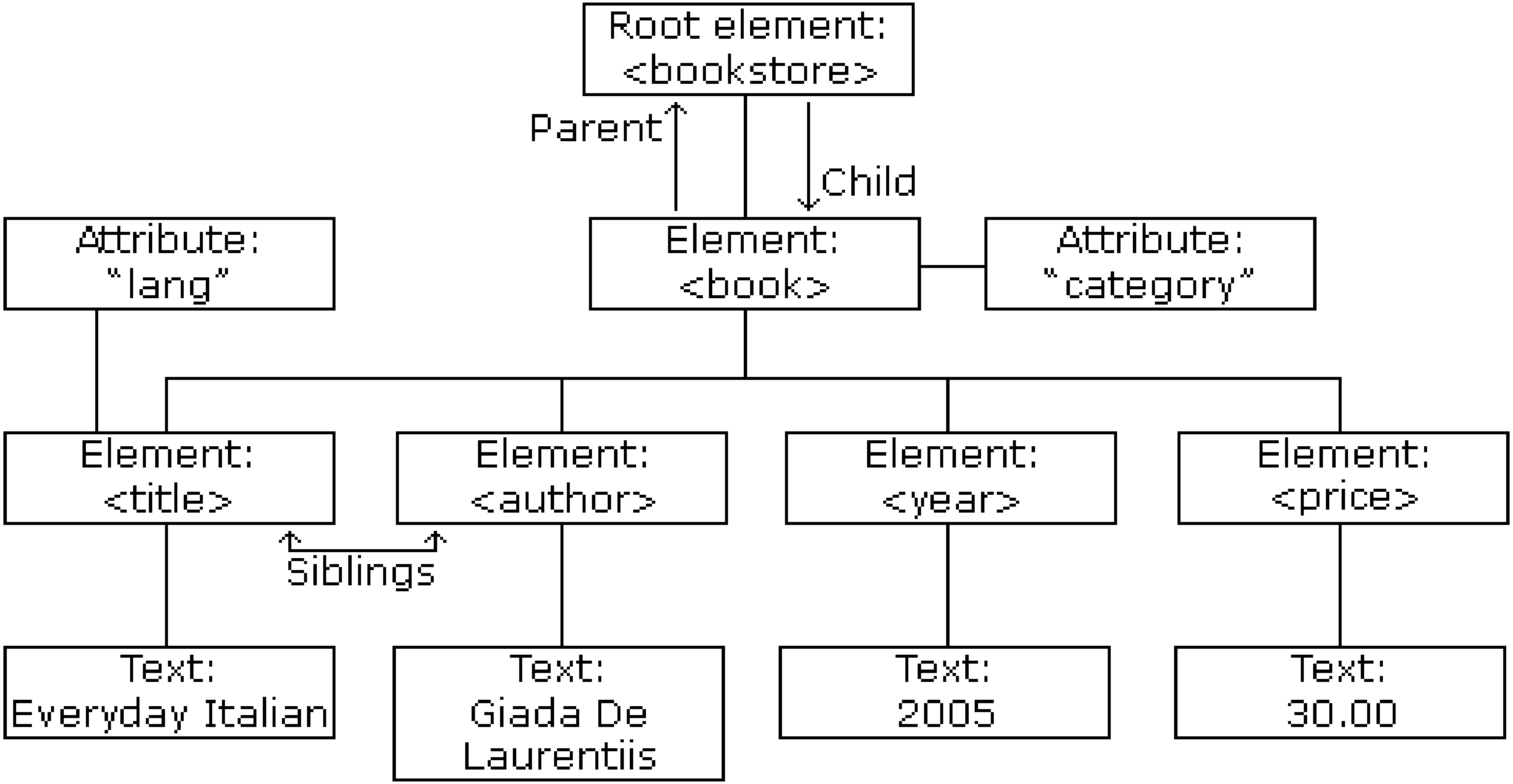
  <year> 2005 </year>

  <price> 3000 </price>

- </book>
- <book> .. .. </book>
- . . . .. </bookstore>

Document Object Model (DOM)
https://www.w3schools.com/xml/dom_intro.asp

Graph: rooted, unique path from root to leaf, acyclic



Data Interchange

- [Stage 1] Program 1 → CSV (comma Separated values)
- Program 2 ← CSV values
- --
- [Stage 2] Program data dump
- Stacks
- ORACLE Database Dump
- Arrays
- Abstract Data Types

- PROGRAMMING → to upload and process
- → knowledge of syntax and semantics

- [Stage 3] Web Data (data sharing among multiple applications) → YAML, Jason, XML, Candle

Web Data: Information Interchange [Stage 3]

- Information System 1 → Amazon Java books
- Information Systems 2 ← Amazon Books
- -------------------------------
- 1. Objects – Books, Rooms with id and (x,y) coordinates, Students, Courses, ….
- 2. Documents – web documents, financial statements of companies, …
- 3. Complex Graphs and structures – Protiens, Maps, …
- Information → Sets (RDB) , relation ! DB !
- →Tree-structured Data (XML), …

- → Syntax and Semantics
- Tree Structured Data → XML, JSON, Candle Markup

Realtional Model Data Elements: How does it work

- 1. Set - No duplicates and no order
  [(3,1,1)- not a set; Set (3,1) is same as set (1,3)]

- 2. Bag – data has no order
  [(3,1,1) is same as (1,3,1)]

- 3. List – has order [(3,1,1) is not same as (1,3,1)]

Content has no form- an island…

1. → Set

Set = Relation;

2. Stored over List

3. List → Processed by **Von Neumann architecture / Turing M/C**

Role of ADTs

- High level Operations
- Storage Structures
- Convenience of processing

Role of Abstract Data Type (ADT)

- Abstract data type (ADT) → a mathematical model

  (a certain class of data structures that have similar behavior);

→ Used for certain data types

(in one or more programming languages that have similar semantics)

→ ADT → is defined indirectly (by the operations that may be performed on it and by mathematical constraints on the effects (and possibly cost) of those operations)

- [Example] Stack → (as an abstraction)

  defined by three operations: push, pop, and peek

→When analyzing the efficiency of algorithms that use stacks, one may also specify that all operations take the same time no matter how many items have been pushed into the stack, and that the stack uses a constant amount of storage for each element.

ADT

Abstract data types are → theoretical entities,

1. → used to simplify the description of abstract algorithms,
2. → to classify and evaluate data structures,
3. → to formally describe the type systems of programming languages.
4. → ADT may be implemented by specific data types or data structures, in many ways and in many programming languages;
5. → or described in a formal specification language.
6. → ADTs are often implemented as modules: the module's interface declares procedures that correspond to the ADT operations, sometimes with comments that describe the constraints.
7. → This information hiding strategy allows the implementation of the module to be changed without disturbing the client programs.

Content: Table – Set/bag (represent as?) DB →web data

company	section	employee
c1	s1	e1
c1	s1	e2
c1	s2	e3

- <company id="c1">
- <section id="s1">
- <employee id="e1"/>
- <employee id="e2"/>
- </section>
- <section id="s2">
- <employee id="e3"/>
- </section>
- </company>

Data + form (on web) (data + DTD)

```
<sectionList>
  <section id="s1">
    <company id="c1"/>
      <employee id="e1"/>
      <employee id="e2"/>
  </section>
  <section id="s2">
    <company id="c1"/>
      <employee id="e3"/>
  </section>
</sectionList>
```

```
<employeeList>
  <employee id="e1">
    <company id="c1"/>
      <section id="s1"/>
  </employee>
  <employee id="e2">
    <company id="c1"/>
      <section id="s1"/>
  </employee>
  <employee id="e3">
    <company id="c1"/>
      <section id="s2"/>
  </employee>
</employeeList>
```

Relational Model (set)- EF Codd 1971 (IBM)

- A. Two levels-

1. User → Sets and set operations

----

2. Storage → list ;

- -User [need elements]; (no navigation)

  -Storage [store over list; provide thru index or list search]

- User [need set operations]

  Storage management← system

Table form → set (product set)

company	section	employee
c1	s1	e1
c1	s1	e2
c1	s2	e3

- Table form of data → set, or bag
- Operations → set operations
  →Query language

Acess methods

- Old Models- Hierarchical Model

  → variation over tree structure

  → Started from Bottom: Query on list

Network Model → linked list

Relational Model: Top Down Approach →

Set + Set operations : Two layers

→ No navigation as in tree or list

→ influence over query operations

Database Query Language - version Progress

- 1. SQL

a) 1971- 1976

b) SQL 2 (1992)

c) SQL 3 (Object – Relational Data Models) (1999)

d) SQL 4 (Web Data – XML) (2003)

e) Web Services (Data Resouce sharing)(2005 – 2010)

f) Semantic Web – Using web (2005 -2020)

[On web New] Database Form of contents

- 1. Web Documents

- 2. Map → Google Map, Yahoo map, MS map

- 3. Bio-medical informatics → web data resources (complex chains of molecules in protiens)
- 4. Electronic Health Records

[XML ?] DB Forms of content

- Content → has a form (structure)
  (not islands of data)
- Representation → 1. list (too simple)
  2. set
  3. graph

Low level (Disk/Memory) ← list

Processing → Content; intermediate representation (may be); storage (list)

[New] DB Forms of Content

- Web Document → XML
- Web-based Maps → KML (google)
- Bio-Medical Data Resources → XML, or similar to XML
- Electronic Health Records → ADL (similarities with XML, used in conjunction with XML)

- 1. Document form → graph (not set)

Content → not Island → Graph Data

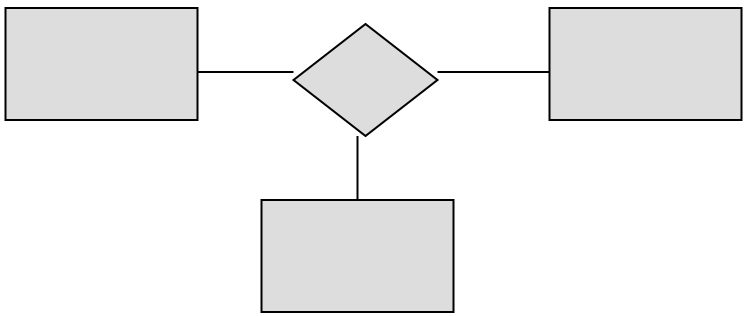
A graph G = (V,E) is a collection of nodes (vertices) and edges.

A graph → "relationship structure" among different data elements.

A *graph database* is a collection of different graphs representing different relationship structures.

Notes:
a) Storage Level → list structures, b) multiple levels, c) intermediate forms (XML → Lists)

Compare: Graph database and (set) Relational database



A relational database maintains *different instances* of the *same relationship structure* (represented by its ER schema)

A graph database maintains *different* relationship structures

← Web Documents, maps, Bio-Medical Informatics, Electronic Health Records

← Store in intermediate forms– XML,KML,ADL

Queries over New DB Contents

- Attribute Queries (Type A)
  - Queries over attributes and values in nodes and edges. (**Equivalent to a relational query within a given schema**

- Structural Queries (type B)

← [Not Main focus of our Discussion]
  - Queries over the relationship structure itself. Examples: Structural similarity, substructure, template matching, etc.

Graph Database Applications- (Type A and Type B)

- Software Engineering
  - UML diagrams, flowcharts, state machines, …
- Knowledge Management
  - Ontologies, Semantic nets, …
- Bioinformatics
  - Molecular structures, bio-pathways, …
- CAD
  - Electrical circuits, IC designs, …
- Cartography, XML Bases, HTML Webs, …

Structural Queries on Graph Data (Type B)

- Undirected Graphs
  - Structural similarity, substructure
- Directed Graphs
  - Structural similarity, substructure, reachability
- Weighted Graphs
  - Shortest paths, “best” matching substructure
- Labeled Graphs
  - Labeled structural similarity, unlabeled structural similarity

Structural Queries (Type B)

- Substructure query
  - Given a graph database $G = \{G_1, G_2, \ldots G_n\}$ and a query graph Q, return all graphs G_i where Q is a subgraph of G_i.

- Structural similarity
  - Given a graph database $G = \{G_1, G_2, \ldots G_n\}$ and a query graph Q and a threshold t, return all graphs G_i where the *edit distance* between Q and G_i is at most t.
  - The edit distance between two graphs is the number of edge modifications (additions, deletions) required to rewrite one graph into the other

Structural Queries

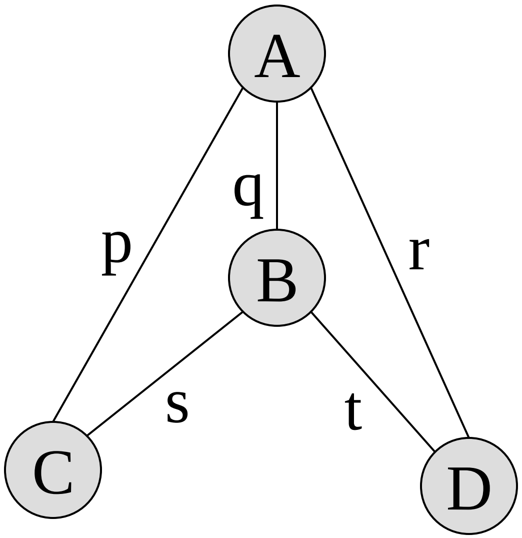
- In graph databases structure matching has to be performed against a *set* of graphs!

- Method of storage, pre-processing and index structures → crucial

(if structural searches are to be practical)

Storing Graph Data → set

Attributed Relational Graphs (ARGs)



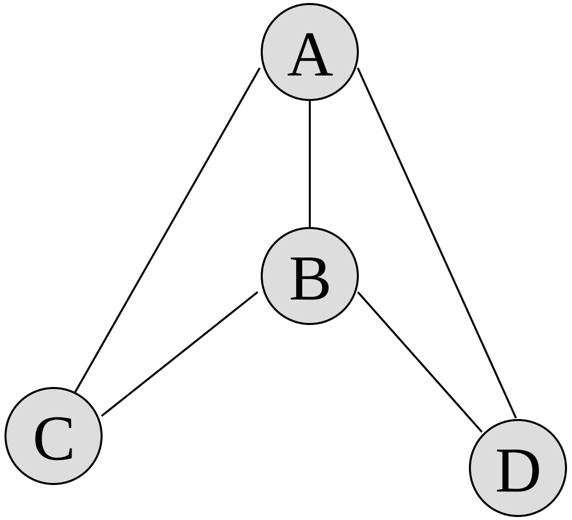
A	B	q
B	C	s
B	D	t
A	C	p
A	D	r

Storing Graph Data

- ARGs
  - ARGs store a graph as a set of rows, each depicting an edge
  - Amenable to storage in an RDBMS and easy attribute searches using SQL
  - New Query Languages (←Research Type A)
  - Costly structural searches, requiring complex nesting of SELECT statements
  - Each graph needs a separate table
  - Type B (VLDB , SIGMOD, many forums)

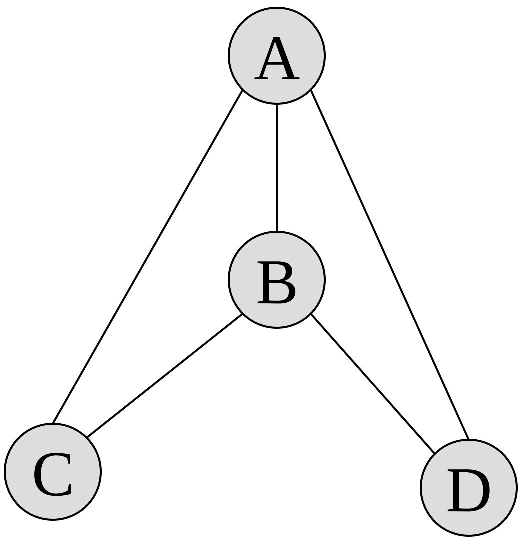
1. Storing Graph Data in XML (rooted tree,acyclic, unique path from root)

- <node id=“A”>
-         <node id=“B”>
-                 <node id=“C”>
-                 </node>
-                 <node id=“D”>
-                 </node>
-         </node>
-         <node id=“C”>
-         </node>
-         <node id=“D”>
-         </node>
- </node>



2. Storing Graph Data in XML (arbitrary graph)

XML with IDs and IDREFS:



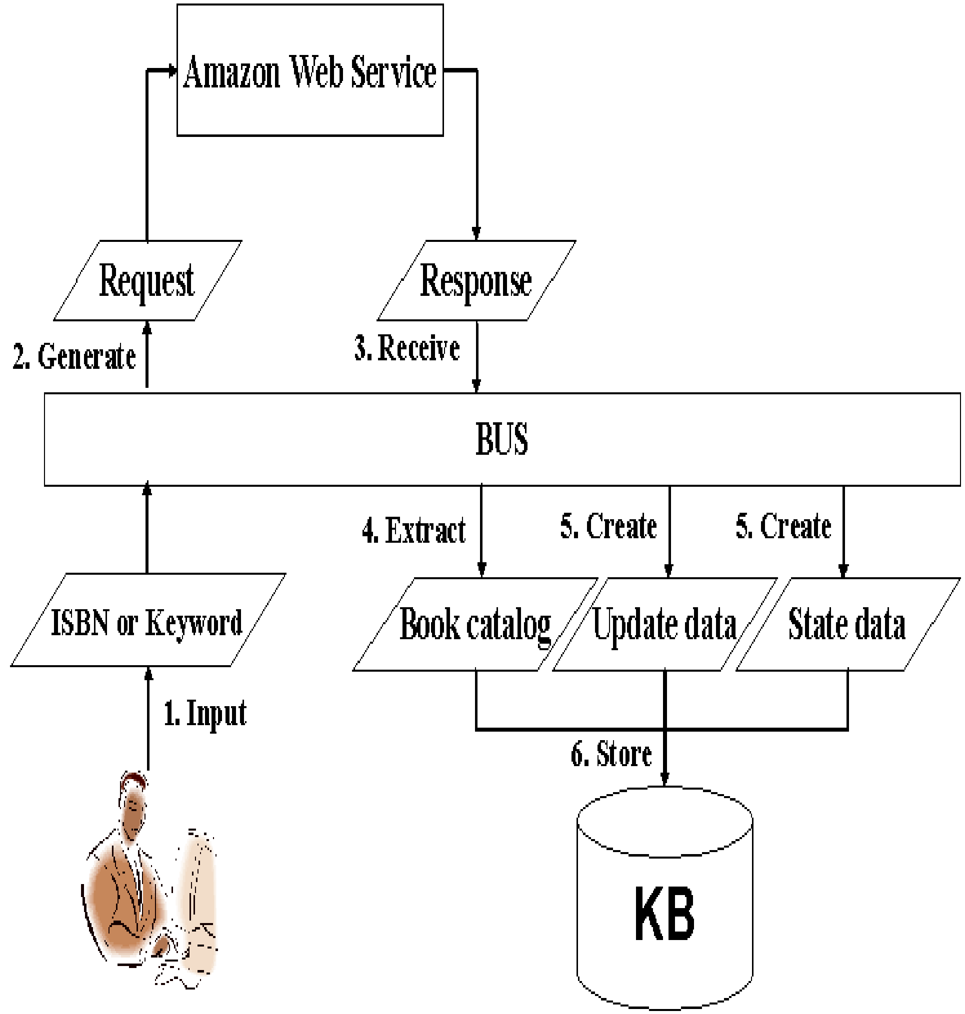

```
<node id="A", adj="C D">
        <node id="B">
                <node id="C">
                </node>
                <node id="D">
                </node>
        </node>
</node>
```

Storing Graph Data

- XML (with or without IDREFS)
  - Reduces graph database to an XML base
  - Use XPath / XQuery engines for attribute querries and structural queries
  - Widely supported by a variety of XML parsers
  - Costly structure/sub-structure matching
  - Needs distinction between IDREF edges and hierarchy edges

Example
Usage Application - 1. Web Documents

1. Input ISBNs or Keywords (of author or title).
2. Send request data to Amazon Web Service.
3. Receive response.
4. Extract Documents from the response.
5. Add update data and state data to the book catalog.
6. Store these data into KB.



KB: Data Structure

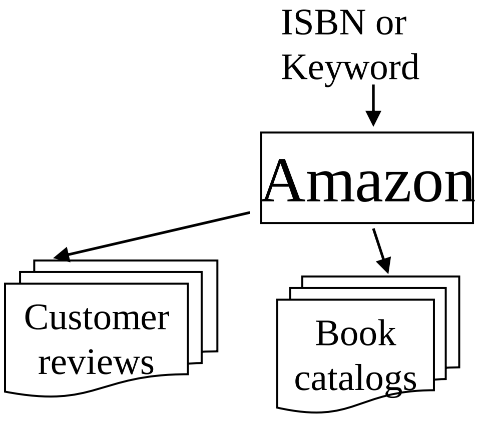
Book

- URL of image: text
- - ASIN: number (※1)
- - Title: text
- - Average rating: number (※2)
- - Author name: text
- - URL of detail page: text
- - Price: text
- - Publisher name: text
- - Publication date: text
- - Number of pages: text

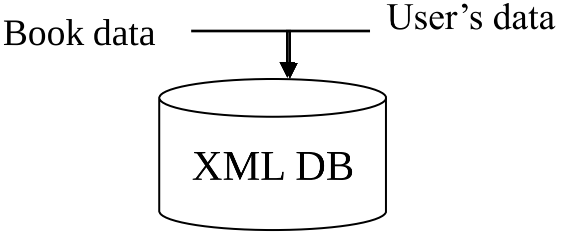
- Sales rank: number
- .. . .

Web Documents

- Web Services：Web API
  1. Amazon E-Commerce Service
  2. Yahoo! Search Web Service
  3. Google AJAX Search API
  4. Technorati Search API



- XML DB:DBMS for XML
  1. Knowledge Base (KB)
     - A collection of book data for BUS.
  2. Information Repository of User's Needs (IRUN)
     - A collection of data that consists of user's interest and needs.



Amazon E-Commerce Service

- Product information (e.g. catalogs, reviews, rating) retrieval for:

1. **Books**
2. Music
3. DVD
4. Electronics
5. Kitchen
6. Software
7. Video Games
8. Toys

Yahoo! Search Web Service

- Web information (e.g. URL, content or hit count) retrieval:

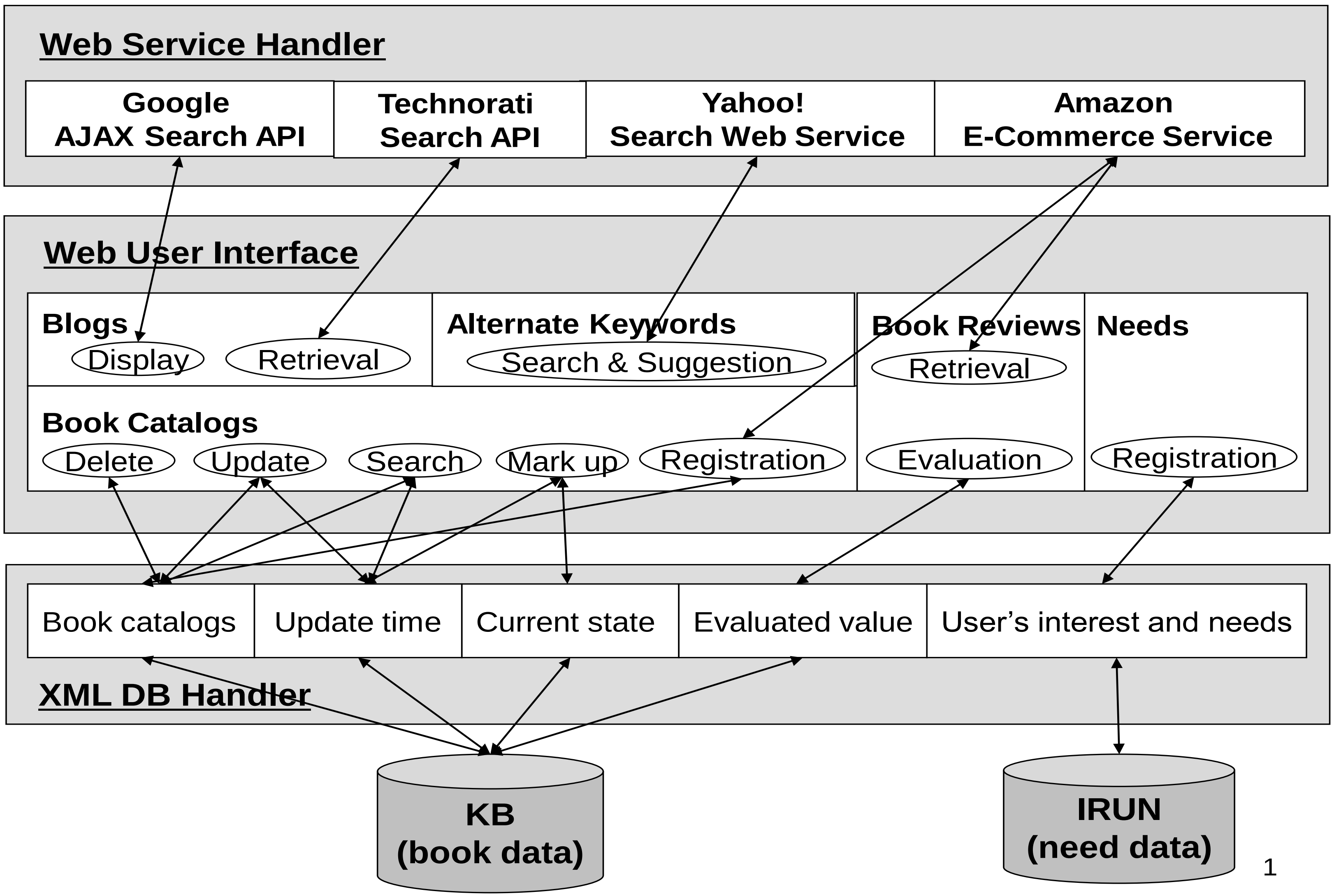
1. **Web pages**
2. Images
3. Movies

Google AJAX Search API

- Embed search box in a web page and display search results of:

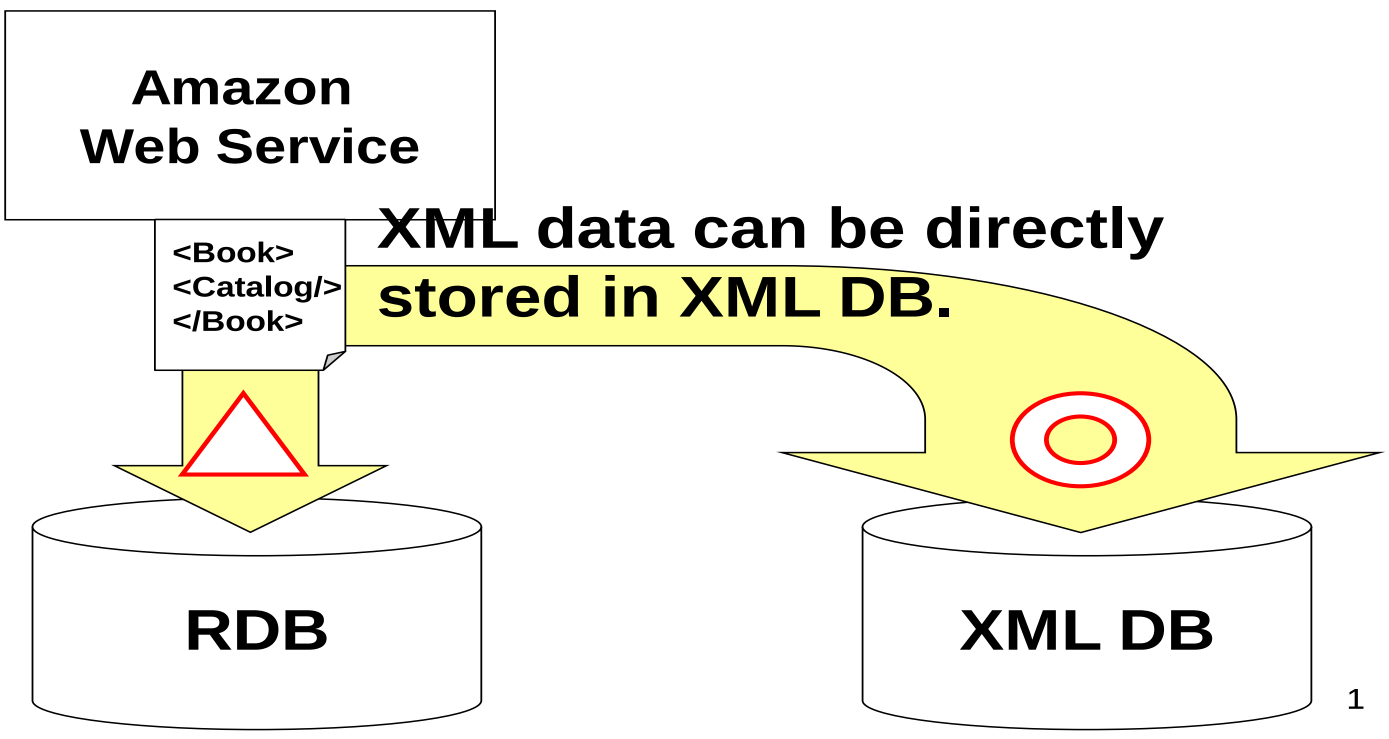
1. Web pages
2. News
3. Video
4. Maps
5. **Blogs**

Book Utilization System



A). Direct Storage of XML

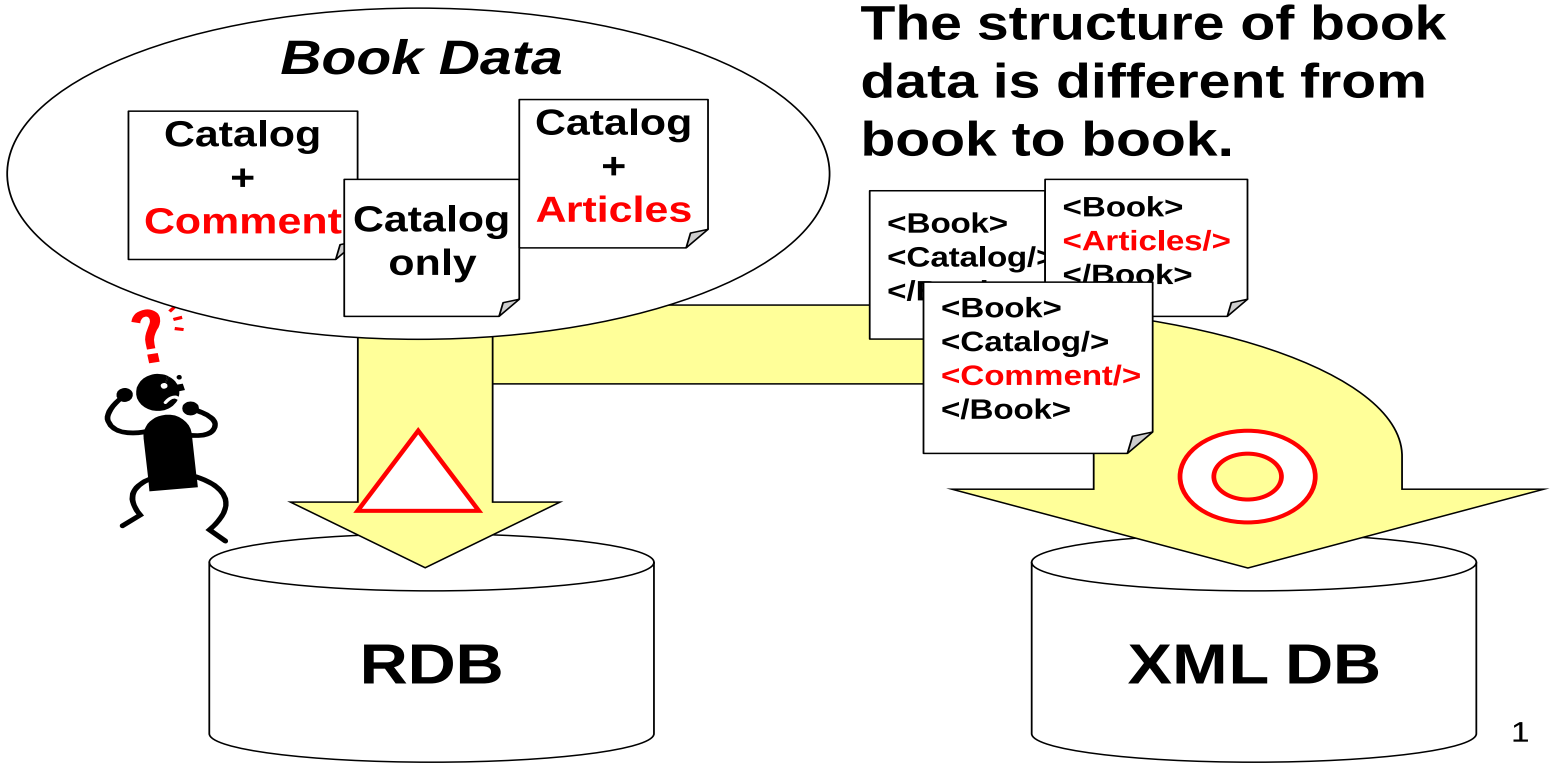
A). Direct Storage of XML



Semi-structured Data Handling

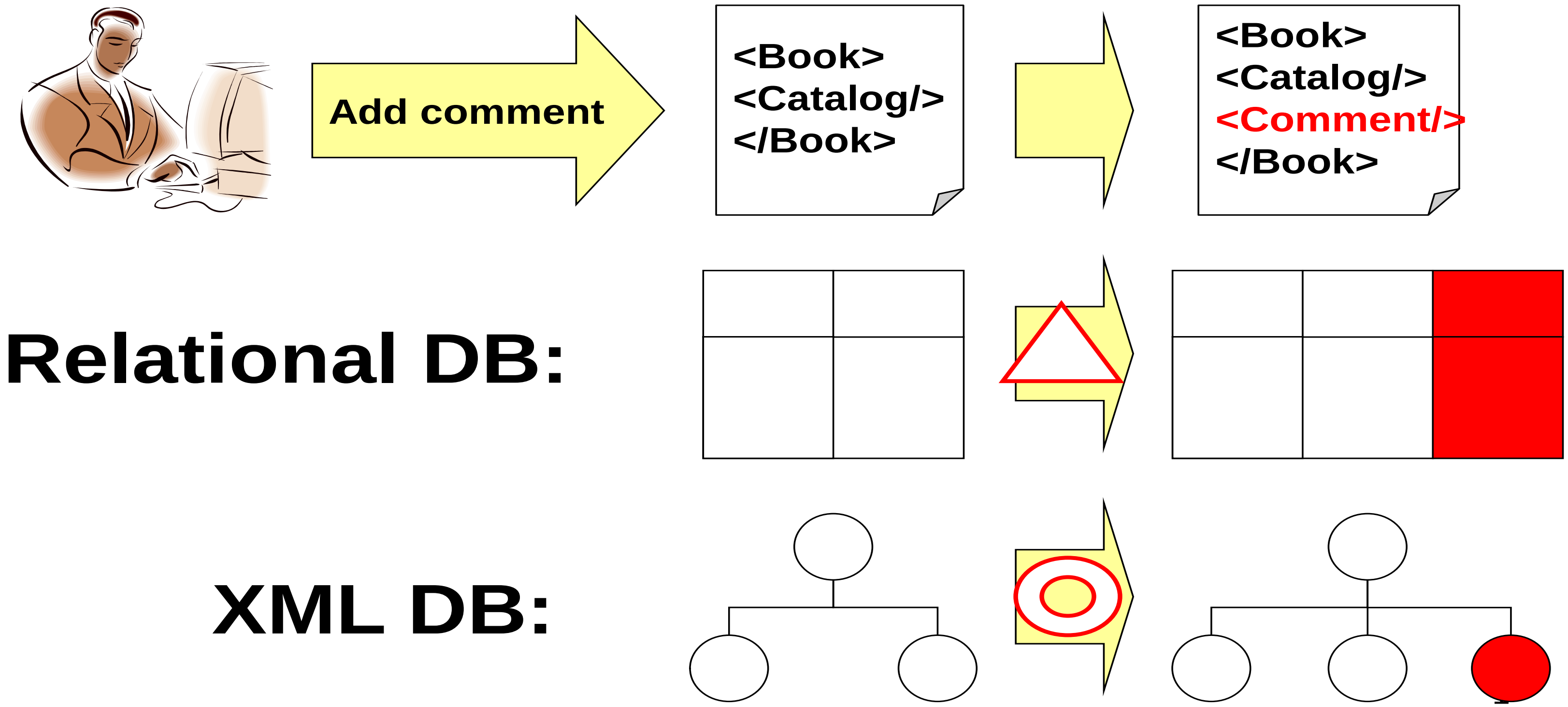
B). Semi-structured Data Handling

The structure of book data is different from book to book.



Web Document

C). Frequent Structural Change



Content – 1. Web Document

```
- <BookShelf>
  - <Book>
    + <Image>
      <ASIN>0201702525</ASIN>
    + <Title>
      <AverageRating>4.5</AverageRating>
    + <Author>
    + <DetailPageURL>
      <Price>$44.99</Price>
    + <Publisher>
      <PublicationDate>2001-11-24</PublicationDate>
      <NumberOfPages>504</NumberOfPages>
    - <Update>
        <Latest>20071224185048</Latest>
        <Added>20071214182858</Added>
        <Commented>20071224184913</Commented>
        <Recommended>20071224185048</Recommended>
        <Searched>20071214183018</Searched>
      </Update>
      <SalesRank>459835</SalesRank>
      <State>Recommended</State>
    - <Extended>
      - <Cooment>
        - <![CDATA[
              Good book for software developer

          ]]>
        </Cooment>
      </Extended>
    </Book>
  </BookShelf>
```

Update information:
- Added time
- Commented time
- Recommended time
- Searched time

Current state of a book

Comment added by a user.

Semi-structured Data

- Web ⟷ Data
- Information interchange, exchange → document structure
- Semi-structured Data
- { name: ”Alan”, tel: 2157786,
  email: ”abc@wwexch.net” }

Web Data Forms and Labels

- Duplicate labels

  { name: ”Alan”, tel: 2157786, tel: 3782535 }

- Many labels or missing labels

{ person:
{name: ”Alan”,tel:2157786, email: bc@wwexch.net”},
person:
{name: {first: ”Sara”, last:”Green”},
tel: 2136877, email: ”sara@the.xyz.edu”},
person:
{name: ”Fred”, tel: 4257783, Height: 183 }
}

A relation and its XML form

Fruits-table = fruit-name, string(6), color, string(5)

[Apple, Green]

[Apple, Red]

- <?XML VERSION ="1.0" STANDALONE = "YES"?>

```
<Fruits-table>
  <FRUITS>
    <FRUIT> <NAME> Apple <\NAME>
            <COLOR> Green <\COLOR>
    <\FRUIT>
    <FRUIT>  <NAME> Apple <\NAME>
            <COLOR> Red <\COLOR>
    <\FRUIT>
  <\FRUITS>
<\Fruits-table>
```

SQL Extensions (SQL 2003)

- xmlelement –→ creates XML elements from tabular data entity
- xmlattributes –→ creates attributes

```
select xmlelement ( name "account,
            xmlattributes (account_number as account_number),
            xmlelement ( name "branch_name",branch_name),
            xmlelement ( name "balance",balance))
 from account
```

SQL → XML

- • SQL 2003 –→ nested XML output
- • Each tuple –→ XML element

```
<bank>
    <account>
        <row>
            <account-number> A-101       </account-number>
            <branch-name>    Downtown </branch-name>
            <balance>        500         </balance>
        </row>
        <row>
                  more data .. . .
        </row>
    </account>
        . . .. . . . .
</bank>
```

Data in XML - SQL 2003

- Ability to specify new tags + create nested tag structures → XML is a way to exchange **data** (db) + documents.
  - XML → extensive use in data exchange applications
- Tags make data (relatively) self-documenting
  - E.g.

```
<university>
    <department>
      <dept_name> Comp. Sci. </dept_name>
      <building> Taylor </building>
      <budget> 100000 </budget>
    </department>
    <course>
       <course_id> CS-101 </course_id>
       <title> Intro. to Computer Science </title>
       <dept_name> Comp. Sci </dept_name>
       <credits> 4 </credits>
    </course>
</university>
```

Data in XML (new std. SQL2003)

- <university-3>
- <department dept name=“Comp. Sci.”>
- <building> Taylor </building>
- <budget> 100000 </budget>
- </department>
- <department dept name=“Biology”>
- <building> Watson </building>
- <budget> 90000 </budget>
- </department>
- <course course id=“CS-101” dept name=“Comp. Sci”
- instructors=“10101 83821”>
- <title> Intro. to Computer Science </title>
- <credits> 4 </credits>
- </course>
- ….
- <instructor IID=“10101” dept name=“Comp. Sci.”>
- <name> Srinivasan </name>
- <salary> 65000 </salary>
- </instructor>
- ….
- </university-3>

1. Different Contents- web documents

Web document	semi-structured data	Web query
Multiple Web documents	Semi-structured data	Web mining
Web structure and links	Structured data	Web mining
Web Usage logs and tables	Structured data	Web mining

Summary - 1

- 1. Content model → usage, interface, query ← Users
- 2. Representation
  - ← 1. storage level
  - → 2. content level

3. XML → widely researched and supported → authoring, editing, parsing, ….

Summary -2

- 1. XML query tools
  → xpath; xquery; xslt (all use xpath)
  → tree / arbitrary graph

2. SQL → can query GIS data and relational data (XML converted to relational form)

3. Query Interfaces → Type A and Type B
4. EHRs → AQL (uses SQL structure + XML addresses) ; XML templates

Summary - 3

- 1. SQL for map data
- 2. a) XML, b) XML query languages,
  c) XBase (free download)
- 3. Web Services and Web Resources
- 4. Recent increase in research activity→
  "New Query Language Interfaces"

5. High-level user interfaces

XML Examples

- Internet – RSS, ATOM
- XHTML, Web Service Formats: SOAP, WSDL

File Format: Microsoft Office, Open Office, Apple's iWork

Industrial- Insurance (ACORD),

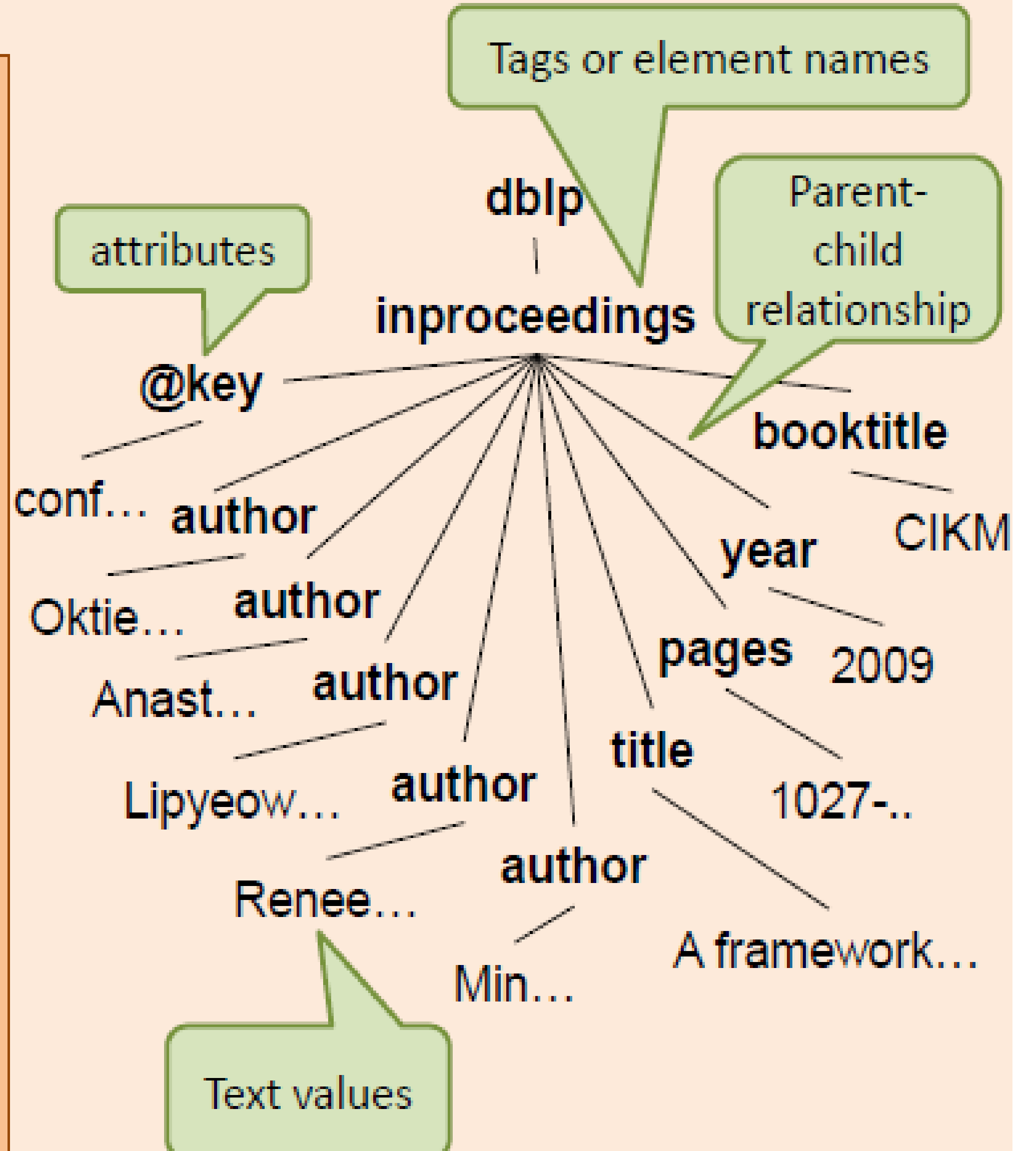
- Clinical Trials (cdisc)
- Financial (FIX, FpML)

- Many Applications use XML- Storage or Data exchange

XML Data Model

XML Document

```
<dblp>
  <inproceedings
    key="conf/cikm/HassanzadehKLMW0
    9" >
   <author>Oktie Hassanzadeh</author>
   <author>Anastasios
    Kementsietsidis</author>
   <author>Lipyeow Lim</author>
   <author>Renée J. Miller</author>
   <author>Min Wang</author>
   <title>
     A framework for semantic link
    discovery over relational data.</title>
   <pages>1027-1036</pages>
   <year>2009</year>
   <booktitle>CIKM</booktitle>
  </inproceedings>
</dblp>
```



Research Issues

- 1. Data → Chemistry structures, EHRs
- → → Structural information is captured in tree model or graph model for querying

2. Graph is more flexible

3. Tree model is simple → Single root, no cycle, unique path from root to a leaf.

Graph → pointer to ancestor and decendents

4. Semi-structured Data → schema sharing

XML – Most Recent Inovations

- Can be a Tree with UNIX directory style paths

- Can maintain redundant IDs to know the linked information

Data Interchange Stage 1, 2, and 3

- Program 1 → CSV (comma Separated values)
- Program 2 ← CSV values
- --
- Program data dump
- Stacks
- ORACLE Database Dump
- Arrays
- Abstract Data Types

- PROGRAMMING → to upload and process
- → knowledge of syntax and semantics
- --
- NEW TRENDS (data sharing among multiple applications) → YAML, Jason, XML, Candle

Information Interchange

- Information System 1 → Amazon Java books
- Information Systems 2 ← Amazon Books
- --
- 1. Objects – Books, Rooms with id and (x,y) coordinates, Students, Courses, ...
- 2. Documents – web documents, finacial statements of companies, …
- 3. Graphs and structures – Protiens, Maps, …
- Information → Sets (RDB) , relation ! DB !
- →Tree-structured Data (XML), …
- --
- → Syntax and Semantics
- Tree Structured Data → XML, JSON, Candle Markup

XML – STYLE MARKUP LANGUAGES

Data Mark-up : Configuration files, Internet Messages, Sharing Data and Objects between programming Languages

Document Mark-up : Web Documents, Database contents

Purpose : Exchange of data or exchange of documents, Storage

JSON → Javascript Object Notation

YAML → cross language, Unicode based, **data** serialization language (Data Mark-up)

Candle Mark-up → (Document mark-up for static data)

The syntax is based on XML, but have many differences

DOT → Graph Description Language

YAML

Designed → common data types of different programming languages.

Superset → JSON (YAML Version 1.2)

Goals:

1. easily readable by humans.
2. portable between programming languages.
3. matches the native data structures of most programming languages.
4. has a consistent model to support generic tools.
5. supports one-pass processing.
6. expressive and extensible.
7. is easy to implement and use.

YAML

YAML integrates and builds upon concepts

(many tools + Software)

described by C,

Java,

Perl, Python, Ruby,

RFC0822 (MAIL),

RFC1866 (HTML),

RFC2045 (MIME),

RFC2396 (URI),

XML, SAX, SOAP, and

JSON.

Reference: http://www.yaml.org/spec/1.2/spec.html (many more)

CANDLE MARKUP

Candle Markup → Document Markup

→**Can do Data Markup easily**

→**is an ideal format for general-purpose data serialization.**

→It works well for both structured object data and mixed text content.

→It has a terse and readable syntax, as well as,

→a clean and strongly-typed data model,

→It is better than many existing textual serialization formats: XML, JSON, YAML.

→Candle Markup is a subset of the Candle language

→used as a document format for static data.

→The syntax of Candle Markup is designed based on XML

XML – JSON

Example (XML)

```
<menu id="file" value="File">
<popup>
<menuitem value="New" onclick="CreateNewDoc()" />
<menuitem value="Open" onclick="OpenDoc()" />
<menuitem value="Close" onclick="CloseDoc()" />
</popup>
</menu>
```

Example (JSON)

```
{"menu": {
"id": "file", "value": "File",
"popup": {
"menuitem": [
{"value": "New", "onclick": "CreateNewDoc()"},
{"value": "Open", "onclick": "OpenDoc()"},
{"value": "Close", "onclick": "CloseDoc()"}
]
}
}}
```

CANDLE MARKUP – CANDLE OBJECT NOTATION

```
<?cmk1.0?>
menu {
id=file value="File"
popup {
menuitem { value="New" onclick="CreateNewDoc()" }
menuitem { value="Open" onclick="OpenDoc()" }
menuitem { value="Close" onclick="CloseDoc()" }
}
}
```

Candle Object Notation (comparison with JSON) :

→ objects have explicit name (instead of encoding it as key string);
→ attribute name does not need to be double quoted;
→ There's no need of delimiter, like comma, between the attributes.

DOT (graph description language)

- example script that describes the bonding structure of an ethane molecule. This is an undirected graph and contains node attributes. Useful for Special searches over bonding
- graph ethane {
- C_0 -- H_0 [type=s];
- C_0 -- H_1 [type=s];
- C_0 -- H_2 [type=s];
- C_0 -- C_1 [type=s];
- C_1 -- H_3 [type=s];
- C_1 -- H_4 [type=s];
- C_1 -- H_5 [type=s];
- }
- Many interfaces for graphic visualization and query

Conclusions

1. Information Interchange →
   documents, database

2. ADTs ➔ objects with schema details
→ Data standards called Languages (XML, JSON, ….)

3. Web data → Storage / transforms
   → scheme (semi-structured data)
   → Query facilities

Self Study - 3

Read about DOT

DOT (graph description language)

https://en.wikipedia.org/wiki/DOT_(graph_description_language)

(no submission is required)