Module 2

XML: Motivation

„If I invent another programming language, its name will contain the letter X.“

(N. Wirth, Software Pioniere Konferenz, Bonn 2001)

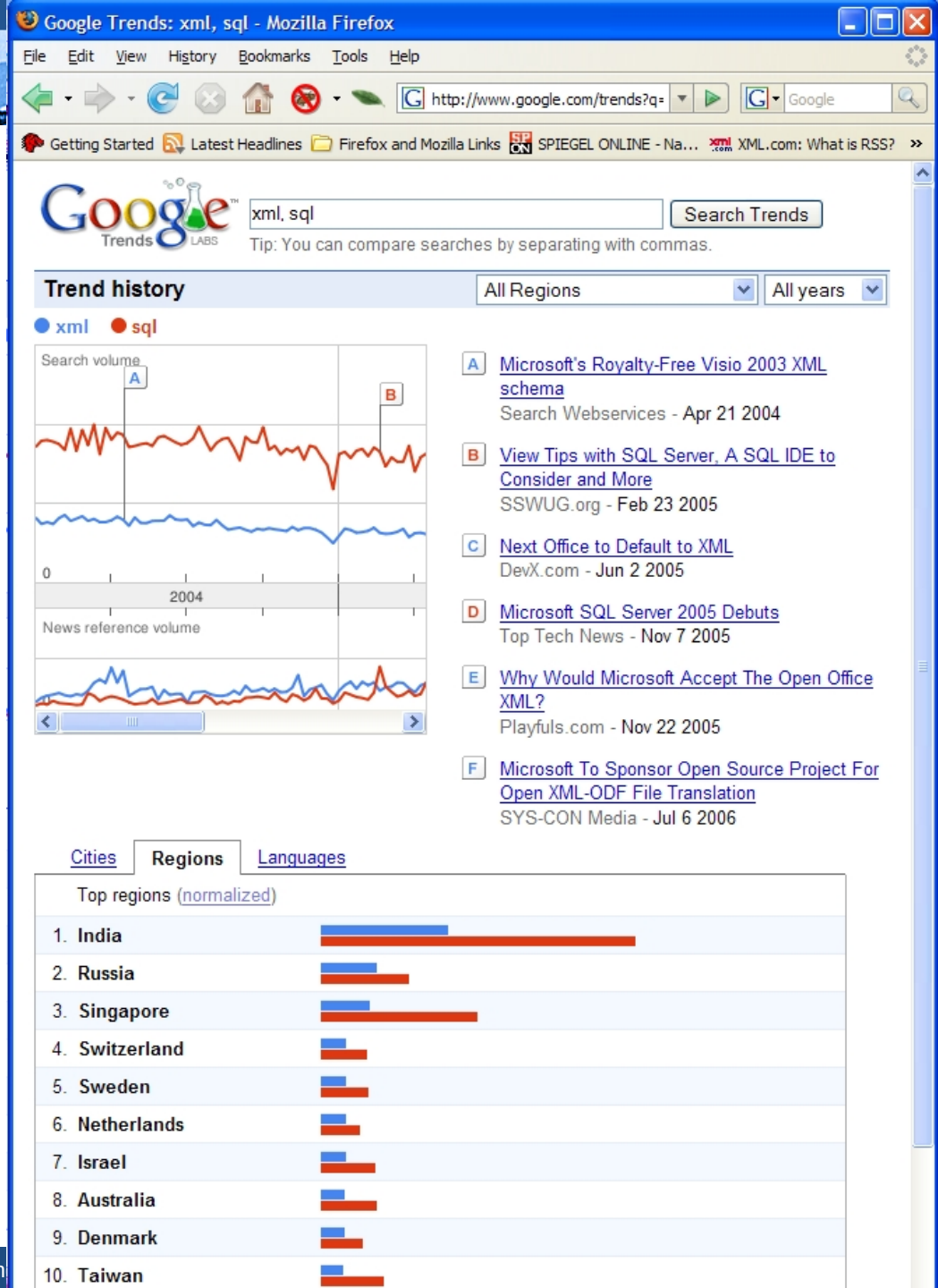
N-Way Googlefight: XML vs ...

XML	656 Mio
ABC	241 Mio
SQL	204 Mio
ETH	10.9 Mio
UBS	21.7 Mio

Love	**2200** Mio
Zurich	94 Mio
Soccer	229 Mio
Swiss	143 Mio
Peter Fischer	871 000
Donald Kossmann	56 500

Google Trends

- Monitoring querying pattern
  - XML is about half as popular as SQL
  - Switzerland is the 4th most active place to search for XML



What can the Web do for you?

- Download + show HTML Documents
- Forms
  - Pre-compiled point queries
  - Updates in specific Web application
- Everywhere, any time, platform independent
- Simple keyword search (Google)
- Good for human-human, human-machine communication

What the Web cannot do?

- Applications do not understand HTML
- Machine-Machine communication difficult
- Distributed Updates
- Long transactions (business processes)
- Powerful Queries
  - Where can I buy three electronic items for the lowest price (including shipping)

Some solutions upcoming (Mashups), technology very much related to course content

What Java and SQL can do?

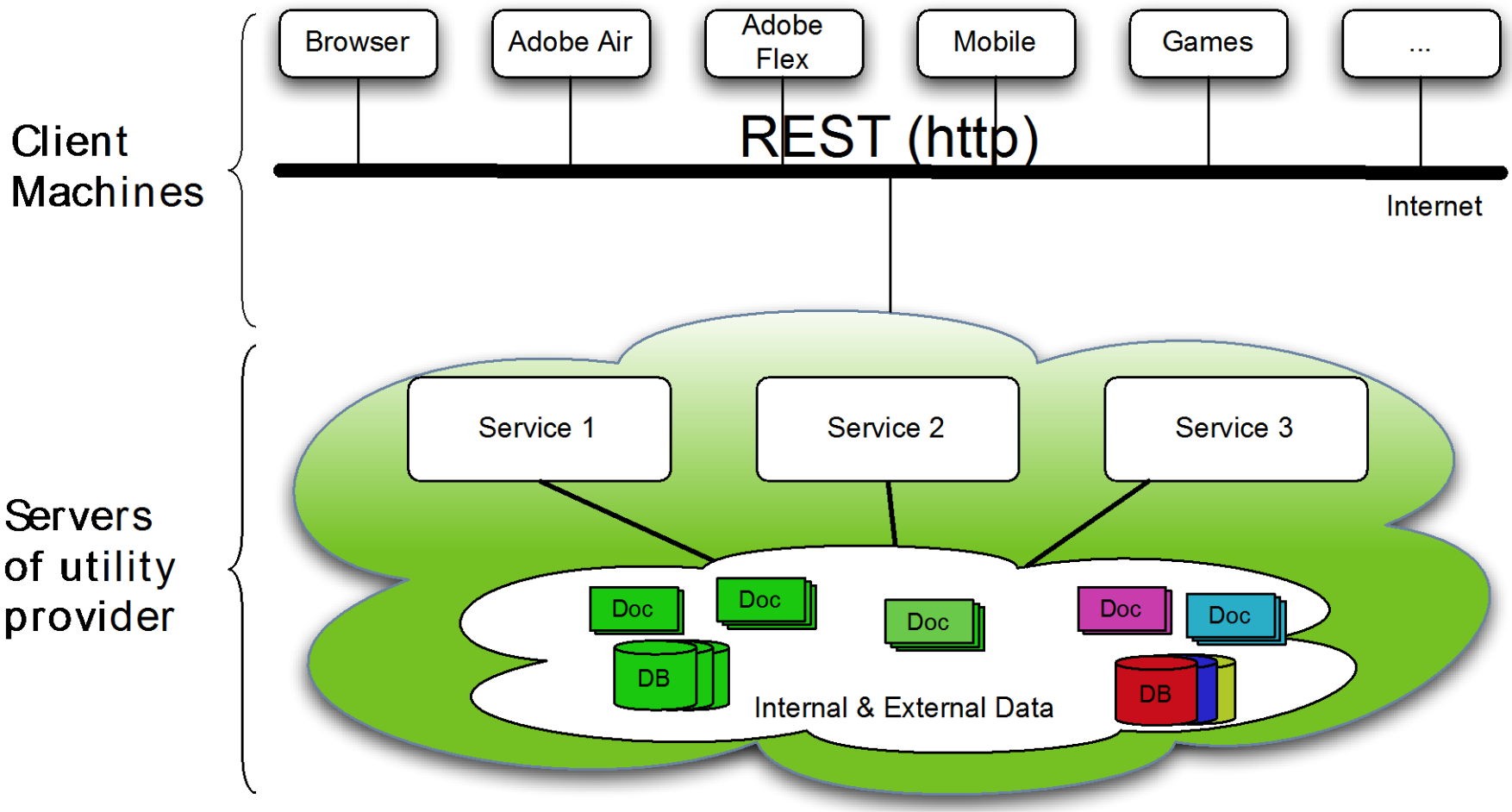
- Great to implement form-based apps
  - E.g., flight reservation, pizza service, etc.
- Okay for Business Intelligence
  - Complex SQL queries with number crunching
- Instead of Java, any other „web“ language could be given: PHP, Ruby, Perl, C#, …

What Java and SQL do not do well

- Documents and semi-structured data
  - Need „schema first“
- Put data in silos
  - Difficult to integrate and communicate data
- Efficiency in the cloud
  - How do you parallelize Java?
  - How do you optimize „Java + SQL“?
- Big war to create and own the next „Java+SQL“
  - NoSQL movement, Microsoft, Web 2.0, etc.
  - XML + XQuery: do not get hung up on marketing

Simple Truths

- „Power of data“
  - the more data the merrier (GB -> TB -> PB)
  - data comes from everywhere in all shapes
  - value of data often discovered later
  - data has no owner within an organization (no silos!)
- Services turn data into $
  - the more services the merrier (10s -> 1000s -> Ms)
  - need to adapt quickly
- Goal: Platforms for data and services
  - any data, any service, anywhere and anytime


Client Machines
Browser
Adobe Air
Adobe Flex
Mobile
Games
...
REST (http)
Internet
Servers of utility provider
Service 1
Service 2
Service 3
Doc
Doc
Doc
Doc
Doc
DB
DB
Internal & External Data

A little bit of history

Database world

- 1970 relational databases
- 1990 nested relational model and object oriented databases
- 1995 semi-structured databases

Documents world

- 1974 **SGML** (Structured Generalized Markup Language)
- 1990 **HTML** (Hypertext Markup Language)
- 1992 **URL** (Universal Resource Locator)

Data + documents = information

1996 **XML** (Extended Markup Language)

URI (Universal Resource Identifier)

What is XML?

- Lots of <>? (tag soup)
- “The Extensible Markup Language (XML) is the **universal format** for **structured documents** and **data** on the Web.”
- A **syntax** to **serialize** data
- **Family** of **standards**:
  Schema, Web Services, Processing, Semantic Web, …
- Base specifications:
  - XML 1.0, W3C Recommendation Feb '98
  - Namespaces, W3C Recommendation Jan '99

XML Data Example

```
<book year="1967">
    <title>The politics of experience
    </title>
     <author>
          <firstname>Ronald</firstname>
          <lastname>Laing</lastname>
     </author>
</book>
```

- Syntax, no abstract model
- Documents, elements and attributes
- Tree-based, nested, hierarchically organized structure

“Facebook” Profile in XML

```
<user id=“4711”>
    <name>John Doe</name>
        <friends>
            <friend id=“2”>Donald</friend>
            <friend id=“3”>Daisy</friend>
        </friends>
            <school>
                …
            </school>
</user>
```

Observation

- Documents are a quite natural way to represent „objects“.
  - A lot of NFNF (i.e., nested sets)
  - A great deal of text and semi-structured info
- Data in documents is often denormalized
  - (e.g., keep id and name of friends in profile)
  - That is also natural in many scenarios

Denormalized Data (ctd.)

- You have learnt to normalize schemas
  - Avoid redundancy
  - Avoid update anomalies
- Real data is often denormalized
  - Think of a FAX with an order
  - immutable: updates -> new version
  - No deletes in Facebook
- Technology Trends make Normalization less critical
  - Cheap storage, good indexing, ...
- But you can also normalize XML data!

XML vs. relational data

- Relational data
  - *Killer application*: Banking
  - Invented as a mathematically clean *abstract data model*
  - *Philosophy*: schema first, then data
- XML
  - First *killer application*: publishing industry
  - Invented as a *syntax for data, o*nly later an abstract data model
  - *Philosophy*: data and schemas should not be correlated, data can exist with or without schema, or with multiple schemas

XML vs. relational data, ctd.

- Relational data
  - Never had a *standard syntax* for data
  - Strict rules for data *normalization*, flat tables
  - *Order* is irrelevant, textual data supported but not primary goal
- XML
  - *Standard syntax* existed before the data model
  - No data *normalization*, flexibility is a must, nesting is good
  - *Order may* be very important, textual data support a primary goal

What about OO approaches?

Reasons for the XML success

- XML is a general data representation format
- XML is human readable
- XML is machine readable
- XML is internationalized (UNICODE)
- XML is platform independent
- XML is vendor independent
- XML is endorsed by the W3C
- XML is not a new technology
- XML is not *only* a data representation format, it's a full infrastructure of technologies

Killer Applications for XML

- Data lives forever (longer than program code)
  - legacy systems: need to keep code to keep data
  - huge IT infrastructures
- *„hello world“* program is very complex
  - Model *before* Data (you need to know what you want)
  - poor „time to market“, high cost
- SQL + Objects are not enough
  - middleware, data marshalling, …
  - No querying of objects, no encapsulation in SQL
  - expensive (five star guru) programmers needed
- **XML: Decouple Data and Schema!!!**

Killer XML advantages

1. *Code/schema/data independence*
2. Covers the continuous spectrum from totally *structured data* to *documents*
   - *from data management to information management*
3. Unique/Uniform model for representing *data,* *metadata* and *code*

Data + metadata + code

- Data (XML), schemas (XML Schemas) and code (XSLT, XQuery): they all have an XML syntax
- Easy to mix and match:
  - Data in the schemas (not yet)
  - Data in code (already done)
  - Code in schemas (current research project): *Unity*
  - Code in the data (already done) : Active XML

Why is XML relevant from DB perspective?

- XML is the becoming *the* data „format“
  - Amount of XML is ever increasing,
  - DBMS are good at handling GBs,TBs of data, getting into PBs now
- Accepted model for semi-structured data
  - Overcome limitations of structured data
  - Extend usefulness of DBMS
- DB technology is not limited to DBMS
  - Apps servers, application integration

Myths about XML

- XML is complicated
  - some unnecessary stuff (documents, ...)
  - some XML family members (XML Schema...)
  - but best package that is out there
- XML is slow
  - only implementations can be slow
- SQL is better
  - Huh??? For what?
- XML is dead
  - there is more XML than relat. data out there!!!

Misunderstanding about XML

- "Data is self-describing."
- Tags don't hold *semantics*, they only hold the *structure* of the information
- The interpretation of the tags is in the *application* that handles the data, not in the tags themselves.

XML handicaps

- "Tree, and not a graph."
  - Difficulty in modeling N:M relationships
  - The notion of reference (e.g. XLink, XPointer) not well integrated in the XML stack
- "Duplication of concepts"
  - Many ways to do the same thing
  - Justification for a "simpler" data model like RDF
- "Concepts that *seem* logically unnecessary"
  - PIs, comments, documents, etc
- Additional complexity factors
  - xsi:nil, QName in content, etc
- "Boring"
  - so is the (enterprise) world where XML lives

Advantages and disadvantages

1. “Handles the dual aspect of information: lexical and binary” : 1 and “01”

- Essential feature for the 21st century information management
  - E.g. XML-based contract to be used in a legal procedure
- Lots of complexity derives from here
  - XML Schema deals with both *lexical* and *binary constraints*
  - XML Data Model has to include both the *dm:typed-value* and *dm:string-value*
  - Processing language like XQuery and XSLT have to define their semantics for *both* aspects
  - XML data *storage* and *indexing* heavily impacted
  - Problems with Signing XML Data (when is XML equivalent)

Advantages and disadvantages

2. “Data is context sensitive.”

- We cannot do *cut and paste* in XML
- Certain aspects of the data depend on the context where the fragment of data occurs (base-URIs, namespaces,etc)
- Valuable feature for document management
- Very hard consequences on storing, indexing and processing XML
- Semantics of expressions also depends on the context where they appear
- Additional consequences on expression evaluation

Sources of XML data ?

1. Inter-application communication data (WS, REST, etc)
2. Mobile devices communication data
3. Logs
4. Blogs (RSS)
5. Metadata (e.g. Schema, WSDL, XMP)
6. Presentation data (e.g. XHTML)
7. Documents (e.g. OOXML, ODF)
8. Views of other sources of data
   - Relational, LDAP, CSV, Excel, etc.
9. Sensor data

It would be interesting to know the pie-chart and the evolution of each branch !

Some vertical app domains for XML

- *HealthCare Level Seven http://www.hl7.org/*
- *Geography Markup Language (GML)*
- *Systems Biology Markup Language (SBML) http://sbml.org/*
- *XBRL, the XML based Business Reporting standard http://www.xbrl.org/*
- *Global Justice XML Data Model (GJXDM) http://it.ojp.gov/jxdm*
- *ebXML http://www.ebxml.org/*
- *e.g. Encoded Archival Description Application http://lcweb.loc.gov/ead/*
- *Digital photography metadata XMP*
- *An XML grammar for sensor data (SensorML)*
- *Real Simple Syndication (RSS 2.0)*

Basically everywhere.

Alternatives: Other formats

- Edifact, CSV, JSON, PDF, ProtBuf, Avro, ...
  - Has conversions to XML
  - Part of any good XQuery library
  - Most of them are application-specific
- Office (Word, Excel, PPT), RSS, Atom, RDF
  - Already XML
- XML is the „mother“ of all data formats
  - Can express everything
  - Comes at a cost!

JSON

- En vogue because of JavaScript

```
{  “book": {
        "title": „The politics of experience“
         “author": {
               “firstname“: „David“
               „lastname“: „Laing“
          }
} }
```

- Pretty much the same as XML
  - Do not worry too much about syntax.
  - From a high-level point very similar

Protocol Buffers

- Used by Google internally
  - nested (EBNF) data structure like JSON and XML
  - http://code.google.com/apis/protocolbuffers
- Apparently much faster to parse

Examples (S. Melnik)

Schema and Data:

```
message Document {
  required int64 DocId;                [1,1]
  optional group Links {
    repeated int64 Backward;           [0,*]
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;         [0,1]
    }
    optional string Url;
  }
}
```

```
DocId: 10
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

```
DocId: 20
Links
  Backward: 10
  Backward: 30
  Forward:  80
Name
  Url: 'http://C'
```

Why do we still talk about XML?

- It is a standard (not owned by anybody)
- Very well documented
- Many tools available
- Mother of all structured / semi-struct. data
  - has the most features
- XML is here to stay
- It actually works! ☺