Large-Scale and Multi-Structured Databases

Document Databases

Introduction

Prof. Pietro Ducange

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Main Features

Document databases are ***non-relational*** databases that store data as structured documents, usually in ***XML*** or ***JSON*** formats.

They ensure a ***high flexibility*** level and allow also to handle ***complex*** data ***structures*** (despite key-value databases).

Document databases are ***schema-less.***

They allow ***complex operations***, such as queries and filtering.

Some document databases allow also ***ACID transactions***.

XML Documents

- XML stands for ***eXtensible Markup Language***.
- A markup language specifies the ***structure*** and ***content*** of a ***document***.
- ***Tags*** are added to the document to provide the ***extra information***.
- XML tags give a reader some idea what some of the ***data means***.
- XML is capable of representing almost ***any form*** of information.

XML Documents: Use cases

1. XML and Cascading Style Sheets (CSS) allowed ***second-generation websites*** to separate data and format.

2. XML is also the basis for many data ***interchange protocols*** and, in particular, was a foundation for web service specifications such as ***SOAP*** (Simple Object Access Protocol).

3. XML is a ***standard*** format for many document types, including word processing documents and spreadsheets (***docx***, ***xlsx*** and ***pptx*** formats are based on XML).

Advantages of XML

- XML is text (Unicode) based.
  - Can be transmitted efficiently.
- One XML document can be displayed differently in different media and software platforms.
- XML documents can be modularized. Parts can be reused.

An Example of XML Document

```
<item>
   <title>Kind of Blue</title>
   <artist>Miles Davis</artist>
   <tracks>
      <track length="9:22">So What</track>
      <track length="9:46">Freddie Freeloader</track>
      <track length="5:37">Blue in Green</track>
      <track length="11:33">All Blues</track>
      <track length="9:26">Flamenco Sketches</track>
   </tracks>
</item>
<item>
   <title>Cookin'</title>
   <artist>Miles Davis</artist>
   <tracks>
      <track length="5:57">My Funny Valentine</track>
      <track length="9:53">Blues by Five</track>
      <track length="4:22">Airegin</track>
      <track length="13:03">Tune-Up</track>
   </tracks>
</item>
<item>
   <title>Blue Train</title>
   <artist>John Coltrane</artist>
   <tracks>
      <track length="10:39">Blue Train</track>
      <track length="9:06">Moment's Notice</track>
      <track length="7:11">Locomotion</track>
      <track length="7:55">I'm Old Fashioned</track>
      <track length="7:03">Lazy Bird</track>
   </tracks>
</item>
```

XML Ecosystem

XPath: useful to ***navigate*** through elements and attributes in an XML document.

XQuery: is the language for ***querying*** XML data and is built on XPath expressions.

XML schema: A special type of XML document that describes the ***elements*** that may be present in a specified class of XML documents.

XSLT (Extensible Stylesheet Language Transformations): A language for ***transforming*** XML documents into alternative formats, including non-XML formats such as HTML.

DOM (Document Object Model): a platform- and language-neutral interface for dynamically managing the content, structure and style of documents such as XML and XHTML. A document is handled as tree.

XML Databases

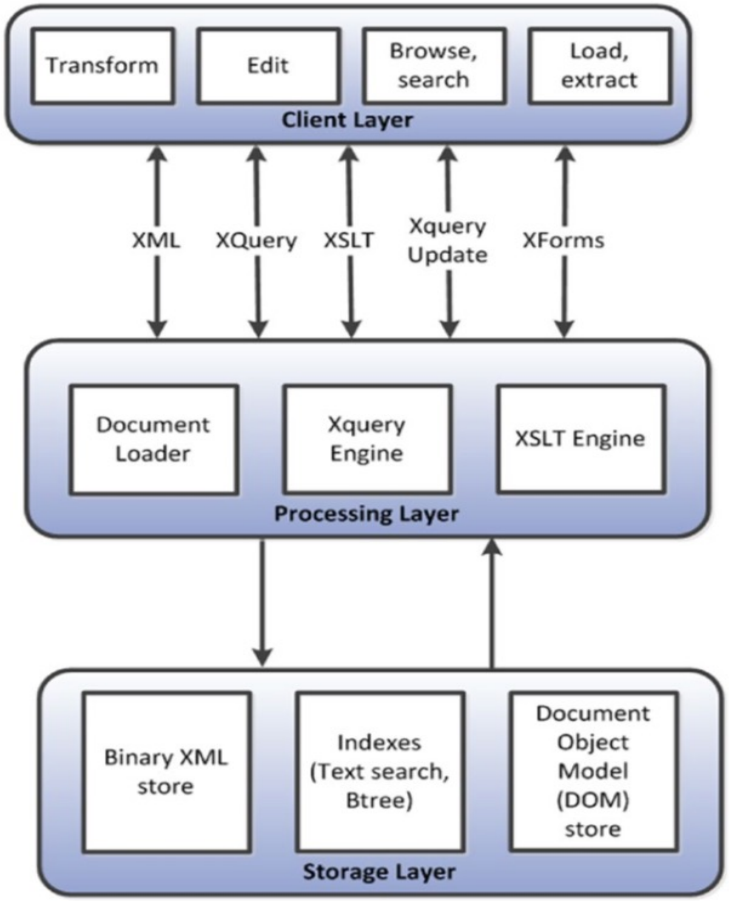


Image extracted from "Guy Harrison, Next Generation Databases, Apress, 2015"

XML databases: ***platforms*** that implement the various XML standards such as XQuery and XSLT,

They provide ***services*** for the ***storage***, ***indexing***, ***security***, and concurrent ***access*** of XML files.

XML databases ***did not represent an alternative*** for RDBMSs.

On the other hand, some RDBMSs introduced XML , allowing the ***storage of XML*** documents within A BLOB (binary large object) columns.

XML : Main Drawbacks

- XML tags are ***verbose*** and ***repetitious***, thus the amount of storage required increases.
- XML documents are ***wasteful of space*** and are also ***computationally expensive*** to parse.
- In general, XML databases are used as ***content-management systems***: collections of text files (such as academic papers and business documents) are organized and maintained in XML format.
- On the other hand, ***JSON-based*** document databases are more suitable to support ***web-based*** operational ***workloads***, such as storing and modifying dynamic contents.

JSON Documents

- JSON acronym of ***JavaScript Object Notation.***
- Used to format data.
- Thanks to its integration with JavaScript, a JSON document has been often preferred to an XML document for ***data interchanging*** on the Internet.

JSON example

- "JSON" stands for "JavaScript Object Notation"
  - Despite the name, JSON is a (mostly) language-independent way of specifying objects as name-value pairs

```
{"skillz": {
        "web":[
                {"name": "html",
                 "years": "5"
                },
                {"name": "css",
                 "years": "3"
                }],
        "database":[
                {"name": "sql",
                 "years": "7"
                }]
}}
```

Image extracted from http://secretgeek.net/json_3mins

JSON syntax

- An ***object*** is an unordered set ***of name/value pairs***
  - The pairs are enclosed within braces, { }
  - There is a colon between the name and the value
  - Pairs are separated by commas
  - Example: { "name": "html", "years": 5 }
- An ***array*** is an ***ordered collection*** of values
  - The values are enclosed within brackets, [ ]
  - Values are separated by commas
  - Example: ["html", ”xml", "css"]

JSON syntax

- A *value* can be: A string, a number, true, false, null, an object, or an array
  - Values can be nested
- *Strings* are enclosed in double quotes, and can contain the usual assortment of escaped characters
- *Numbers* have the usual C/C++/Java syntax, including exponential (E) notation
  - All numbers are decimal--***no octal or hexadecimal***

Example of Nested Objects

```
{
  "sammy" : {
    "username"  : "SammyShark",
    "location"  : "Indian Ocean",
    "online"    : true,
    "followers" : 987
  },
  "jesse" : {
    "username"  : "JesseOctopus",
    "location"  : "Pacific Ocean",
    "online"    : false,
    "followers" : 432
  },
  "drew" : {
    "username"  : "DrewSquid",
    "location"  : "Atlantic Ocean",
    "online"    : false,
    "followers" : 321
  },
  "jamie" : {
    "username"  : "JamieMantisShrimp",
    "location"  : "Pacific Ocean",
    "online"    : true,
    "followers" : 654
  }
}
```

Image extracted from: https://www.digitalocean.com/community/tutorials/an-introduction-to-json

Example of Nested Arrays

```
{
  "first_name" : "Sammy",
  "last_name" : "Shark",
  "location" : "Ocean",
  "websites" : [
    {
      "description" : "work",
      "URL" : "https://www.digitalocean.com/"
    },
    {
      "desciption" : "tutorials",
      "URL" : "https://www.digitalocean.com/community/tutorials"
    }
  ],
  "social_media" : [
    {
      "description" : "twitter",
      "link" : "https://twitter.com/digitalocean"
    },
    {
      "description" : "facebook",
      "link" : "https://www.facebook.com/DigitalOceanCloudHosting"
    },
    {
      "description" : "github",
      "link" : "https://github.com/digitalocean"
    }
  ]
}
```

Image extracted from: https://www.digitalocean.com/community/tutorials/an-introduction-to-json

Comparison of JSON and XML

- ***Similarities***:
  - Both are human readable
  - Both have very simple syntax
  - Both are hierarchical
  - Both are language independent
  - Both supported in APIs of many programming languages
- ***Differences***:
  - Syntax is different
  - JSON is less verbose
  - JSON includes arrays
  - Names in JSON must not be JavaScript reserved words

JSON vs XML

users.xml

```
<users>
    <user>
        <username>SammyShark</username> <location>Indian Ocean</location>
    </user>
    <user>
        <username>JesseOctopus</username> <location>Pacific Ocean</location>
    </user>
    <user>
        <username>DrewSquir</username> <location>Atlantic Ocean</location>
    </user>
    <user>
        <username>JamieMantisShrimp</username> <location>Pacific Ocean</location>
    </user>
</users>
```

users.json

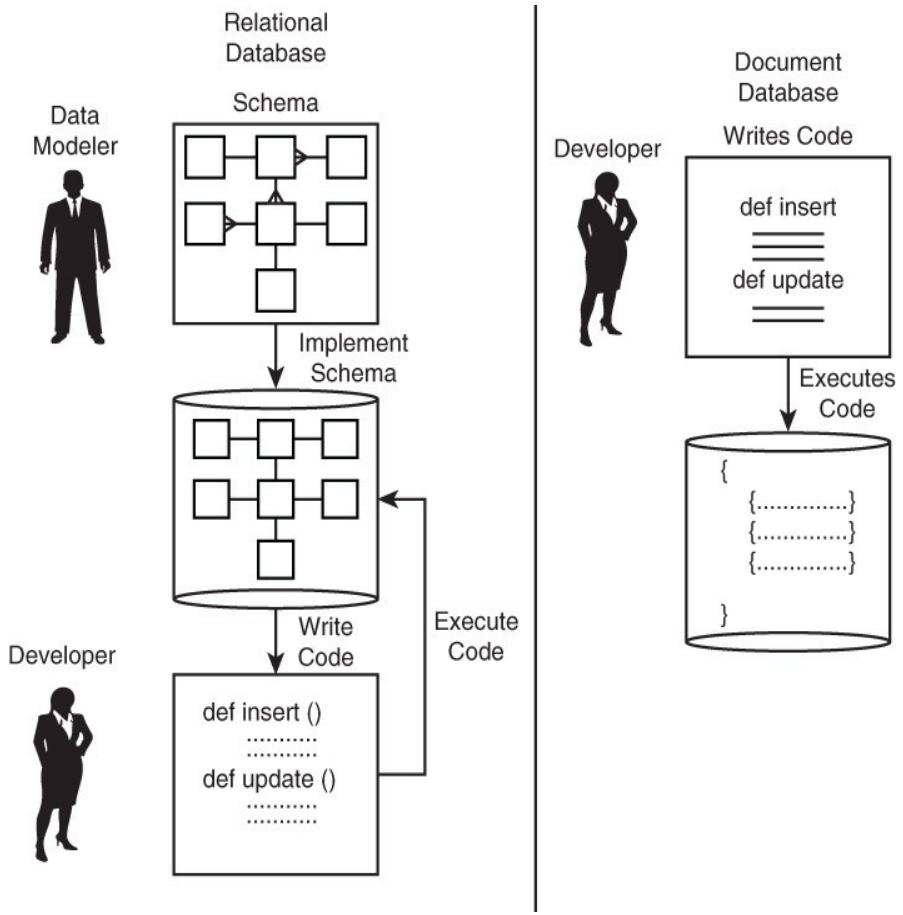
```
{"users": [
  {"username" : "SammyShark", "location" : "Indian Ocean"},
  {"username" : "JesseOctopus", "location" : "Pacific Ocean"},
  {"username" : "DrewSquid", "location" : "Atlantic Ocean"},
  {"username" : "JamieMantisShrimp", "location" : "Pacific Ocean"}
] }
```

Image extracted from: https://www.digitalocean.com/community/tutorials/an-introduction-to-json

Main Feature of JSON Databases

- Data is stored in JSON format.
- A ***document*** is the basic ***unit*** of storage. It includes one or more more ***key-value pairs*** and may also contain ***nested documents*** and ***arrays***.
- ***Arrays*** may also ***contain documents*** allowing for a complex hierarchical structure.
- A ***collection*** or ***data bucket*** is a set of documents sharing some ***common purpose***.
- ***Schema less***: predefined document elements must not be defined.
- ***Polymorphic Scheme***: the documents in a collection may be ***different***.

Schema-less vs Schema Definition



A *schema* is a specification that describes the ***structure of an object***, such as a table.

Data modelers have to define ***tables*** in a relational database before ***developers*** can execute ***code*** to add, remove, or update rows in the table.

Document databases do not require this formal definition step.

Developers can ***create*** collections and documents in collections by simply ***inserting*** them into the database

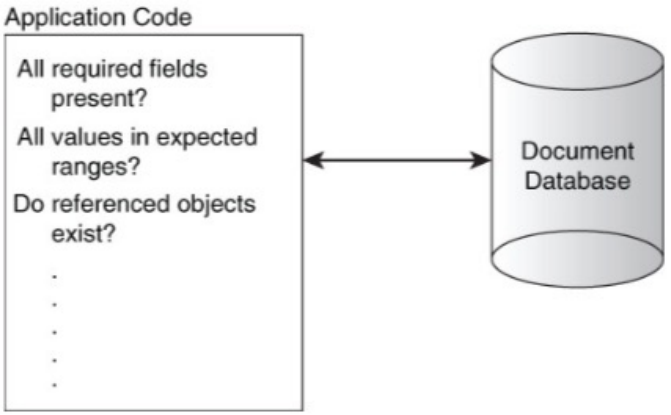
Schema-less pros and cons

Pros: High flexibility in handling the structure of the objects to store

```
{ 'employeeName' : 'Janice Collins',
  'department' : 'Software engineering'
  'startDate' : '10-Feb-2010',
  'pastProjectCodes' : [ 189847, 187731, 176533, 154812]
}
```

```
{ 'employeeName' : 'Robert Lucas,
  'department' : 'Finance'
  'startDate' : '21-May-2009',
  'certifications' : 'CPA'
}
```

Cons: the DBMS may be not allowed to enforce rules based on the structure of the data.



Some considerations

A *document database* could ***theoretically*** implement a ***third normal form schema***.

Tables, as in relational databases, may be "***simulated***" considering collections with ***JSON*** documents with an identical ***pre-defined structure***.

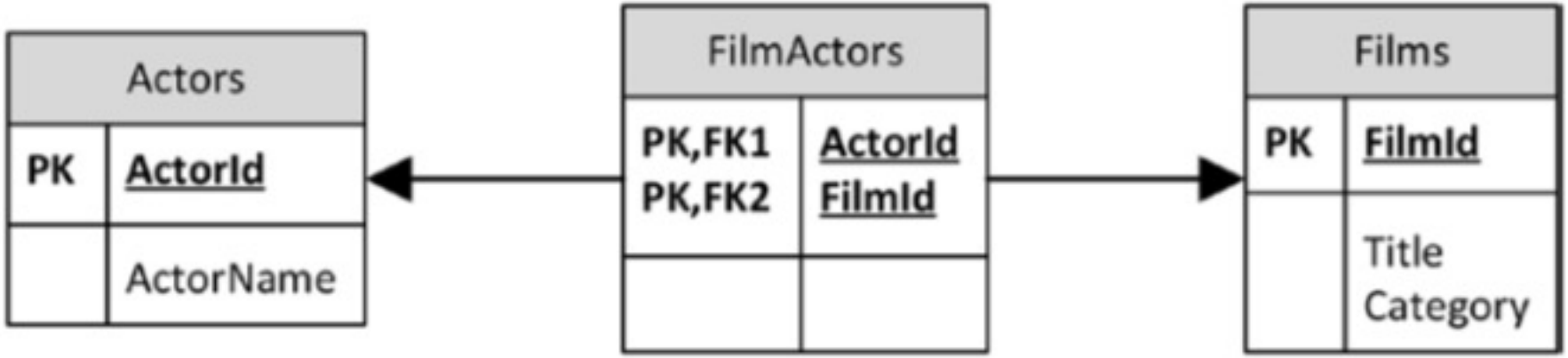


Image extracted from "Guy Harrison, Next Generation Databases, Apress, 2015"

JSON Databases: An example

Document databases usually adopts a ***reduced number*** of collections for modeling data.

Nested documents are used for representing ***relationships*** among the different entities.

Document databases ***do not*** generally provide ***join operations***.

Programmers like to have the JSON structure map ***closely to the object*** structure of their code!!!

Image extracted from "Guy Harrison, Next Generation Databases, Apress, 2015"

Suggested Readings

Chapter 4 of the book "*Guy Harrison, Next Generation Databases, Apress, 2015*".

Chapters 6 of the book "*Dan Sullivan, NoSQL For Mere Mortals, Addison-Wesley, 2015*"