

Document  
Database and  
Map-Reduce

Introduction

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

RDBMS vs  
Document  
database

Document  
Database

Document  
Data-Model

Querying  
collections and  
objects

# Document Database and Map-Reduce

Hogeschool Rotterdam  
Rotterdam, Netherlands

## Lecture topics

- CAP Theorem ACID vs BASE
- Document Databases
- MongoDB
- Map-Reduce
- Summery

## Motivation

- As we mentioned before relational database systems are designed to run on a single server
- RDBMS satisfy the ACID rules to provide consistency and availability of the data for the users
- But how do NoSQL databases deal with the data in their implementation?

## CAP theorem

- States that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:
  - **Consistency** every read receives the most recent write or an error
  - **Availability** every request receives a response, without guarantee that it contains the most recent version of the information
  - **Partition** tolerance (the system continues to operate despite arbitrary partitioning due to network failures)

# NoSQL database and CAP theorem

## ACID vs BASE

- Basically **A**vailable, **S**oft State and **E**ventual **C**onsistent
- Because of this characteristic the query language must be able to process data saved locally and in a cluster (to be discussed in another slide)

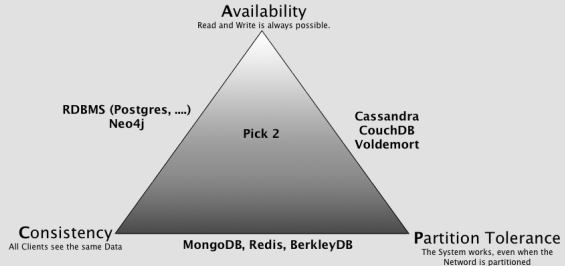


Figure: CAP Theorem

## ACID vs BASE

- Relational databases are considered to be structural
- Document databases uses semi-structured formats
- text files such as logs are unstructured

## Introduction

- A document database is a nonrelational database that stores data as structured documents such as in XML or JSON formats
- Document databases are free to implement ACID transactions or other characteristics of a traditional RDBMS
- A document database allows some form of data description without enforcing a schema
- The alignment with web-development programming practices has resulted in JSON and document databases/storage

## Introduction

- Let us see what are those formats and how they are used!
- We will start with eXtensible Markup Language
- Then we will look at JavaScript Object Notation and it's Binary version



## eXtensible Markup Language (XML)

- Defined by the WWW Consortium (W3C)
- Extensible, unlike HTML, users can add new tags, and separately specify how the tag should be handled for display
- XML has become the basis for all new generation data interchange formats. For instance bank transfers and secure document exchange
- Documents have tags giving extra information about sections of the document. Those tags can also be nested

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="UTF-8"/>
    <title>Polyglot (X)HTML Template</title>
  </head>
  <body>
    <p>content....</p>
  </body>
</html>
```

## XML

- Each XML based standard defines what are valid elements, using XML type specification languages to specify the syntax
- DTD (Document Type Descriptors): describes the structure of an XML document
- XML Schema (newer than DTD): a special type of XML document that describes the elements that may be present
- Sample implementation database BaseX ([basex.org](http://basex.org))

# Document Type Descriptors

Document  
Database and  
Map-Reduce

Introduction

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

RDBMS vs  
Document  
database

Document  
Database

Document  
Data-Model

Querying  
collections and  
objects

```
<! ELEMENT department(dept_name, building, budget)>
<! ELEMENT dept_name (#PCDATA)>
<! ELEMENT budget (#PCDATA)>
<!ELEMENT university ( ( department | course |
    instructor | teaches )+)>
```

Notation:

	:	alternatives
+	:	1 or more occurrences
*	:	0 or more occurrences
#PCDATA	:	Parsed character data i.e. parsed string

## XML Processing

- XPath: A syntax for retrieving specific elements from an XML document using wildcards.
- XQuery: A query language provides mechanisms for modifying a document. XQuery is sometimes referred to as “the SQL of XML”.
- 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): An object-oriented API that programs can use to interact with XML, XHTML, and similarly structured documents.

## Tree Model of XML Data

- Query and transformation languages are based on a tree model of XML data
- An XML document is modeled as a tree, with nodes corresponding to elements and attributes

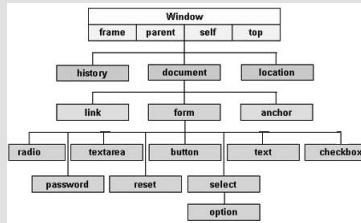


Figure: DOM sample of an HTML document

## XML Processing: XPath

- XPath: is used to address (select) parts of documents using path expressions
  - The initial denotes root of the document (above the top-level tag)
  - Think of file names in a directory hierarchy
  - Selection predicates may follow any step in a path, in [ ]
  - It is possible to apply selection criteria on the values using comparison operators <sup>a</sup>

---

<sup>a</sup>Demonstrate an example

## XML Processing: XQuery and XPath

- XQuery is derived from the Quilt query language, which itself borrows from SQL, XQL and XML-QL
- XQuery uses a: for ... let ... where ... order by ... result ...
  - $\begin{matrix} a & b \end{matrix}$
  - for = from
  - where = where
  - order by = order by
  - result = select

<sup>a</sup>**let**: allows temporary variables, and has no equivalent in SQL

<sup>b</sup>Demonstrate an example



## JavaScript Object Notation JSON

- JSON is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs
- JSON Schema is based on the concepts from XML Schema, but is JSON-based
- Document databases use JSON documents in order to store records, just as tables and rows store records in a relational database

## BSON

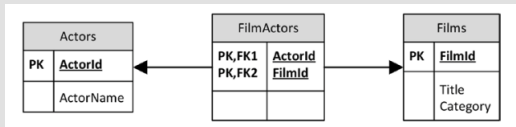
- BSON: binary-encoded format used in MongoDB instead of JSON
- BSON extends the JSON model to provide additional data types such as integer and float to be efficient for encoding and decoding within different languages.
- BSON implementation supports embedding objects and arrays within other objects and arrays

## MongoDB

- A MongoDB instance may have zero or more databases
- A database may have zero or more collections.
  - Can be thought of as the relation (table) in DBMS, but with many differences.
- A collection may have zero or more documents.
  - Docs in the same collection don't even need to have the same fields
  - Docs are the records in RDBMS
  - Docs can embed other documents
  - Documents are addressed in the database via a unique key differences.
- A document may have one or more fields.
- There is no join provided in MongoDB. You have to implement it manually.

## Relational Model

- Suppose you have the following entities and their relationships



- How would we model this in a document structure?

# Document Data-Model

Document  
Database and  
Map-Reduce

Introduction

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

RDBMS vs  
Document  
database

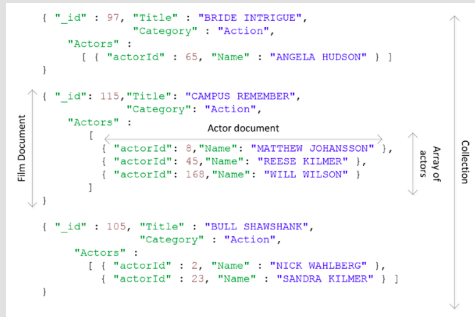
Document  
Database

Document  
Data-Model

Querying  
collections and  
objects

## Embedded

- First mapping possibility is to map to one embedded collection.



- document database are not designed to be normalized and data repetition is accepted, but could have side effects.

## Linking using object \_id

- Second mapping possibility is to map to different collections and link the documents

```
{ "_id": 115, "Title": "CAMPUS REMEMBER", "Category": "Action",  
  "Actors": [8, 45, 168]}
```

```
{ "actorId": 168, "Name": "WILL WILSON" }
```

```
{ "actorId": 45, "Name": "REESE KILMER" }
```

```
{ "actorId": 8, "Name": "MATTHEW JOHANSSON" }
```

- This approach is less suited for document databases since the binary data of those collections are not stored as a continuous stream.
- Another disadvantage of this approach is the lack of join query

# Querying collections and objects

Document  
Database and  
Map-Reduce

Introduction

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

RDBMS vs  
Document  
database

Document  
Database

Document  
Data-Model

Querying  
collections and  
objects

## Selecting data

- Select queries in MongoDB and SQL

<code>SELECT a,b FROM users</code>	<code>db.users.find({}, {a:1,b:1})</code>
<code>SELECT * FROM users</code>	<code>db.users.find()</code>
<code>SELECT * FROM users WHERE age=33</code>	<code>db.users.find({age:33})</code>
<code>SELECT a,b FROM users WHERE age=33</code>	<code>db.users.find({age:33}, {a:1,b:1})</code>
<code>SELECT * FROM users WHERE age=33 ORDER BY name</code>	<code>db.users.find({age:33}).sort({name:1})</code>
<code>SELECT * FROM users WHERE age&gt;33</code>	<code>db.users.find({age:{\$gt:33}})</code>
<code>SELECT * FROM users WHERE age!=33</code>	<code>db.users.find({age:{\$ne:33}})</code>

# Querying collections and objects

Document  
Database and  
Map-Reduce

Introduction

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

NoSQL  
database and  
CAP theorem

RDBMS vs  
Document  
database

Document  
Database

Document  
Data-Model

Querying  
collections and  
objects

## Inserting data

- Adding query examples or live demo



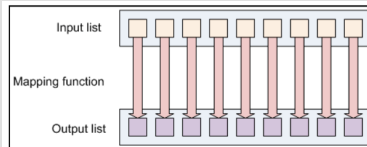
## Introduction

- Map-Reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results.
- Map- and Reduce-functions are commonly used in functional programming
- In INFDEV02-2 and INDEV02-3 we already introduced HOFs
- Higher Order Functions HOFs are very powerful in the context for NoSQL databases.
- The following functions will be further discussed : FlatMap, Map and Reduce

## Map Function

- Apply the function  $f$  to each element of list  $x$
- $\text{map}(f, x[0\dots n-1])$
- in Python:

```
def square(x): return x * x
map(square, [1, 2, 3, 4])
would return [1, 4, 9, 16]
```



map/reduce images src: Apache Hadoop tutorials

## Reduce Function

- Repeatedly apply binary function  $f$  to pairs of items in  $x$ , replacing the pair of items with the result until only one item remains
- $\text{reduce}(f, x[0\dots n-1])$
- in Python:  

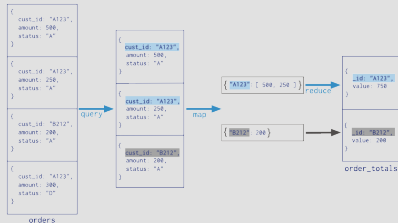
```
def add(x,y): return x+y  
reduce(add, [1,2,3,4])
```

would result in a 10

## MapReduce Function in MongoDB

- Not done yet

```
Collection
↓
db.orders.mapReduce(
  map → function() { emit( this.cust_id, this.amount ); },
  reduce → function(key, values) { return Array.sum( values ); },
  {
    query → { status: "A" },
    output → "order_totals"
  }
)
```



## MapReduce Function in a Cluster

- The distributed MapReduce idea is similar to (but not the same as!):  $\text{reduce}(f2, \text{map}(f1, x))$
- Key idea: "data-centric" architecture – Send function  $f1$  directly to the data : Execute it concurrently
- Then merge results with reduce: Also concurrently