CouchDB experience: wrapup

Genoveva Vargas-Solar
CR1, CNRS, LIG-LAFMIA
Genoveva.Vargas@imag.fr

http://vargas-solar.com, Montevideo, 18nd November, 2014

DATA MODEL

Data model: Document

- An object with named attributes and «*attachments*»:
  - Identified by one unique ID and a version number
  - Different data types: Text, numbers, booleans, dates, lists, maps
  - Does not use locks for dealing with concurrency control: conflicts can be merged
- Examples:
  - `“Title”: “CouchDB: The Definitive Guide: Time to Relax (Animal Guide)”`
  - `“Authors”: [“Chris Anderson”, “Jan Lehnardt”, “Noah Slater”]`
  - `“Keywords”: [“NoSQL databases”, “Document databases”]`

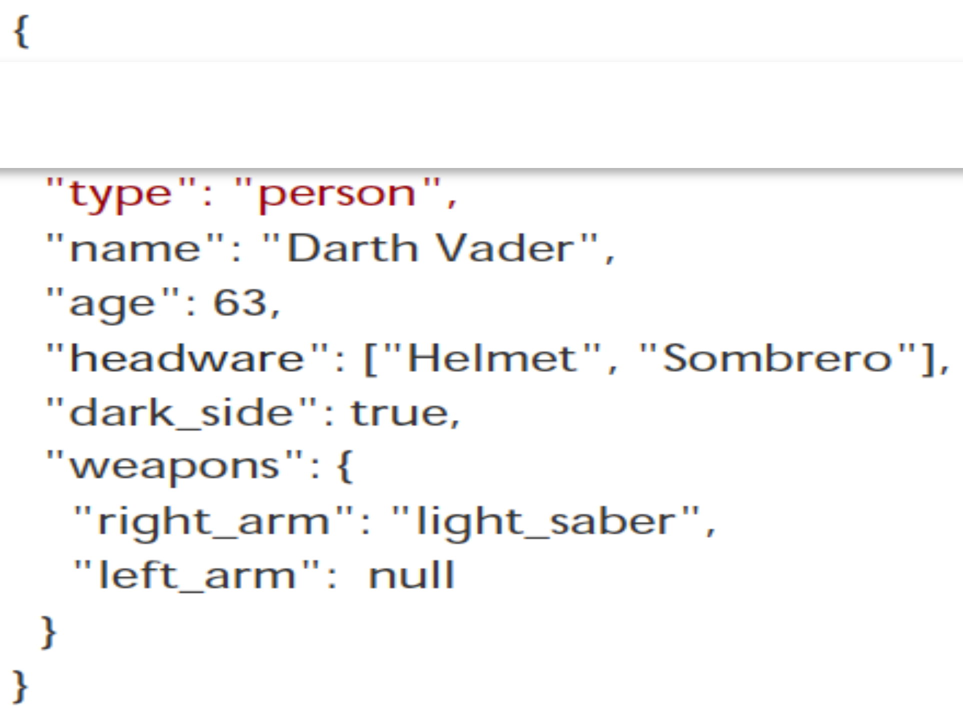
JavaScript Object Notation

JSON

- Lightweight text format for exchanging structured data:
  - Based on a complex data model
  - Similar to XML
- **Fundamental** concepts:
  - Object : unordered set of *name : value* pairs
  - Array : ordered set of values
  - Value : member of the String, Boolean, Object or Array set

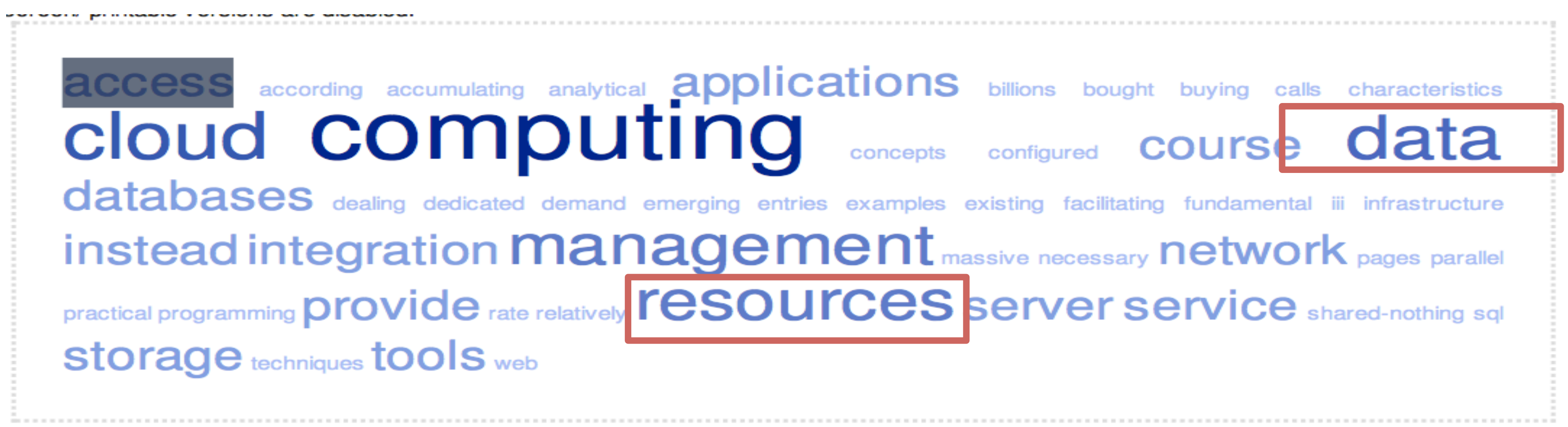
JSON document

Example



```
{
  "type": "person",
  "name": "Darth Vader",
  "age": 63,
  "headware": ["Helmet", "Sombrero"],
  "dark_side": true,
  "weapons": {
    "right_arm": "light_saber",
    "left_arm":  null
  }
}
```

Course tag cloud



screen. printable versions are disabled.

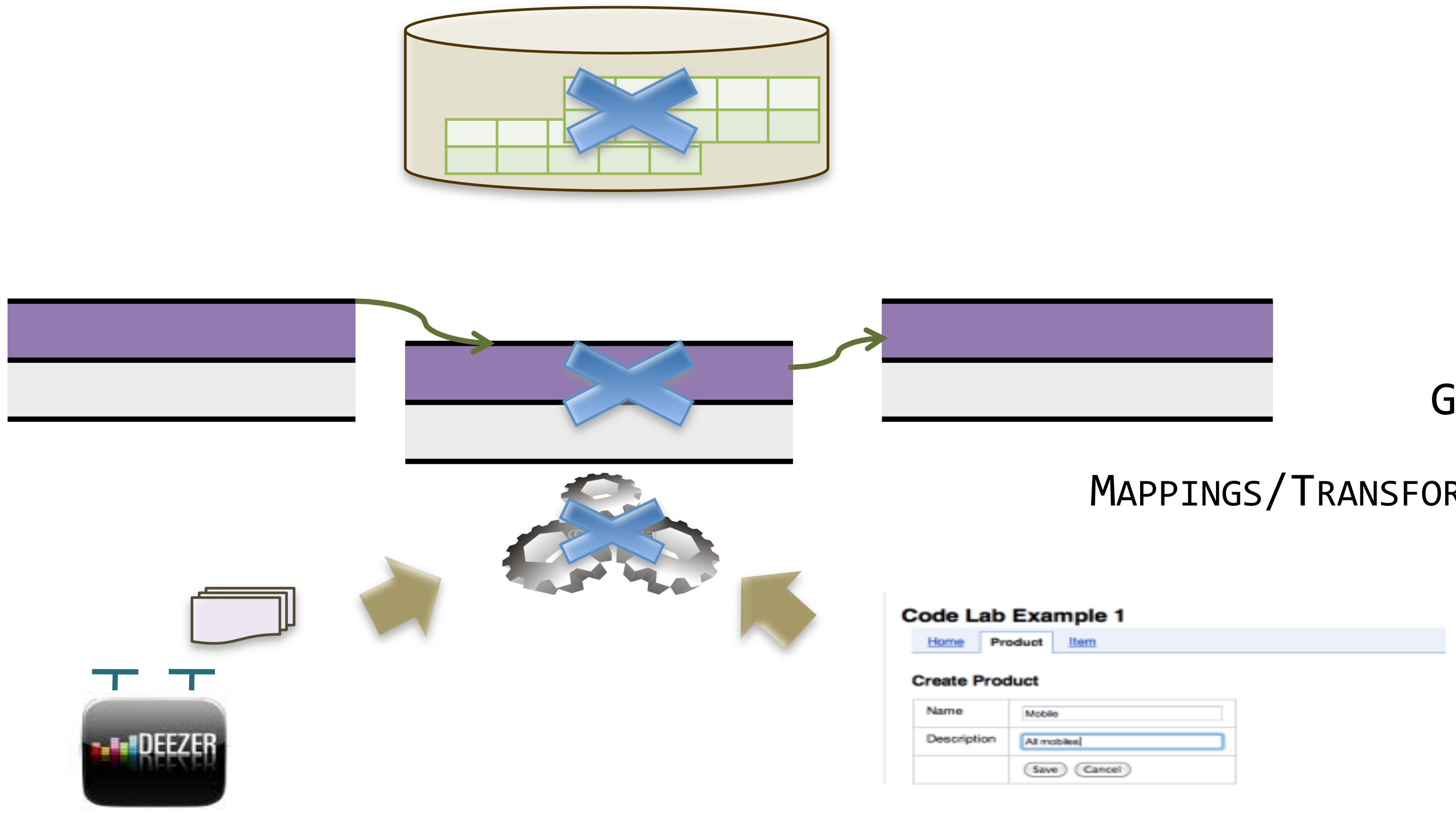
access according accumulating analytical applications billions bought buying calls characteristics cloud computing concepts configured course data databases dealing dedicated demand emerging entries examples existing facilitating fundamental iii infrastructure instead integration management massive necessary network pages parallel practical programming provide rate relatively resources server service shared-nothing sql storage techniques tools web

DESIGN AND BUILDING PROCESS PRINCIPLES

DB: without schema

- Several Web applications are document oriented. When data requirements change:
  - Database schema must change and this has implications in the servers involved
  - Document oriented databases combine old and new document versions

Populating the database



Code Lab Example 1

Home Product Item

Create Product

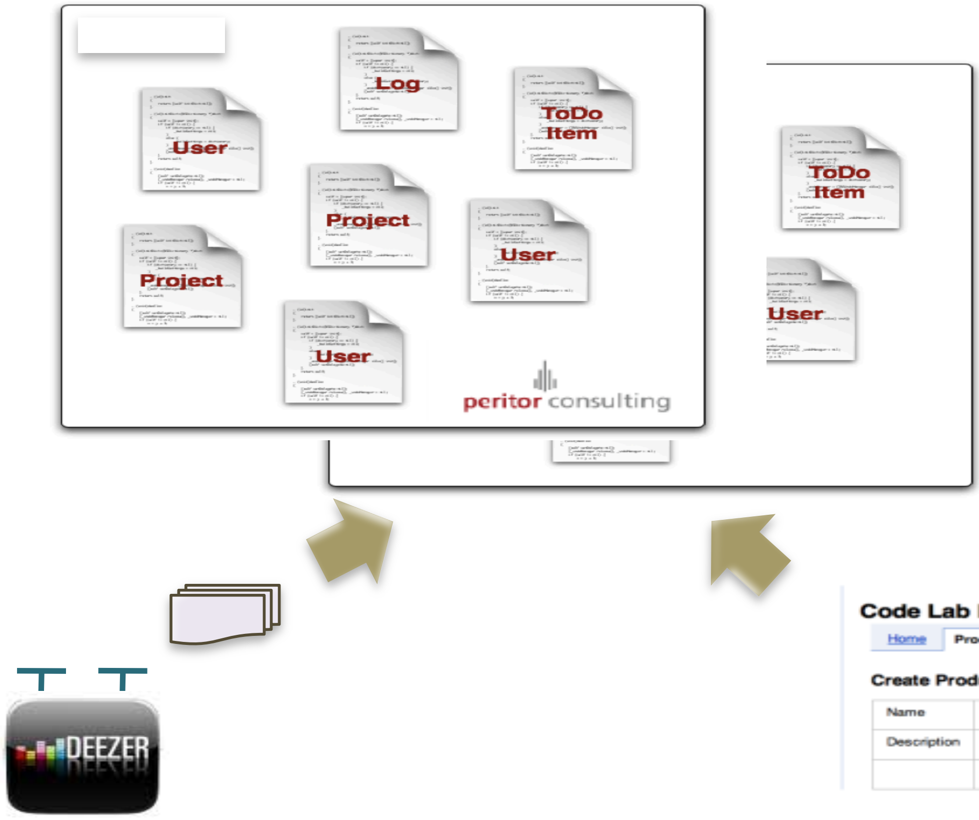
Name	Mobile
Description	All mobiles
	Save Cancel

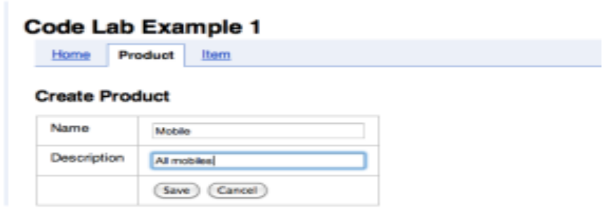
Schema

GAV/LAV

Mappings/Transformation

Populating the database



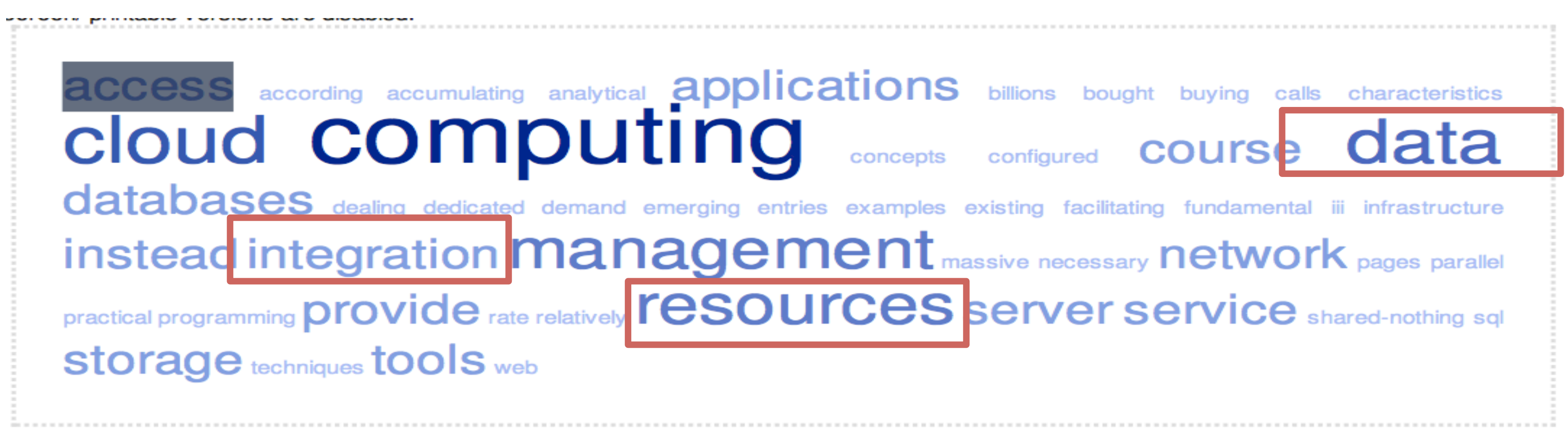

Code Lab Example 1

Home | Product | Item

Create Product

Name	Mobile
Description	All mobiles
	Save Cancel

Course tag cloud



screen. printable versions are disabled.

access according accumulating analytical applications billions bought buying calls characteristics cloud computing concepts configured course data databases dealing dedicated demand emerging entries examples existing facilitating fundamental iii infrastructure instead integration management massive necessary network pages parallel practical programming provide rate relatively resources server service shared-nothing sql storage techniques tools web

EXTERNAL AND INTERNAL DATA MANAGEMENT PRINCIPLES

View model (1)

- For querying data in key-value storage models:
  - The key used for storing data must be specified
  - This is a problem when various registers are involved
- Views are used for creating a definition for indexing and selecting information stored in the database

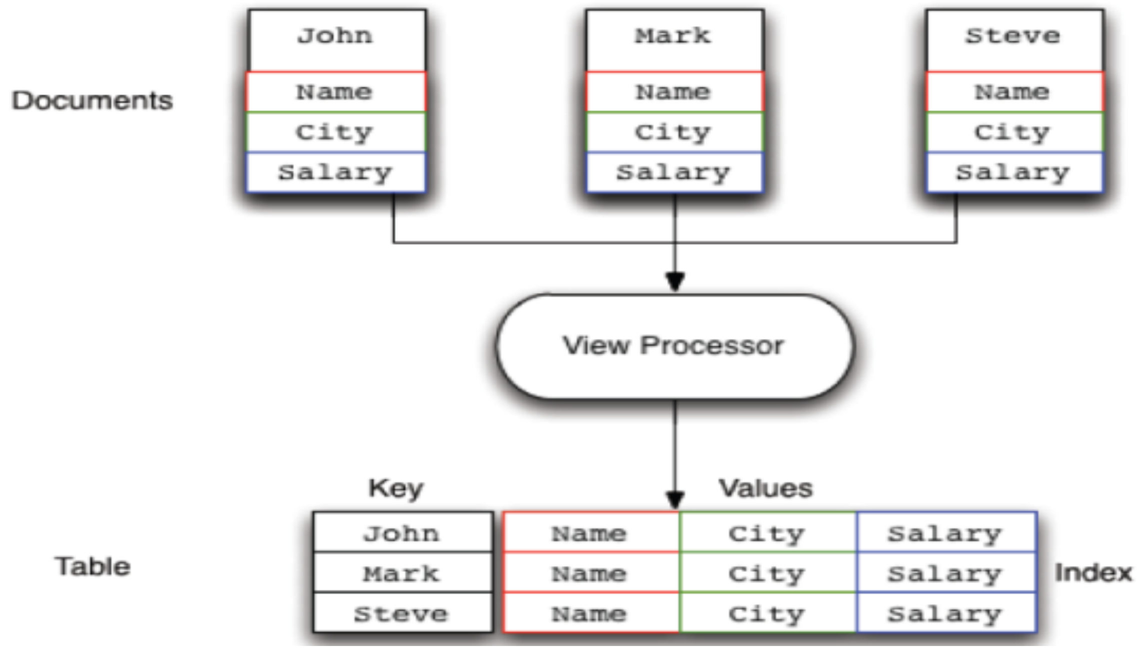
View model (2)

- It is similar to the notion of view in SQL:
  - Add structure to semi-structured documents
  - Can be created dynamically on demand
  - Do not touch the document involved
  - Several views can be associated to the same document
- Views, as documents, are mapped to individual files:
  - Can be stored in different discs for increasing performance
  - A view only has one associated B-tree

View: basic concepts (1)

- A view:
  - Uses the JSON structure for stored documents
  - Specifies the attributes to be queried
  - Builds an index on the selected information
- The result is associated to the document that is being queried:
  - Built iterating data selection specified by the view
  - Is stored for future queries and is updated on every access

View: basic concepts (2)



«*memcached*»

- «*memcached*» is a memory management protocol based on a cache:
  - Uses the key-value notion
  - Information is completly stored in RAM

- «*memcached*» protocol for:
  - Creating, retrieving, updating, and deleting information from the database
  - Applications with their own «*memcached*» manager (Google, Facebook, YouTube, FarmVille, Twitter, Wikipedia)

Local function (1)

- Views definitions are stored in design documents and their ID is prefixed by `"_design"`:
  - A design document can have zero or n views
  - A bucket can have zero or n design documents
- Views in a design document can be updated incrementally when they are accessed:
  - Are updated using the last version of the documents
  - Can be updated automatically as a result of executed operations

Local function (2)

- A complete view can be recomputed when its definition is modified
- Each design document hasan index created with the information necessary for building all its views
- If queryed documents have a “not-freshed” flag, the content of the view will be computed before the pending updates

B-Tree: physical level

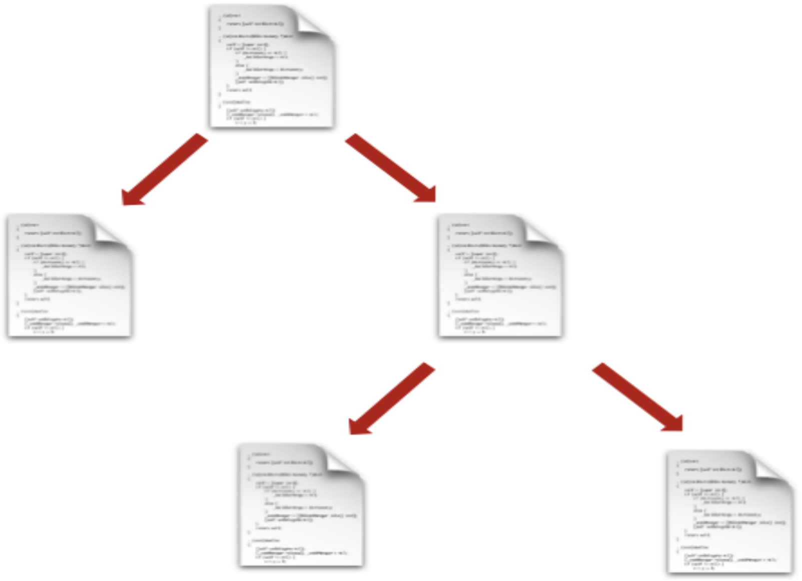
Append only

Concurrency (MVCC)

Crash resistant

Hot backups

Compaction



peritor consulting

Storage on disc (1)

- For efficiency reasons, information is stored using the RAM:
  - Work information is in RAM in order to answer to low latency requests
  - Yet, this is not always possible and desirable

➢ The process of moving data from RAM to disc is called "*eviction*”; this process is configured automatically for every bucket

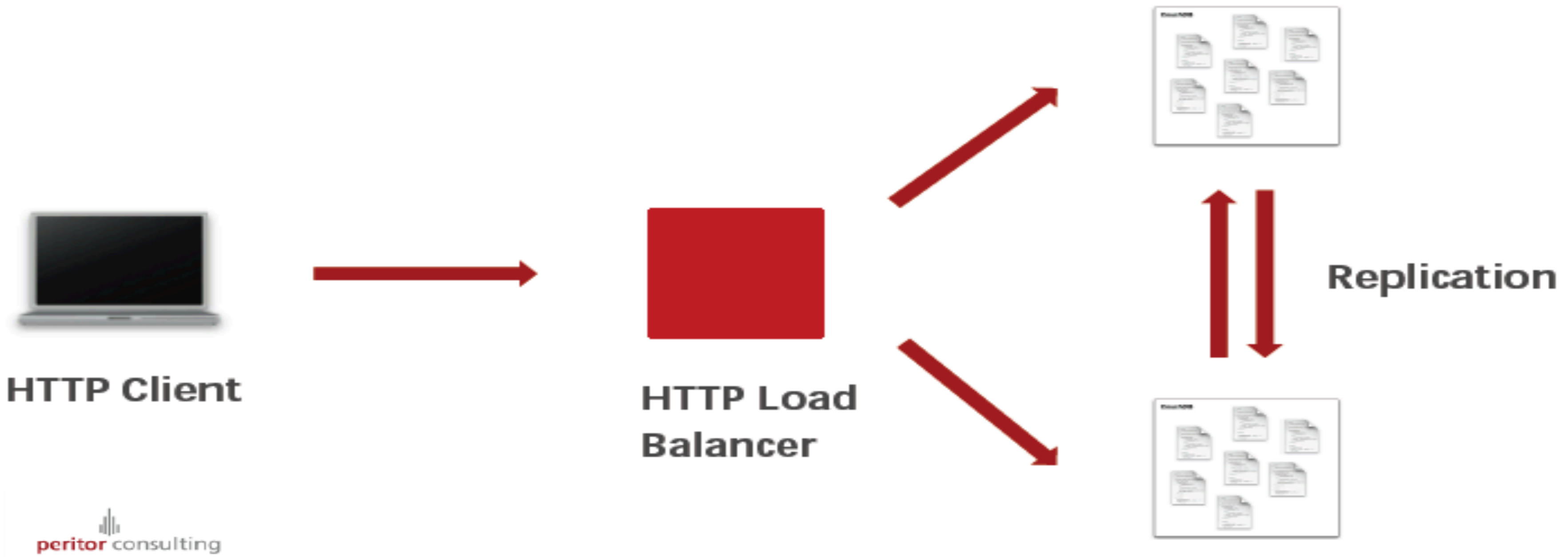
Storage on disc (2)

- NoSQL servers support the storage of key-value pairs on disc:
  - Persistency–can be executed by loading data, closing and reinitializing it without having to load data from another source
  - Hot backups– loaded data are sotred on disc so that it can be reinitialized in case of failures
  - Storage on disc– The disc is used when the quantity of data is higher thant the physical size of the RAM, frequently used information is maintained in RAM and the rest es stored on disc

Storage on disc (3)

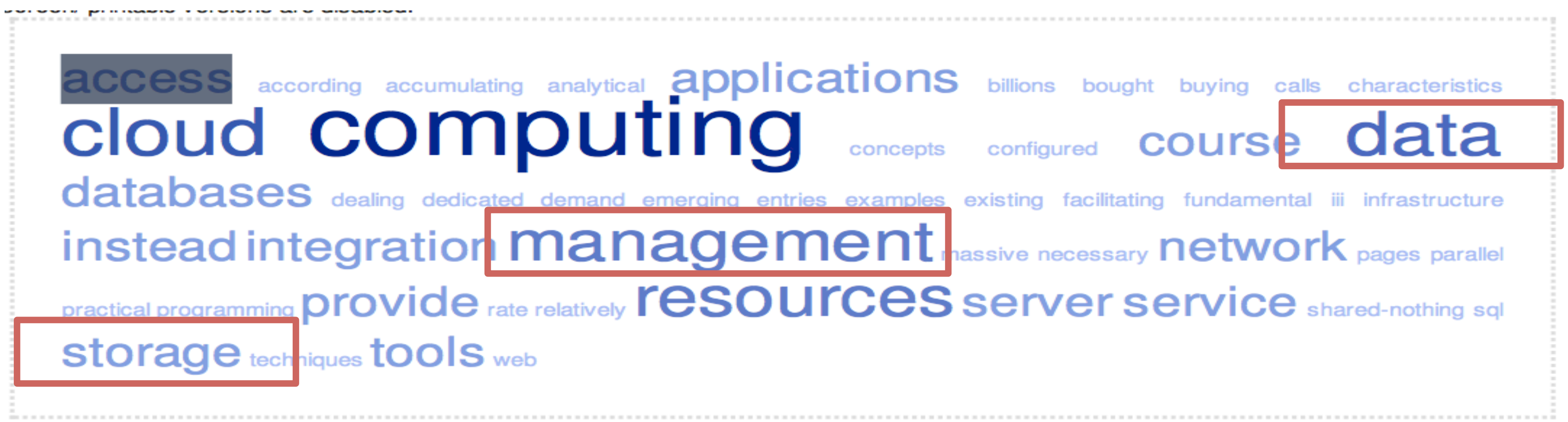
- Strategies for ensuring:
  - Each node maintains in RAM information on the key-value pairs it stores. Keys:
    - may not be found, or
    - they can be stored in memory or on disc
  - The process of moving information from RAM to disc is asynchronous:
    - The server can continue processing new requests
    - A queue manages requests to disc
    - ➢ In periods with a lot of writing requests, clients can be notified that the server is termporaly out of memory until information is evicted

Replication



peritor consulting

Course tag cloud



access according accumulating analytical applications billions bought buying calls characteristics cloud computing concepts configured course data databases dealing dedicated demand emerging entries examples existing facilitating fundamental iii infrastructure instead integration management massive necessary network pages parallel practical programming provide rate relatively resources server service shared-nothing sql storage techniques tools web

ARCHITECTURE PRINCIPLES

+ Only NoSQL? : Principles (1)

Elasticity

- High availability:
  - Profits the advantages of a cluster architecture where all nodes are identical
  - Automatically creats information replicas for tolerating faults
- Hability to expand along multiple servers (scalability):
  - Data are automatically redistributed along the cluster
  - Changes the cluster capacities adding or deleting nodes and balancing the load

Only NoSQL? : Principles (2)

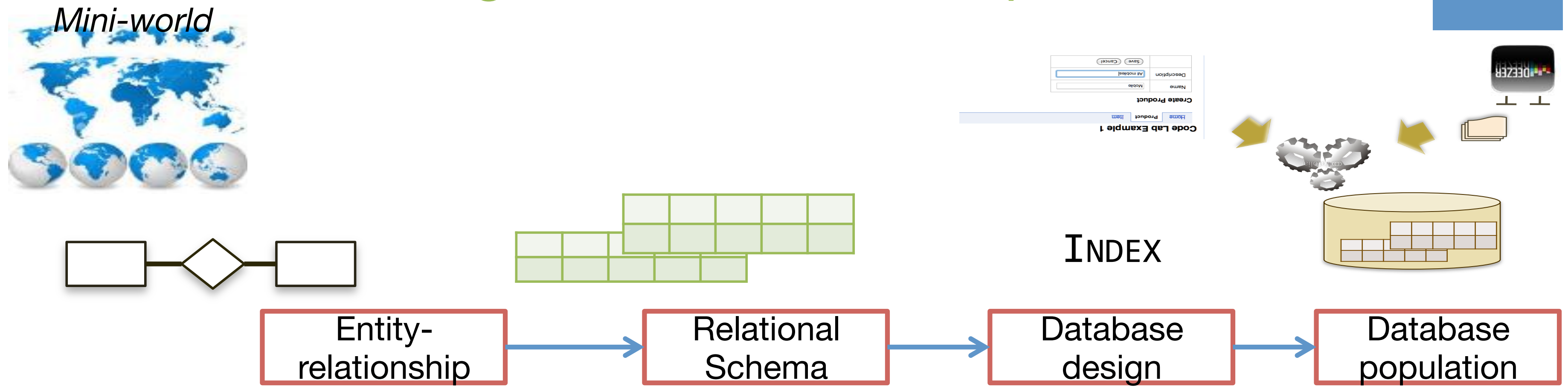
- Simplicity:
  - Does not require to create or manage databases, tables and schemata
  - Data distribution on different nodes without having to normalize or share data
- Efficiency:
  - «*in-memory*» database
  - Performance and latency almost deterministic without having to deal with load pics

Agenda

- NoSQL overview
- Principles: data model, design and building process, external and internal data management and architecture
- Putting NoSQL on the data management systems arena
- Hand on …

+ NoSQL vs. RDBMs

Relational DB design and construction process



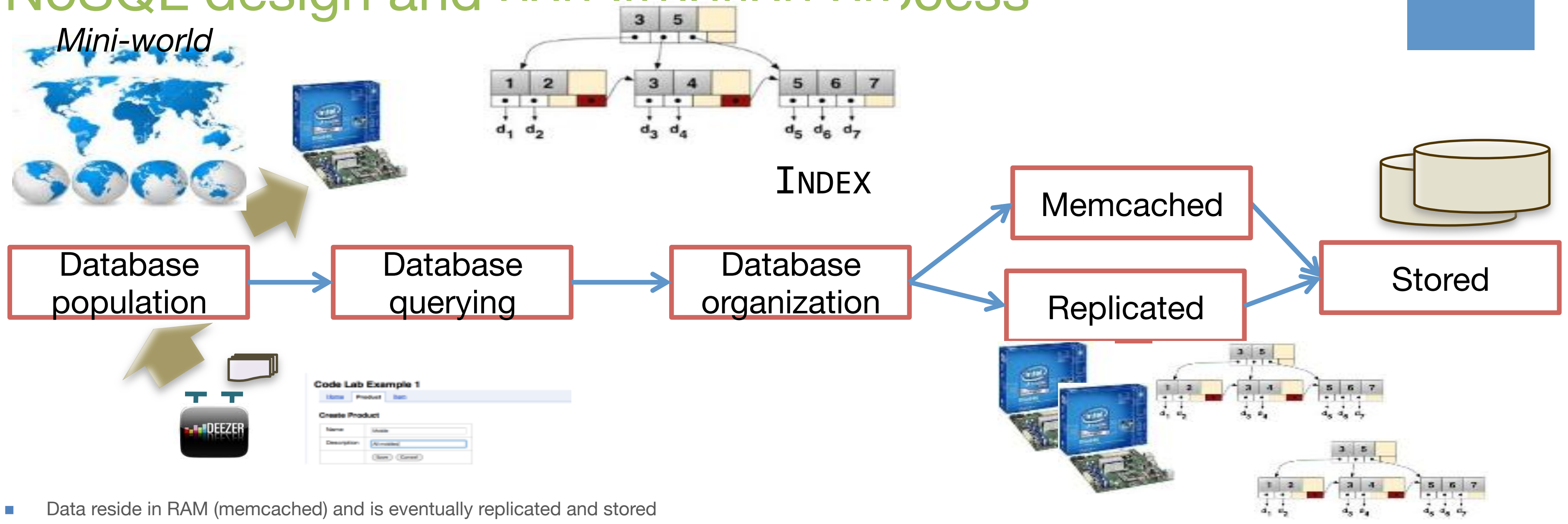
- Data convey to the predefined structure of the database
- The database design is done by an administrator according to the possible queries (mathematically defined in advance)
- Concrete well defined architecture (ANSI/SPARC)

NoSQL vs. RDBMs

- RDBMS maintain a distributed consistency and this makes it difficult for scaling objectives:
  - Before writting in a node, it executed synchronization protocols → 2PC
  - Clients' requests are blocked until all nodes reach an agreement and data are validated by all nodes
  - As more nodes are implied, there is more communication overhead and more time is required for reaching an agreement

+ NoSQL vs. RDBMs

NoSQL design and construction process



- Data reside in RAM (memcached) and is eventually replicated and stored
- Querying = designing a database according to the type of queries / map reduce model
- "On demand" data management: the database is virtually organized per view (external schema) on cache and some view are made persistent
- An elastic easy to evolve and explicitly configurable architecture

CouchDB in the wild…

http://wiki.apache.org/couchdb/CouchDB_in_the_wild