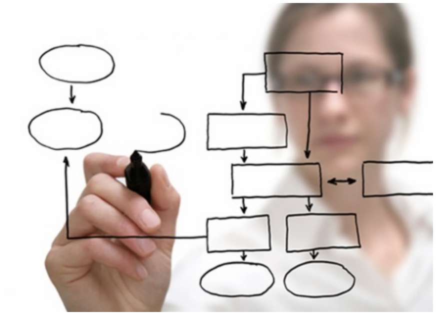
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT



Distributed Data Management

Lecture 5: Document Oriented DBs; MongoDB

RDBMS Limitations → NOSQL Databases

In the previous week…

❑ Relational Database Management System (RDBMS) problems:

1. Impedance mismatch.
2. Rigid schema.
3. Not suitable for scaling out.

❑ NOSQL → Alternative to do very well these 3 things. 4 families:

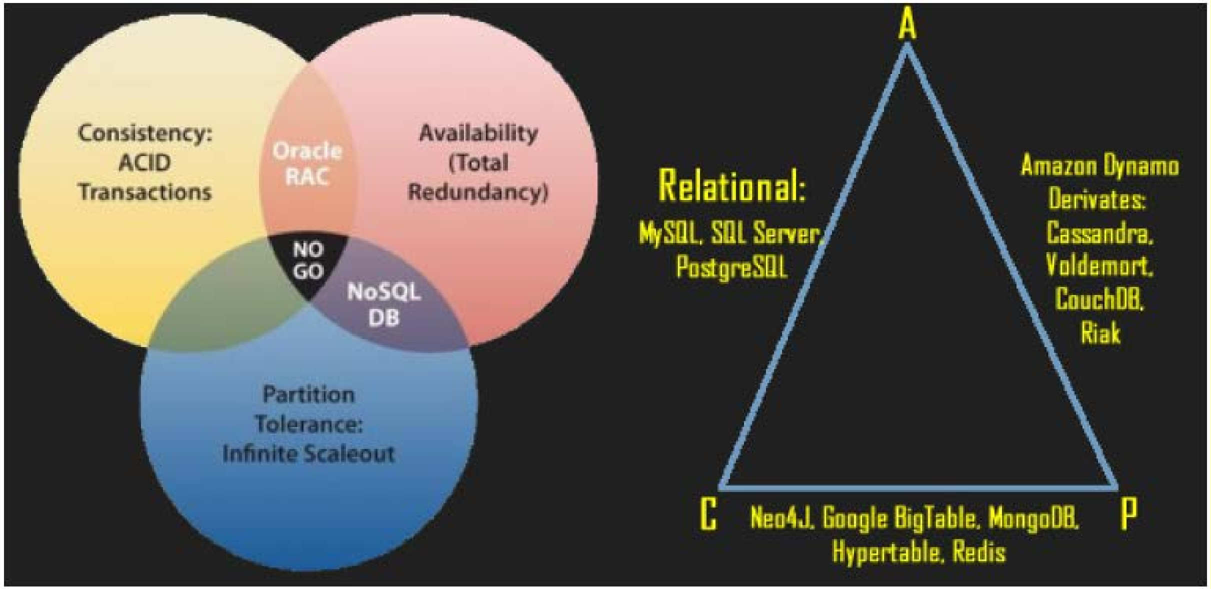
- Pure Key Value-based (e.g., Riak, Redis, Amazon Dynamo DB).
- Document Oriented-based (e.g., MongoDB, CouchDB).
- Column-based (e.g., Cassandra, HBase).
- Graph-based (e.g., Neo4J).

RDBMS Limitations → NOSQL Databases

In the previous week…

❑ NOSQL main problems:

- Lack of expressive queries (in particular, no joins).
- Too many options to pick from.
- Because of the CAP theorem, if they decide to be Partition Tolerant, they have to give up either Consistency or Availability.


Consistency: ACID Transactions
Oracle RAC
Availability (Total Redundancy)
NO GO
NoSQL DB
Partition Tolerance: Infinite Scaleout
A
Relational: MySQL, SQL Server, PostgreSQL
Amazon Dynamo Derivates: Cassandra, Voldemort, CouchDB, Riak
C
Neo4J, Google BigTable, MongoDB, Hypertable, Redis
P

RDBMS Limitations → NOSQL Databases

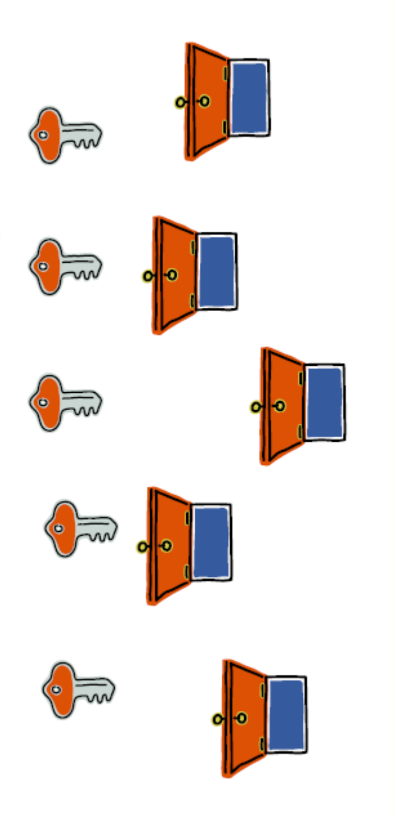
Key	Value
Key$_1$	Value$_1$
…	…
Key$_n$	Value$_n$

In the previous week…

❑ Pure Key-Value NOSQL databases.

❑ Advantages:

- Easy interface: Add, get, delete.
- Very light.
- No complex relations.
- Easily scalable.
- Lack of query language.
- Fast operations.
- Data abstraction in both key and value.

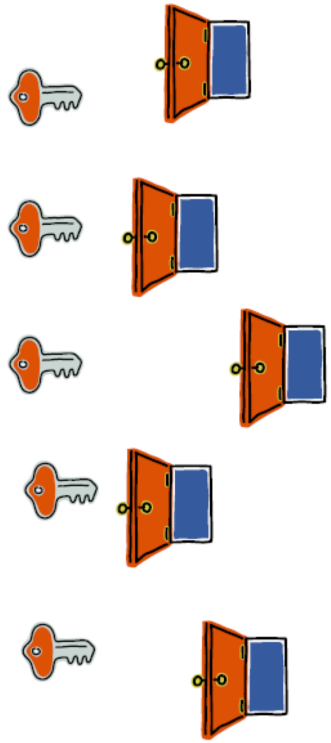


RDBMS Limitations → NOSQL Databases

In the previous week…

- □ Pure Key-Value NOSQL databases.
- □ Disadvantages:
  - No notion about the value returned, up to user to interpret it.
  - No queries can be performed in the value!
  - No update operations!

Key	Value
Key$_1$	Value$_1$
…	…
Key$_n$	Value$_n$



Outline

1. Document Oriented vs. Pure Key Value.
2. MongoDB Main Concepts.
3. MongoDB overcoming RDBMS limitations.
   - I. Impedance Mismatch.
   - II. Rigid Schema.

Outline

1. Document Oriented vs. Pure Key Value.
2. MongoDB Main Concepts.
3. MongoDB overcoming RDBMS limitations.
   I. Impedance Mismatch.
   II. Rigid Schema.

Document Oriented vs. Pure Key Value

- ❑ Document oriented-based are the most flexible and popular NoSQL databased.
- ❑ They arguably belong to the key-value family:
  - o Individual field data is still stored in a key-value format.
  - o However, values have some structure → document format.
  - o A value / document is encoded in a JSON (plain text) or BSON (binary version of JSON):
    - ▪ Accessible by most popular programming languages.
    - ▪ Very natural and flexible to be queried.

- ❑ JSON / BSON Queries Flexibility → You can query on each field (even on multiple fields at the same time).

Document Oriented vs. Pure Key Value

- ❑ Document-oriented vs. pure key value-based main advantages:
  - The interface is not as small and neat, but the interface language is still relatively small and easy.
  - Still no complex relations → No joins.
  - Still very good at scalability → Replication + Sharding (because of aggregation – more later).
  - It can have a query language, but usually based on general purpose languages (e.g., JavaScript, Python).
  - Although it cannot be as fast as pure key value-based, state-of-the-art data analytics can be applied for fast querying.
  - The lack of value abstraction supports much more complex queries (sorting, aggregation, filtering, etc).

Document Oriented vs. Pure Key Value

- ❑ Document-oriented vs. pure key value-based main limitations:
  - There is a notion of the value being returned (encoded in JSON / BSON format, so it is possible to access each piece of it).
  - Queries can be performed on the value (again, JSON / BSON format).
  - Update operations are definitely allowed.

Document Oriented vs. Pure Key Value

- Document-oriented **share** some concepts with RDBMS:
  - Setup consists of a database server and n ≥ 0 clients connected to it.
  - A database in document-oriented consists of c ≥ 0 collections (in RDBMS, t ≥ 0 relational tables).
  - A collection in document-oriented consists of d ≥ 0 documents (in RDBMS a relational table on t ≥ 0 tuples).
  - A document consists of f ≥ 1 *key* fields, each with an associated *value* (in RDBMS an entry consists of c ≥ 1 columns, each of them with an associated value).
  - Each document has a unique key value within the collection (in RDBMS each entry has a unique primary key). Often created automatically by the database.

Document Oriented vs. Pure Key Value

- Document-oriented **has these main differences** w.r.t. RDBMS:
  - Collections are schema-less:
    - Different documents of a same collection can have different fields).
    - Two documents with a same field can have different datatypes values for them.
    - However, implied schema means documents often map directly to classes or structs in high-level programming languages such as Python or C++ (i.e. implied schema).

Outline

1. Document Oriented vs. Pure Key Value.
2. MongoDB Main Concepts.
3. MongoDB overcoming RDBMS limitations.
   I. Impedance Mismatch.
   II. Rigid Schema.

MongoDB Main Concepts

MongoDB is the most popular document oriented DB.

□ It is a cross-platform database, which provides:

- High performance.
- High availability.
- Easy scalability.

□ It supports db ≥ 0 databases…

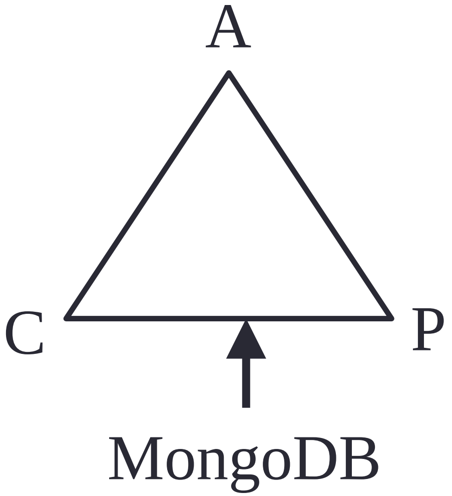
…each of them with c ≥ 0 collections…

…each of them with d ≥ 0 documents…

…each of them with f ≥ 1 key-value pairs (fields and their values).

MongoDB Main Concepts

MongoDB is the most popular document oriented DB.



- CAP theorem → MongoDB chose CP.
  - Other document oriented such as CouchDB chose AP.
- That’s why it is interesting that it claims to be highly available (obviously not fully)!
- We will see how it achieves this full consistency + high availability via → Replication + Sharding.

MongoDB Main Concepts

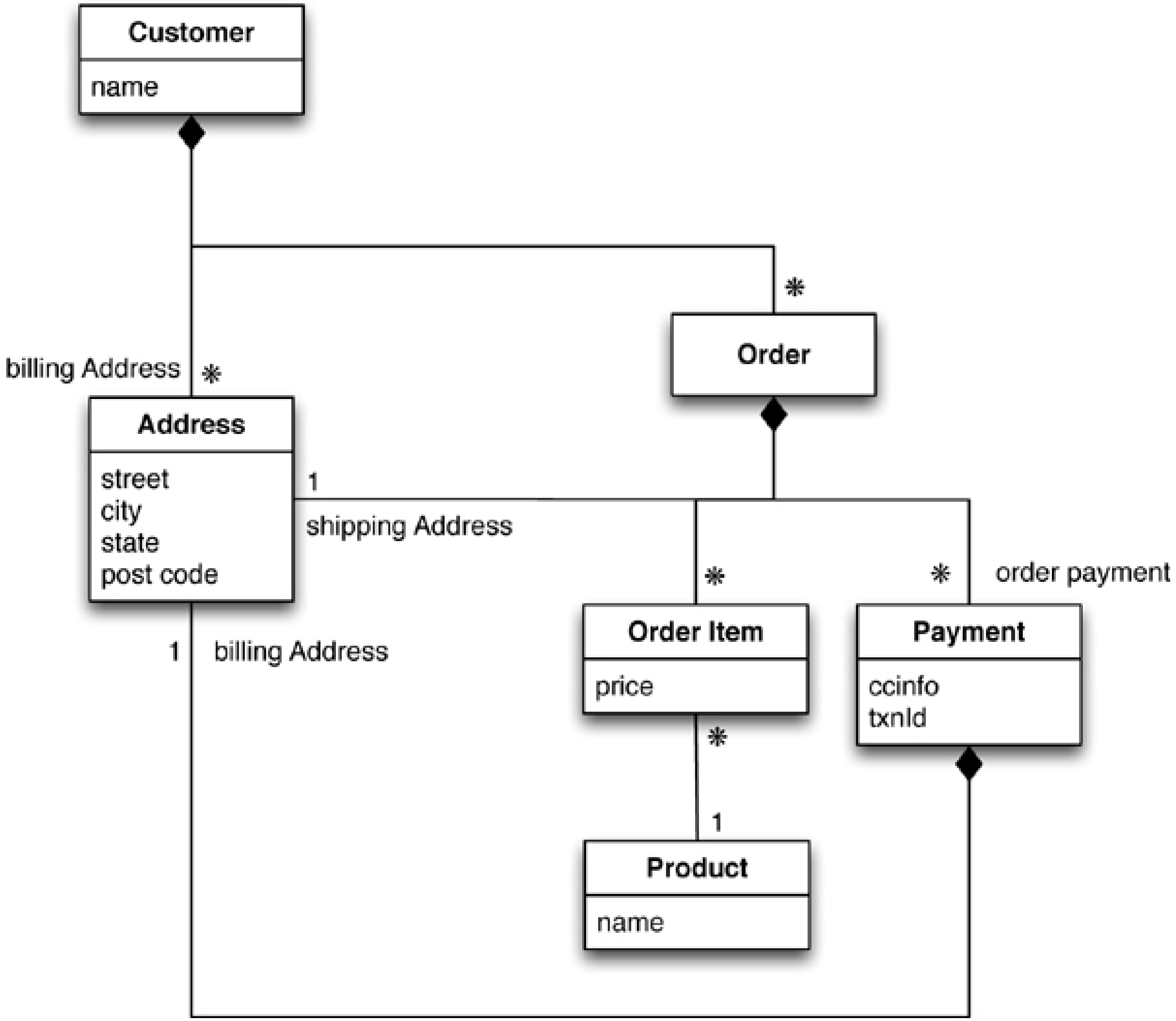
Data Stored → JSON-based documents:

- ❑ Key – value pairs.
  - o Key → String;
  - o Value → String, int, list of elements…and embedded JSON (i.e. sub-documents).

```
{ "name": "John",
  "age": 30,
  "sports": ["football", "hurling"],
  "embedded": {"birthday" : "September",
                              "pets" : 2 }
}
```

MongoDB Main Concepts

Example: Invoice document made up of customer, order, product, etc.



MongoDB Main Concepts

Example: Invoice document made up of customer, order, product, etc.

```
{
 "customer": {
 "id": 1,
 "name": "Martin",
 "billingAddress": [{"city": "Chicago"}],
 "orders": [
   {
     "id":99,
     "customerId":1,
     "orderItems":[
     {
     "productId":27,
     "price": 32.45,
     "productName": "NoSQL Distilled"
     }
   ],
   "shippingAddress":[{"city":"Chicago"}],
   "orderPayment":[
     {
     "ccinfo":"1000-1000-1000-1000",
     "txnId":"abelif879rft",
     "billingAddress": {"city": "Chicago"}
     }],
   }]
 }
}
```

Aggregation into single document (this will help with performance and sharding but can lead to duplication of data)

MongoDB Main Concepts

Collection: Set of documents

Since documents are schema-less and can vary widely in structure…

❑ Why do we need to put them into collections?

1. Easy to manage for everybody (e.g., collections of blogs, addresses, followers, etc).
2. It’s faster to get specific document types by sorting them in a collection.
3. Data locality: Similar data resides together.
4. Collections are easy to index!

MongoDB Main Concepts

Querying a Collection:

❑ Highly expressive general purpose programming language:

- E.g. Java, Python (we have seen examples in the lab and more to come).

❑ Expressive query commands:

- Create / Check / Delete database & collection.
- Insert / Update / Delete documents of collection.
- Set index on document fields.
- Query documents by relational / logic / existence / regex conditions.
- Perform aggregations.
- Conditionals / Ranges / Limits / Skips / Sorts.
- Analytics.

Outline

1. Document Oriented vs. Pure Key Value.
2. MongoDB Main Concepts.
3. MongoDB overcoming RDBMS limitations.
   - I. Impedance Mismatch.
   - II. Rigid Schema.

MongoDB Overcoming RDBMS Limitations

❑ So far we have talked about the 3 pillars that trigger the development of NoSQL databases:

- o Impedance mismatch of RDBMS.
- o Rigid Schema of RDBMS.
- o Impossible to Scale out for RDBMS.

❑ In lab 1 we used saw how

1. Impedance mismatch → Let’s see how easy is for a language like Python to use JSON and to query MongoDB.
2. Schema-less → Let’s see how a collection supports different fields and different data types.
3. Scale out → Let’s see how MongoDB replication + sharding support a distributed cluster for a given collection.

Thank you for your attention!