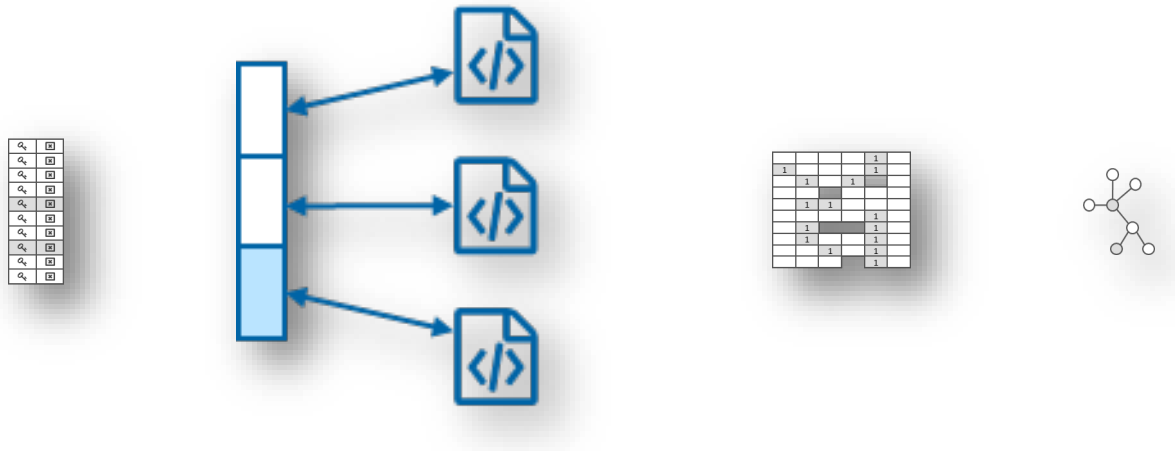
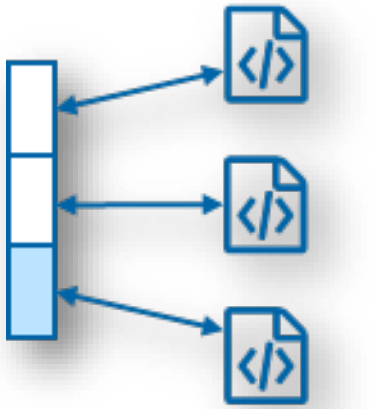
NoSQL

Document-oriented



Content

- Introduction
- MongoDB
- Connect
- Query

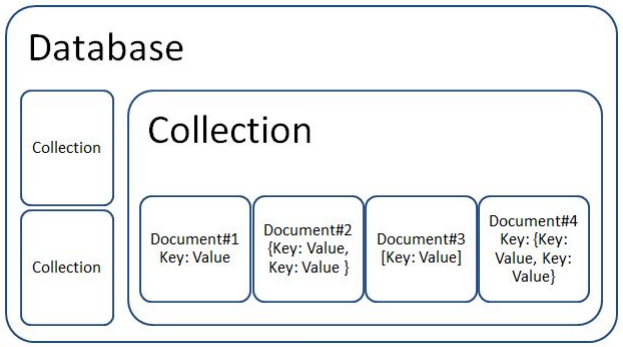


NoSQL

Introduction

Basics

- Data is stored in documents with a given data format
- Typical data formats are JSON and XML
- Indexing based on document properties (filenames are irrelevant)
- Documents containing similar content are grouped in collections
- Document structure is not fixed
- Allows defining rules based on the content



Customer Document

```
"customer" =
{
   "id": "Customer:1",
   "firstName": "John",
   "lastName": "Wick",
   "age: 25,
   "address": {
        "country": "US",
        "city": "New York",
        "state": "NY"
        "street": "21 2nd Street",
      },
   "hobbies": [ Football, Hiking ],
   "phoneNumbers": [
      {
        "type": "Home",
        "number": "212 555-1234"
      },
      {
        "type": "Office",
        "number": "616 565-6789"
      }
    ]
}
```

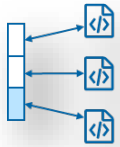
Introduction

Database-Management Systems

Key-Value

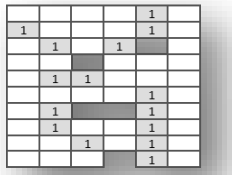
Apache CouchDB
ArangoDB
BaseX
Clusterpoint
Couchbase
Cosmos DB
IBM Domino
MarkLogic
OrientDB
Qizx
RethinkDB

Document



Aerospike
Apache Ignite
ArangoDB
Couchbase
Dynamo
FairCom c-treeACE
FoundationDB
InfinityDB
MemcacheDB
MongoDB
MUMPS
Oracle NoSQL Database
OrientDB
Redis
Riak
Berkeley DB
SDBM/Flat File dbm
ZooKeeper

Column



Accumulo
Cassandra
Druid
Hbase
Vertica

Graph

AllegroGraph
ArangoDB
InfiniteGraph
Apache Giraph
MarkLogic
Neo4J
OrientDB
Virtuoso

Content

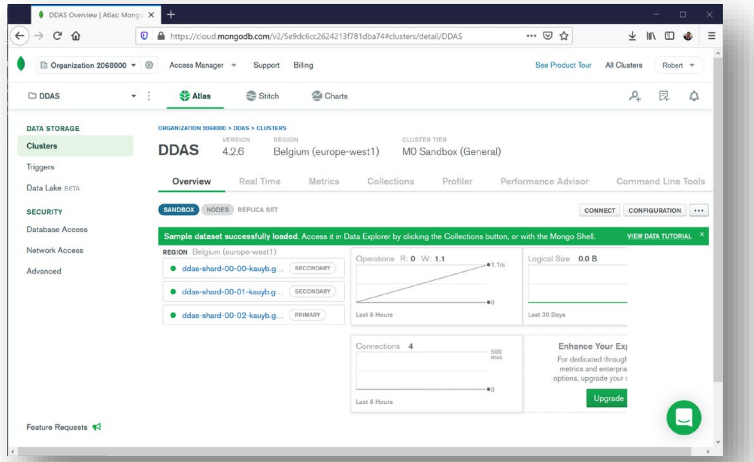
- Introduction
- MongoDB
- Connect
- Query

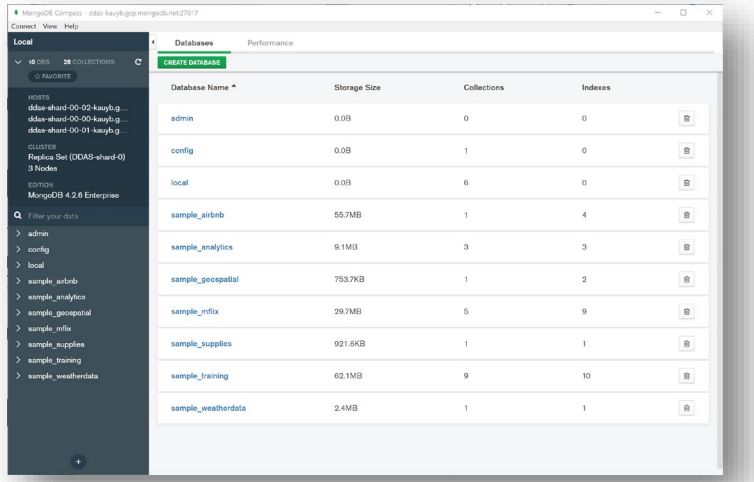
mongoDB

Overview

- document-oriented database
- uses JSON-like documents
- server-client setup
- official drivers for major programming languages
- MongoDB Compass is native GUI

We are not looking into mongoDB very thoroughly. It will simply serve for a hands-on example for document-oriented databases





mongoDB

Python and R

- Python → PyMongo

```
# import mongoDB module
import pymongo
```

- R → mongolite

```
# load mongoDB package
library(mongolite)
```

Content

- Introduction
- MongoDB
- Connect
- Query

Connect

The first thing to do in Python to access a mongoDB database is to establish a connection to the server.

```
# import mongoDB module
import pymongo

# create a new connection to the server
myConn = pyymongo.MongoClient('server url')

# select a specific database on the server
database = myConn.databasename
or
database = myConn['databasename']
```

Connect

credentials

To establish a connection to a mongoDB server, the credentials are provided within the server URL:

mongodb+srv://<username>:<password>@<servername>

Here is the URL for our hands-on example:

mongodb+srv://student:MongoDB4DDAS@ddas-kauyb.gcp.mongodb.net

Connect

config.py

config.py

```
# store information
DATABASE = {'server': 'servername',
            'user': 'username',
            'password': 'password'}
```

myCode.py

```
# import
from config import DATABASE
import pymongo

# create a new connection to the mongodB server
myConn = pymongo.MongoClient('mongodb+srv://' + \
                             DATABASE['user'] + ':' + \
                             DATABASE['password'] + '@' + \
                             DATABASE['server'] )
```

Content

- Introduction
- MongoDB
- Connect
- Query

Query

collections

- A collection is a grouping of MongoDB documents. It is the equivalent of an RDBMS table.
- A collection exists within a single database.
- Collections do not enforce a schema. Documents within a collection can have different fields.
- Typically, all documents in a collection have a similar or related purpose.

```
# list all collections in the database
collection_list = database. list_collection_names()

# select a collection in the database
collection = database.collectionname
or
collection = database['collectionname']
```

Query

Insert

To insert a new document simply define a dictionary in python and insert it into the collection

```
# create new document
new_doc = {'key1':value1,
           'key2':value2,
             :
           'keyN':valueN}

# insert new document
doc_id = collection.insert_one(new_doc).inserted_id
```

When a document is inserted a special key, "_id", is automatically added if the document doesn’t already contain an "_id" key.
The value of "_id" must be unique across the collection.

Query

Retrieve

A way of retrieving documents is similar to searching for files with a specific content.

```
# retrieve documents with a given content
documents = collection.find({'some_key':'some_value'})

# simply retrieve everything
documents = collection.find()

# loop through the results
for d in documents:
        print(d)
```

Advanced queries (multiple conditions, sorting, etc.) is out of scope for this module