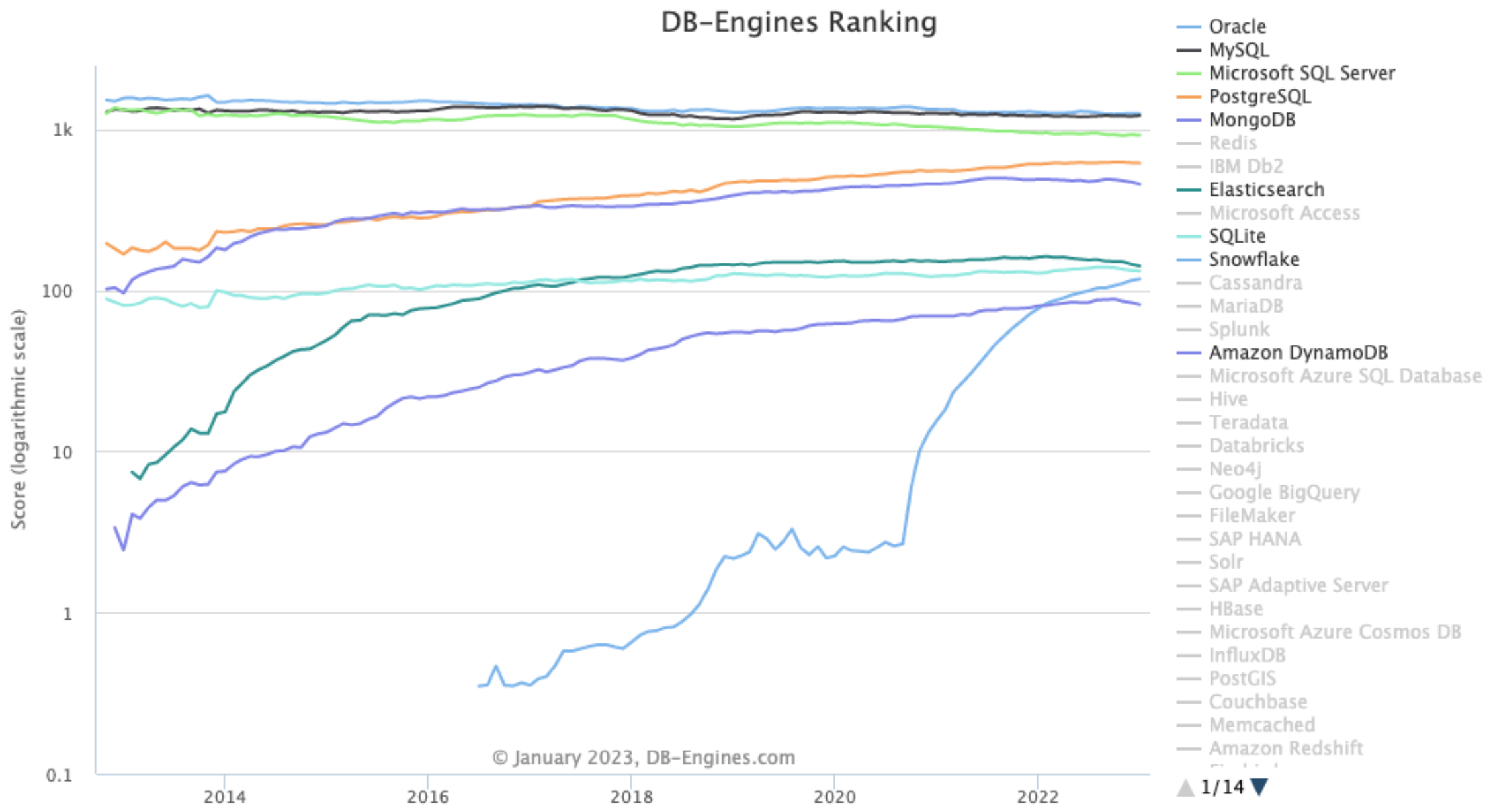
MongoDB

NoSQL Datastore


DB-Engines Ranking
Score (logarithmic scale)
1k
100
10
1
0.1
2014
2016
2018
2020
2022
© January 2023, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
Redis
IBM Db2
Elasticsearch
Microsoft Access
SQLite
Snowflake
Cassandra
MariaDB
Splunk
Amazon DynamoDB
Microsoft Azure SQL Database
Hive
Teradata
Databricks
Neo4j
Google BigQuery
FileMaker
SAP HANA
Solr
SAP Adaptive Server
HBase
Microsoft Azure Cosmos DB
InfluxDB
PostGIS
Couchbase
Memcached
Amazon Redshift
1/14

MongoDB	MySQL
HIGH AVAILABILITY When you need high availability of data with automatic, fast and instant data recovery.	**LOW-MAINTENANCE** If you're just starting and your database is not going to scale much, MySQL will help you in easy and low-maintenance setup.
IN-BUILT SHARDING In future, if you're going to grow big as MongoDB has in-built sharding solution.	**LIMITED BUDGET** If you want high performance on a limited budget.
UNSTABLE SCHEMA If you have an unstable schema and you want to reduce your schema migration cost.	**FIXED SCHEMA** If you've fixed schema and data structure isn't going to change over the time like WikiPedia
NO DBA If you don't have a Database Administrator (but you'll have to hire one if you're going to go BIG).	**HIGH TRANSACTION** If high transaction rate is going to be your requirement (like BBC around 30,000 inserts/minute, 4000 selects/hour)
CLOUD COMPUTING If most of your services are cloud-based, MongoDB is best suitable for you	**DATA SECURITY** If data security is your top priority, MySQL is the most secure DBMS.

UNSTABLE SCHEMA

If you have an unstable schema and you want to reduce your schema migration cost.

FIXED SCHEMA

If you've fixed schema and data structure isn't going to change over the time like WikiPedia

NO DBA

If you don't have a Database Administrator (but you'll have to hire one if you're going to go BIG).

HIGH TRANSACTION

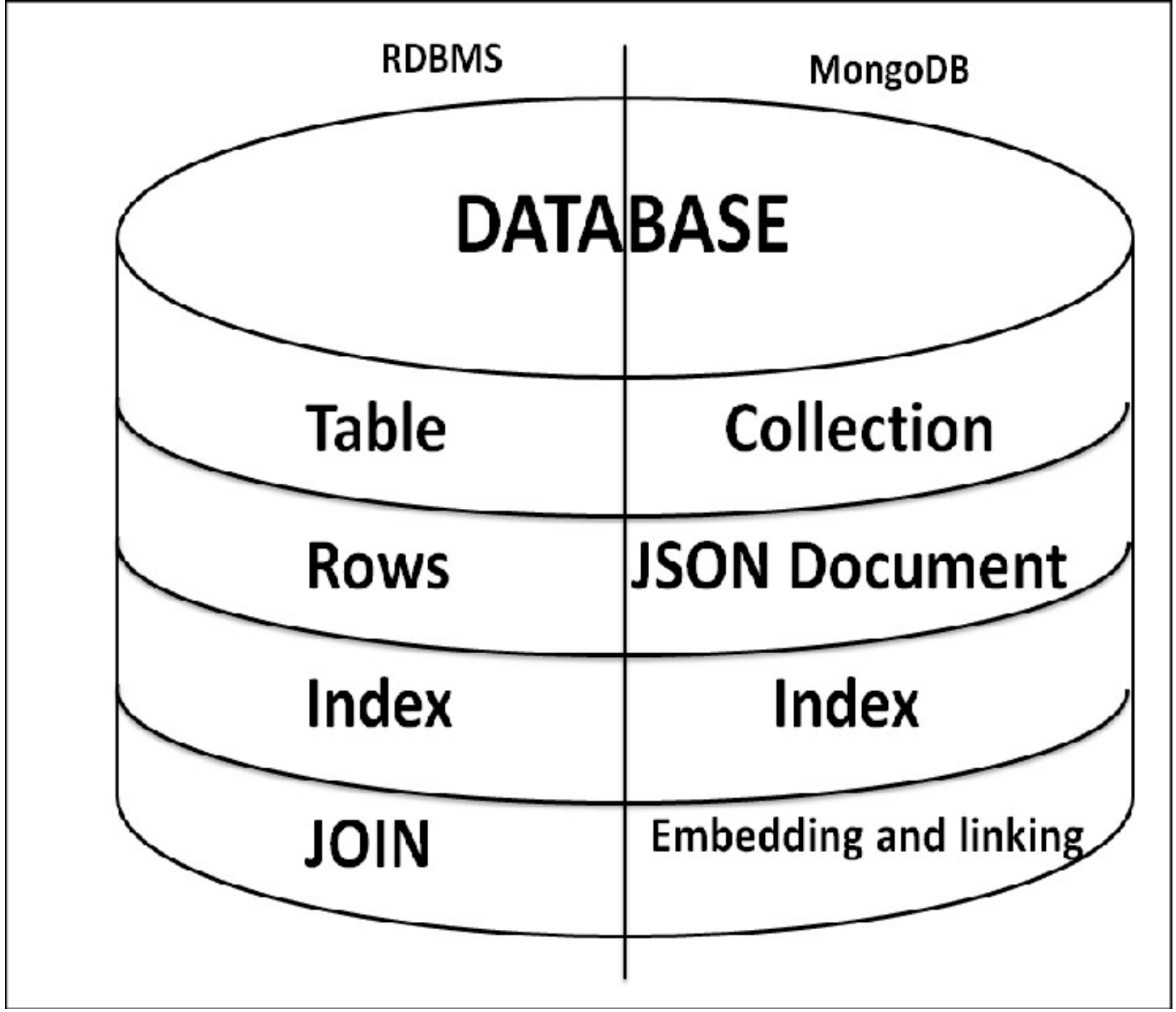
If high transaction rate is going to be your requirement (like BBC around 30,000 inserts/minute, 4000 selects/hour)

CLOUD COMPUTING

If most of your services are cloud-based, MongoDB is best suitable for you

DATA SECURITY

If data security is your top priority, MySQL is the most secure DBMS.


RDBMS
MongoDB
DATABASE
Table
Collection
Rows
JSON Document
Index
Index
JOIN
Embedding and linking

Data Models: Relational to Document

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

— no relation

Car:

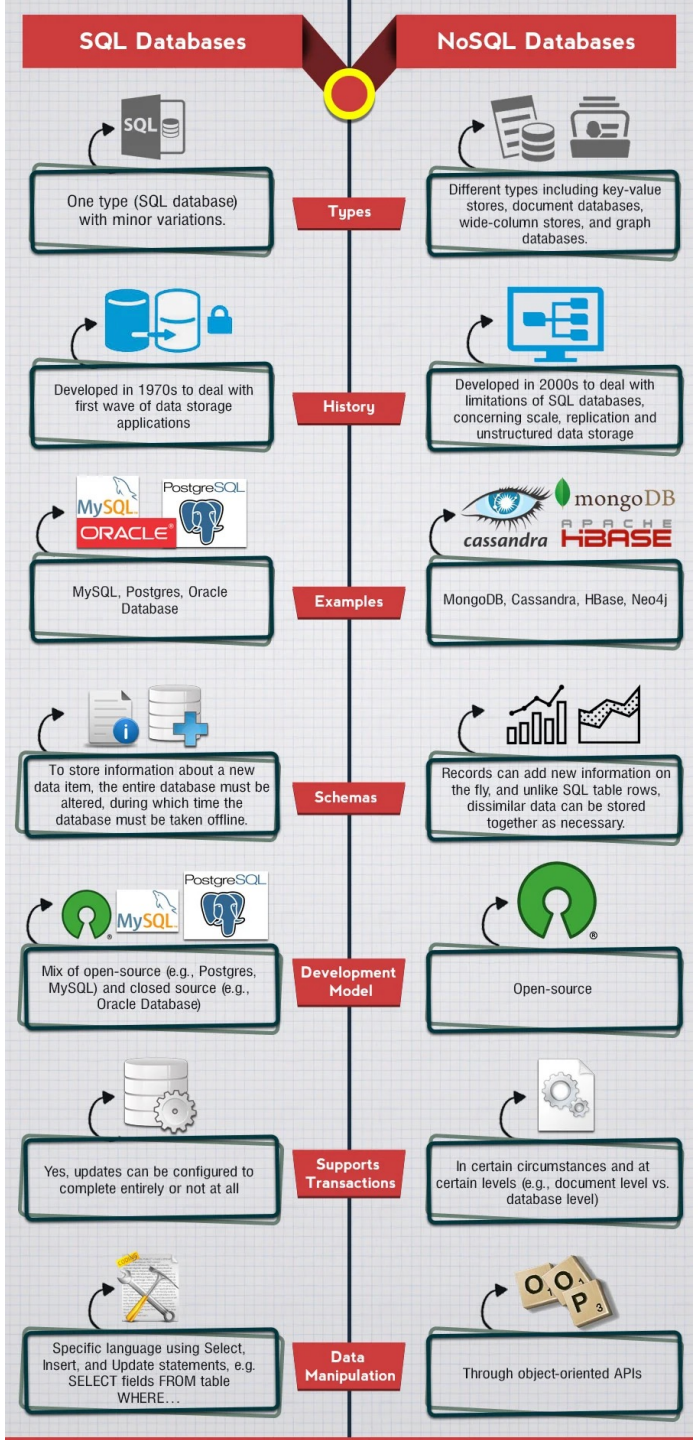
Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

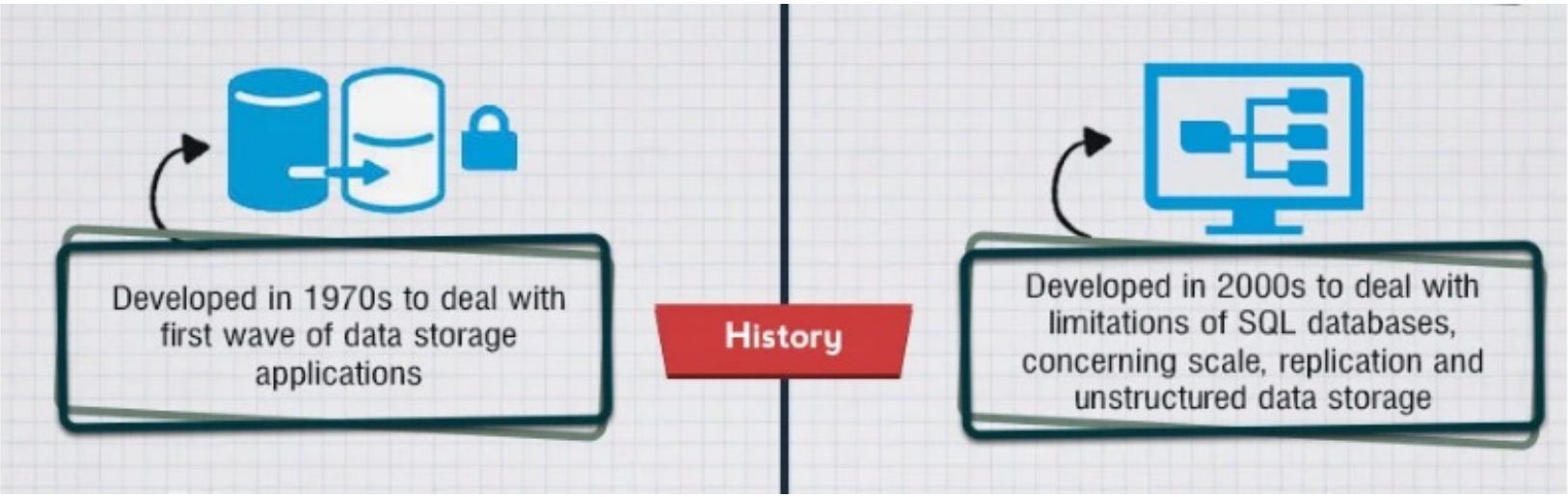
MongoDB Document

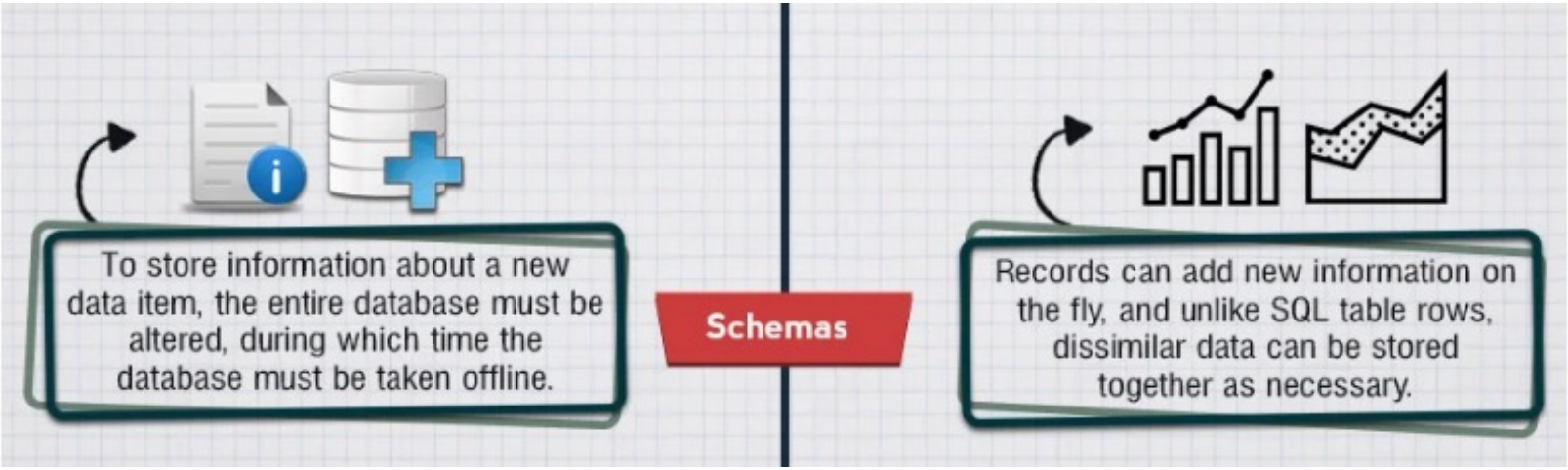
```
{
  first_name: 'Paul',
  surname: 'Miller'
  city: 'London',
  location: [45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```


mongoDB

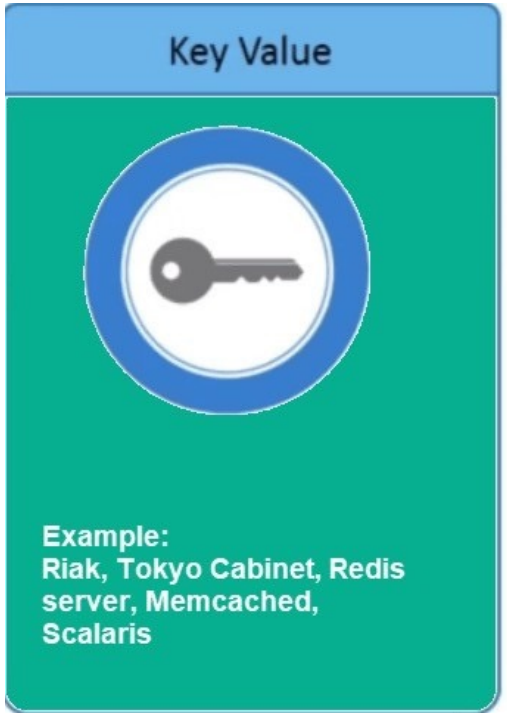
```
{    customer_id : 1,
     first_name : "Mark",
     last_name : "Smith",
     city : "San Francisco",
     phones: [ {
          type : “work”,
          number: “1-800-555-1212”
     },
     {    type : “home”,
          number: “1-800-555-1313”,
          DNC: true
     },
     {    type : “home”,
          number: “1-800-555-1414”,
          DNC: true
     }
   ]
}
```

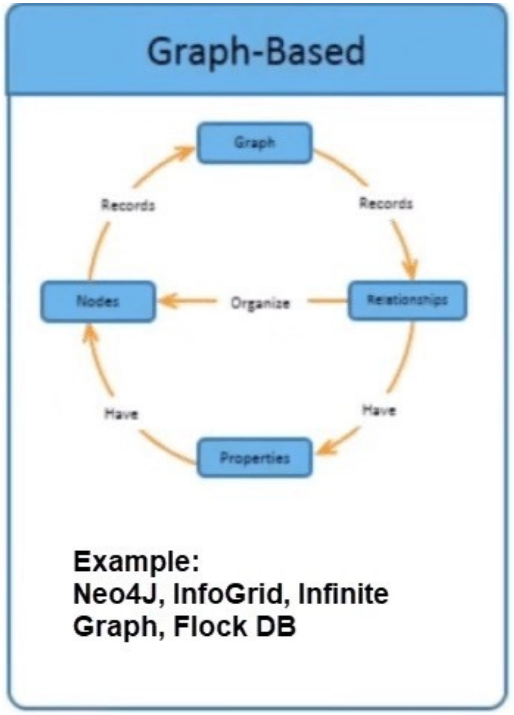

SQL Databases
NoSQL Databases
Types
One type (SQL database) with minor variations.
Different types including key-value stores, document databases, wide-column stores, and graph databases.
History
Developed in 1970s to deal with first wave of data storage applications
Developed in 2000s to deal with limitations of SQL databases, concerning scale, replication and unstructured data storage
Examples
MySQL, Postgres, Oracle Database
MongoDB, Cassandra, HBase, Neo4j
Schemas
To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.
Records can add new information on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary.
Development Model
Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)
Open-source
Supports Transactions
Yes, updates can be configured to complete entirely or not at all
In certain circumstances and at certain levels (e.g., document level vs. database level)
Data Manipulation
Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE...
Through object-oriented APIs

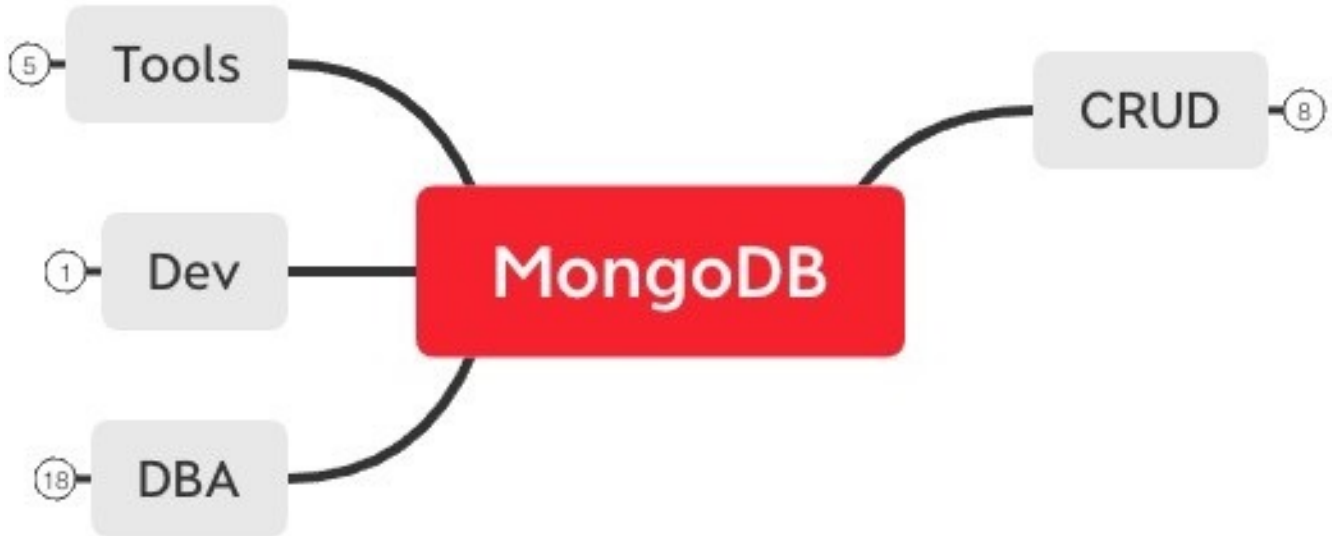

Developed in 1970s to deal with first wave of data storage applications
History
Developed in 2000s to deal with limitations of SQL databases, concerning scale, replication and unstructured data storage


To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.
Schemas
Records can add new information on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary.

NoSQL Landskab






MongoDB
Tools
5
Dev
1
DBA
18
CRUD
8

MongoDB – CRUD-operations

C	• insert()
R	• find()
U	• update()
D	• remove()

Query Selectors : Comparison	
$gt:Val	Greater then Val
$gte:Val	Greater then equals Val
$lt:Val	Lower then Val
$lte:Val	Lower then equals Val
$all:Array	All Array elements are included in field array value
$in:Array	Elements with values contained in Array
$nin:Array	Elements with values Not contained in Array
$ne:Val	Not equal

Val can be any Scalar Integer, String, Date, etc

$exists

```
db.users.find({ name: { $exists: true } })
```

$not

```
db.users.find({ name: { $not: { $eq: "Kyle" } } })
```

```
db.users.insertOne(
  {
    name: "sue",
    age: 26,
    status: "pending"
  }
)
```

collection

field: value

field: value

field: value

document

```
db.users.find(
  { age: { $gt: 18 } },
  { name: 1, address: 1 }
).limit(5)
```

collection

query criteria

projection

cursor modifier

```
db.users.updateMany(
  { age: { $lt: 18 } },
  { $set: { status: "reject" } }
)
```

collection

update filter

update action

```
db.users.deleteMany(
  { status: "reject" }
)
```

collection

delete filter

MongoDB – CRUD-operations

C	• insert()
R	• find()
U	• update()

https://github.com/jeroen/mongolite
https://jeroen.github.io/mongolite/index.html
https://cran.r-project.org/web/packages/mongolite/mongolite.pdf

find(

```
        query = '{}',
        fields = '{"_id" : 0}',
        sort = '{}',
        skip = 0,
        limit = 0,
        handler = NULL,
        pagesize = 1000
)
```

Retrieve fields from records matching query. Default handler will return all data as a single dataframe.

insert(

```
        data,
        pagesize = 1000,
        stop_on_error =TRUE,
...)
```

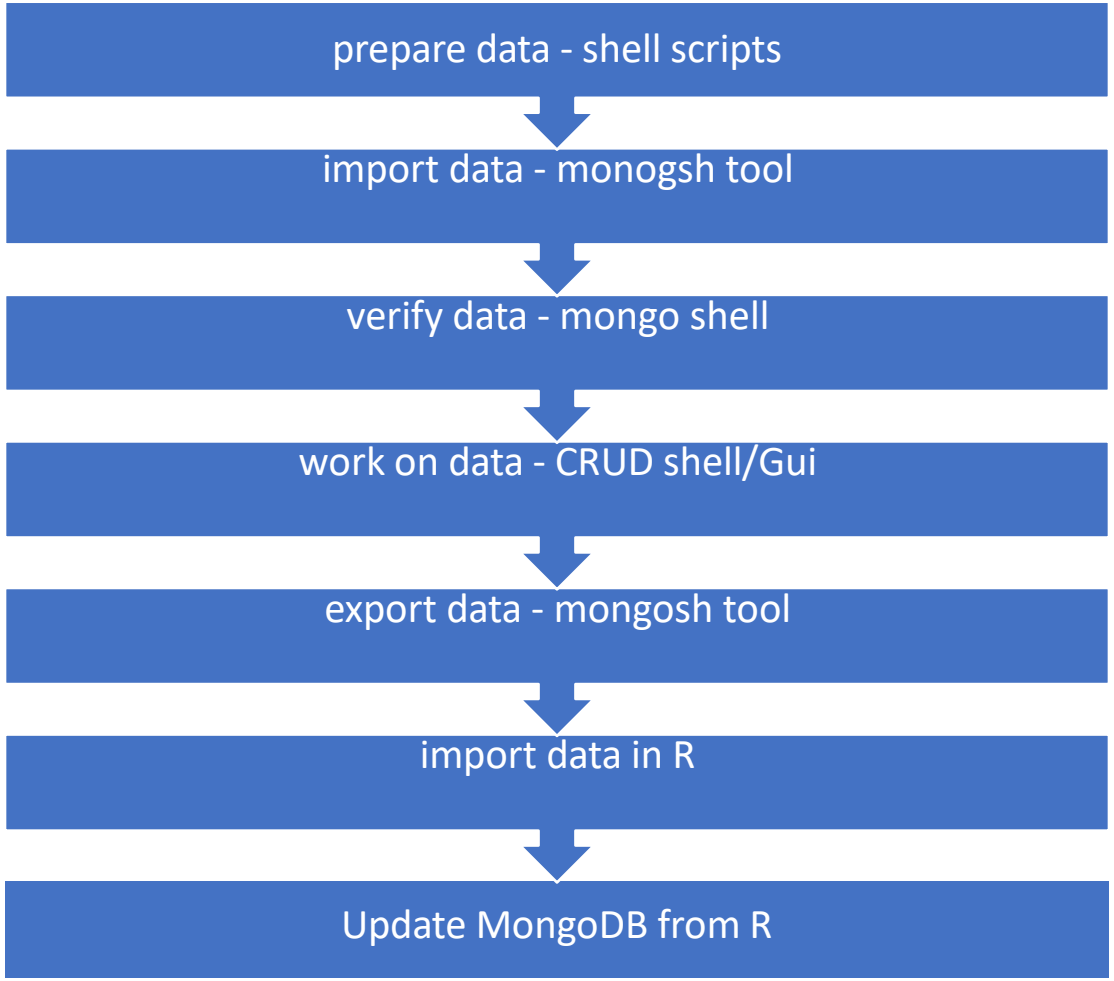
Insert rows into the collection.
Argument 'data' must be a **data-frame**, named list (for single record) or character vector with json strings (one string for each row).

update(

```
        query = '{}',
        update = '{"$set":{}}',
        upsert = FALSE,
        multiple = FALSE
)
```

Modify fields of matching record(s) with value of the update argument

Flow


prepare data - shell scripts
import data - monogsh tool
verify data - mongo shell
work on data - CRUD shell/Gui
export data - mongosh tool
import data in R
Update MongoDB from R

prepare data - postman

Home Workspaces Explore Search Postman Sign In Create Accou

Working locally in Scratch Pad. **Switch to a Workspace**

Scratch Pad New Import

Collections
APIs
Environments
Mock Servers
Monitors

API Sample Calls Copy
- GET Search Query on Players
- GET Search Query on Teams
- GET Search Query on Referees
- GET Search Query on Competitions
- GET Search Query on Players (wo...
- GET Profile Coaches WyID
- GET Areas
- GET Competitions
- GET Competitions WyID

No Environment

API Sample Calls Copy / **Competitions** Save

GET https://apirest.wyscout.com/v3/competitions?areaId=POL Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Type Basic Auth

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. variables ↗

Username ticqica-h61xjv80e-bbkd0kv-lofd51h5ga

Password ••••••••••••••••••••••••••••••

Body Cookies Headers (12) Test Results Status: 200 OK Time: 338 ms Size: 740 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "competitions": [
        {
            "wyId": 692,
            "name": "Ekstraklasa",
            "area": {
                "id": 616,
                "alpha2code": "PL",
                "alpha3code": "POL",
                "name": "Poland"
            },
            "format": "Domestic league",
            "type": "club",
            "category": "default",
            "gender": "male",
```

prepare data - shell scripts

```bash
#!/bin/bash
wyid=$1
#url='https://apirest.wyscout.com/v3/competitions?are0aId=NLD'
reqtype=$2

case $reqtype in

  mt)
    url='https://apirest.wyscout.com/v3/competitions/'$wyid'/matches'
    ;;

  aplr)
    url='https://apirest.wyscout.com/v3/competitions/'$wyid'/players?limit=100&page=1'
    ;;

  smt)
    url='https://apirest.wyscout.com/v3/matches/'$wyid
    ;;

  asmt)
    url='https://apirest.wyscout.com/v3/matches/'$wyid'/advancedstats?useSides=1'
    ;;

  evt)
    url='https://apirest.wyscout.com/v3/matches/'$wyid'/events'
    ;;

  plr)
    url='https://apirest.wyscout.com/v3/plr/'$wyid'/events'
    ;;

  *)
    echo "no"
    ;;

esac

curl -u 'username:password' -H 'Accept:application/json' $url -s | python -mjson.tool > ${reqtype}_${wyid}.json
~
```

```
2579  cat ids_nl | while read x; do ./req.sh $x evt; echo "done"; sleep 5;echo "done sleep";done &
3009  gr ids
```

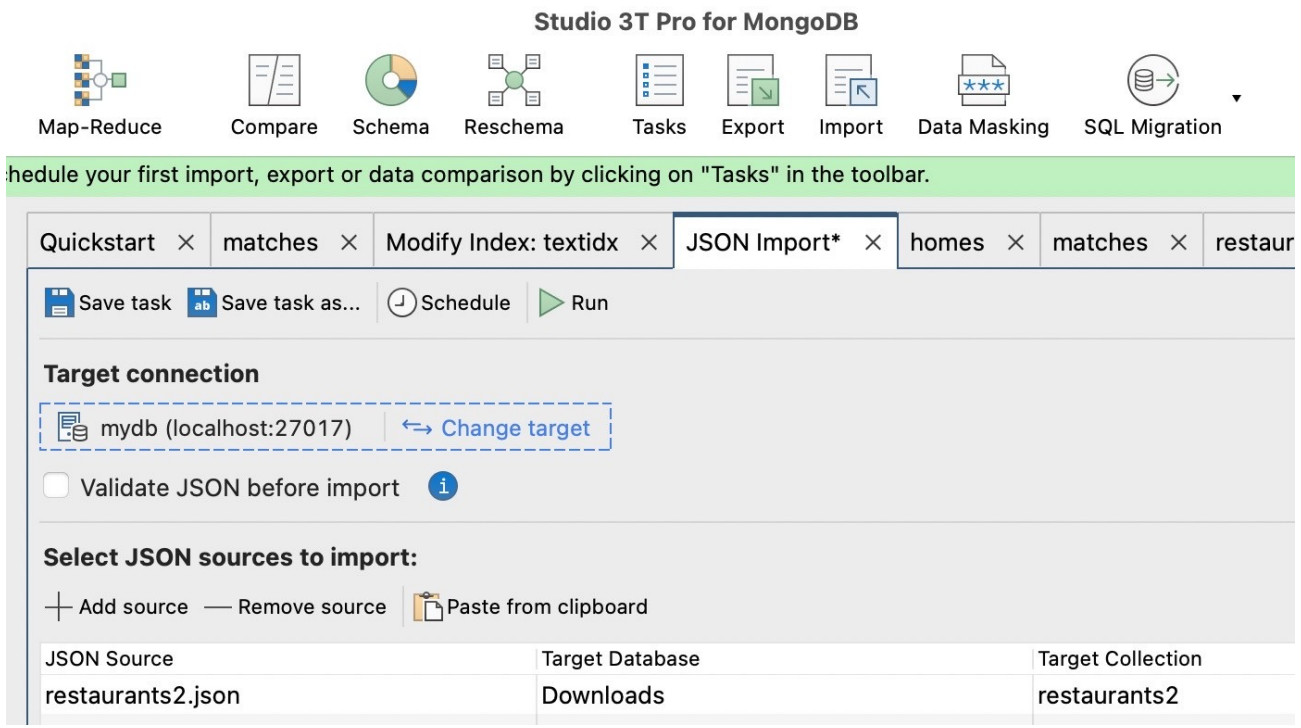
prepare data - shell scripts

```
-rw-r--r--  1 thor  staff   2,3M 25 Jan 16:50 new_evt_5360153.json
-rw-r--r--  1 thor  staff   2,1M 25 Jan 16:50 new_evt_5360154.json
-rw-r--r--  1 thor  staff   2,3M 25 Jan 16:50 new_evt_5360155.json
-rw-r--r--  1 thor  staff   2,3M 25 Jan 16:50 new_evt_5360156.json
-rw-r--r--  1 thor  staff   2,2M 25 Jan 16:50 new_evt_5360157.json
-rw-r--r--  1 thor  staff   2,0M 25 Jan 16:50 new_evt_5360158.json
-rw-r--r--  1 thor  staff   2,3M 25 Jan 16:50 new_evt_5360159.json
-rw-r--r--  1 thor  staff    22B 25 Jan 16:46 ss
(base) ____________________________________________________________
| ~/Git/wyscout-api/dutch/done @ 10153-62 (thor)
| => head new_evt_5360147.json
{
  "_id": 1468116055,
  "matchId": 5360147,
  "matchPeriod": "1H",
  "minute": 0,
  "second": 1,
  "matchTimestamp": "00:00:01.893",
  "videoTimestamp": "6.893996",
  "relatedEventId": 1468116056,
  "type": {
```

import data - monogsh tool

```
ls evt_5* | while read x; do mongoimport -d=wyscoutdutch -c=games $x; done
mongoimport --uri "mongodb+srv://soccer.cukt54z.mongodb.net/myFirstDatabase&authSource=admin" -d=wyscoutdutch
```

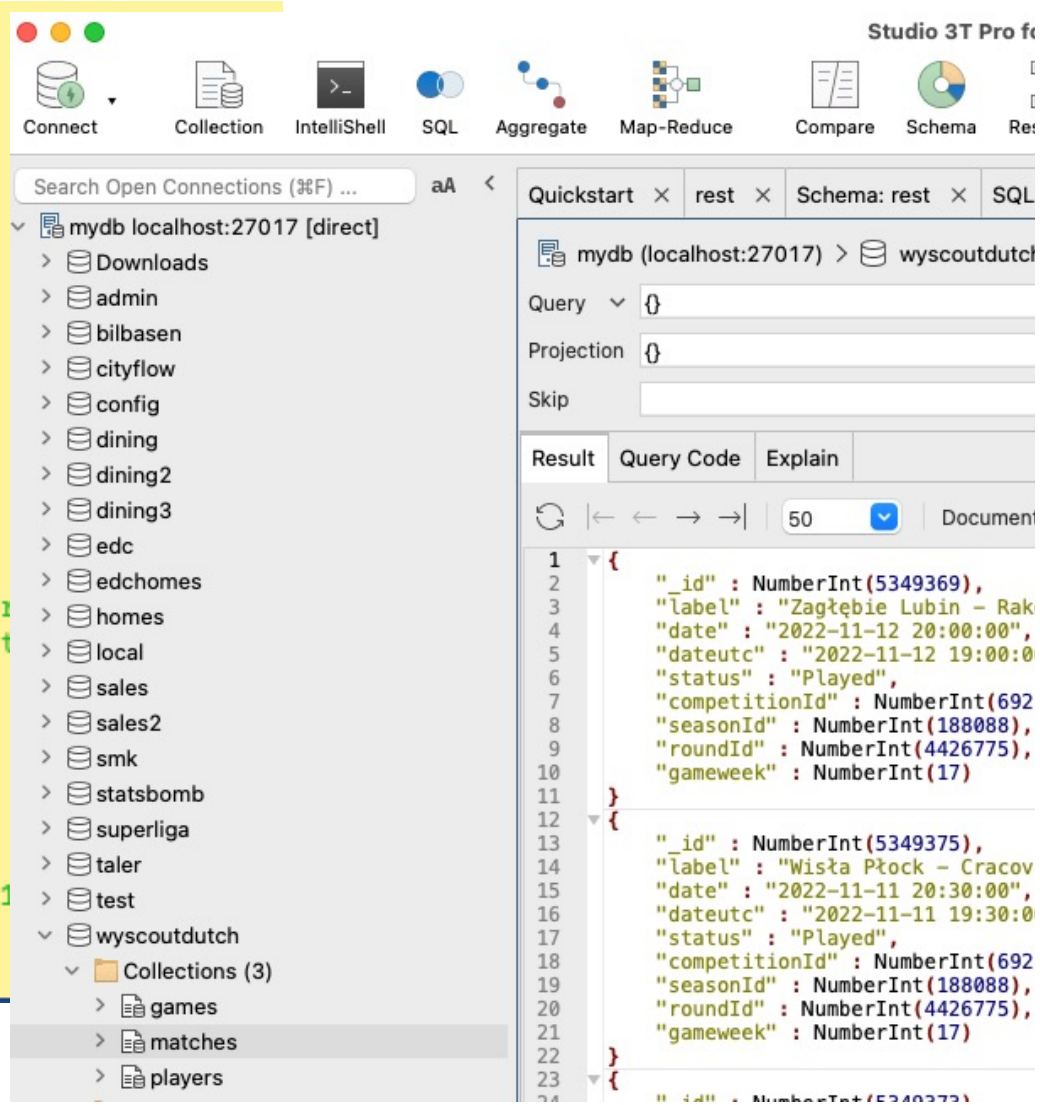
Or GUI



verify data - mongo shell

```
dining2> use wyscoutdutch
switched to db wyscoutdutch
wyscoutdutch> show collections;
games
matches
players
wyscoutdutch> db.players.findOne()
{
  _id: 869465,
  shortName: 'T. van Bergen',
  firstName: 'Thom',
  middleName: '',
  lastName: 'van Bergen',
  height: 0,
  weight: 0,
  birthDate: '2004-01-06',
  birthArea: { id: 528, alpha2code: 'NL', alpha3code: 'NLD', na
  passportArea: { id: 528, alpha2code: 'NL', alpha3code: 'NLD',
  role: { name: 'Forward', code2: 'FW', code3: 'FWD' },
  foot: null,
  currentTeamId: 23564,
  currentNationalTeamId: null,
  gender: 'male',
  status: 'active',
  imageDataURL: 'https://cdn5.wyscout.com/photos/players/public/ndplayer_1
}
wyscoutdutch>
```

Or GUI



work on data - CRUD shell/Gui

```
sales2> db.sales2.find({"storeLocation":/Lond/},{"storeLocation":1,"items.quantity":1,saleDate:1,_id:0}).limit(2)
[
  {
    saleDate: ISODate("2014-11-11T02:13:51.893Z"),
    items: [
      { quantity: 4 },
      { quantity: 7 },
      { quantity: 5 },
      { quantity: 5 },
      { quantity: 3 },
      { quantity: 5 },
      { quantity: 1 },
      { quantity: 2 },
      { quantity: 3 },
      { quantity: 1 }
    ],
    storeLocation: 'London'
  },
```

```
sales2> db.sales2.find({"storeLocation":/Lond/},{"storeLocation":1,max:{$max:"$items.quantity"},saleDate:1, _id:0}).limit(2)
[
  {
    saleDate: ISODate("2014-11-11T02:13:51.893Z"),
    storeLocation: 'London',
    max: 7
  },
```

```
sales2> db.sales2.updateOne({_id:ObjectId("5bd761dcae323e45a93ccff3")},{$set:{storeLocation:"Nyborg"}})
```

export data - mongodb tool

```
| => mongoexport -v --collection=sales --db=sales -q='{"storeLocation":"London"}' --pretty --jsonArray --out dumplondon.json
2022-03-05T12:44:49.136+0100    will listen for SIGTERM, SIGINT, and SIGKILL
2022-03-05T12:44:49.142+0100    connected to: mongodb://localhost/
2022-03-05T12:44:49.242+0100    exported 794 records
```

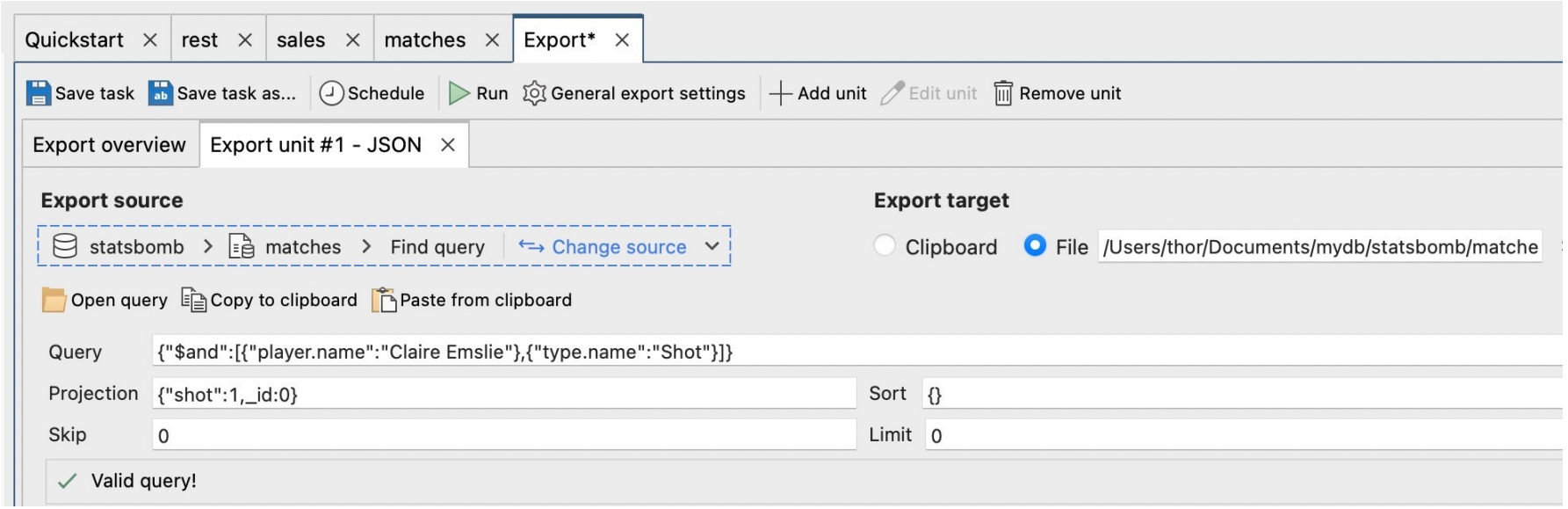
```
| => head dumplondon.json
[{
        "_id": {
                "$oid": "5bd761dcae323e45a93ccfed"
        },
        "saleDate": {
                "$date": "2015-09-02T16:11:59.565Z"
        },
        "items": [
                {
                        "name": "binder",
```

export data - mongodb tool

```
| => mongoexport -v --collection=sales --db=sales -q='{"storeLocation":"London"}' --pretty --jsonArray --out dumplondon.json
2022-03-05T12:44:49.136+0100    will listen for SIGTERM, SIGINT, and SIGKILL
2022-03-05T12:44:49.142+0100    connected to: mongodb://localhost/
2022-03-05T12:44:49.242+0100    exported 794 records
```

```
mongoexport --db=statsbomb --collection=matches --query='{"player.name":{ "$in": ["Claire Emslie","Lucy Graham"]}}' --out
dump.json
```

Or GUI



import data in R

RStudio

housing.R | Untitled1* | london | ulst | ulst | df3

```r
library(jsonlite)
library(rjson)

london <- do.call(rbind,
                  lapply(paste(readLines
                               (file("/Users/thor/Downloads/dumplondon.json"), warn=FALSE),
                               collapse=""),
                         jsonlite::fromJSON))
```

Environment | History | Connections | Tutorial

Global Environment

info	Large list (1000 elements, 1.9 MB)
json_events	12841 obs. of 20 variables
london	794 obs. of 7 variables

```
$ _id          :'data.frame': 794 obs. of  1 variable:
 ..$ $oid: chr  "5bd761dcae323e45a93ccfed" "5bd761dcae323e45a93ccfec" "5bd761dcae3…
$ saleDate     : POSIXct, format: "2015-09-02 16:11:59" "2017-12-03 18:39:48" "20…
$ items        :List of 794
 ..$ :'data.frame':    2 obs. of  4 variables:
 .. ..$ name    : chr  "binder" "binder"
 .. ..$ tags    :List of 2
 .. .. ..$ : chr  "school" "general" "organization"
 .. .. ..$ : chr  "school" "general" "organization"
 .. ..$ price   :'data.frame': 2 obs. of  1 variable:
 .. .. ..$ $numberDecimal: chr  "13.44" "16.66"
```

Update MongoDB from R

https://jeroen.github.io/mongolite/

g.R × | ulst × | ulst × | Untitled1* × | df3 × | london × | Mongo_Rest.R ×

Source on Save

```r
library(mongolite)

con <-mongo(
  collection = "rest",
  db = "dining2",
  url = "mongodb://localhost",
  verbose = TRUE
)

restorig <- con$find(limit=10)

numt=2
for (i in 1:ncol(rests)) {
  numt<-numt*0.95
  rests[i,"Viggo"]=numt[1]
}

for (i in 1:ncol(rests)) {
  key <- rests[i,6]
  val <- rests[i,9]
  con$update(
    query=paste0('{"restaurant_id": "',key,'"}'),
    update=paste0('{"$set":{"lbl": "',val,'"}}')
  )
}
```

restorig	10 obs. of 8 variables
rests	10 obs. of 9 variables

From mongo

name	restaurant_id	age	lbl
Wendy'S	30112340	30112340	kurt
Wilken'S Fine Food	40356483	30112340	kurt
Brunos On The Boulevard	40356151	30112340	kurt
Riviera Caterer	40356018	30112340	kurt
Kosher Island	40356442	30112340	kurt
Taste The Tropics Ice Cream	40356731	30112340	kurt
Regina Caterers	40356649	30112340	kurt
Dj Reynolds Pub And Restaurant	30191841	30112340	NA
Morris Park Bake Shop	30075445	30112340	NA
May May Kitchen	40358429	30112340	NA

Update mongi via R

Modif dataframe

name	restaurant_id	age	lbl	Viggo
Wendy'S	30112340	30112340	kurt	1.900000
Wilken'S Fine Food	40356483	30112340	kurt	1.805000
Brunos On The Boulevard	40356151	30112340	kurt	1.714750
Riviera Caterer	40356018	30112340	kurt	1.629012
Kosher Island	40356442	30112340	kurt	1.547562
Taste The Tropics Ice Cream	40356731	30112340	kurt	1.470184
Regina Caterers	40356649	30112340	kurt	1.396675
Dj Reynolds Pub And Restaurant	30191841	30112340	NA	1.326841
Morris Park Bake Shop	30075445	30112340	NA	NA
May May Kitchen	40358429	30112340	NA	NA