

Complementos de Bases de Dados – NoSQL– (*mongoDB*)

Engenharia Informática
2º Ano / 1º Semestre

Cláudio Miguel Sapateiro – claudio.sapateiro@estsetubal.ips.pt
João Portelinha Santos - joao.portelinha@estsetubal.ips.pt

Sumário

- Introdução
- Bases de Dados NoSQL
- Modelos de armazenamento
- Relacional vs NoSQL
- MongoDB (demonstração)

Introdução

Sistemas não-relacional e o NoSQL

- Surgimento de tecnologias
 - AJAX (*Asynchronous JavaScript and XML*)
 - JSON (*JavaScript Object Notation*): depressa ultrapassou o XML e foi considerado o **standard de facto** para armazenamento e serialização de objetos em disco, e comunicação
- Websites começam a usar JSON para comunicar e armazenam como documentos em colunas de bases de dados relacionais
- A componente relacional é eliminada, ou deferida, e recorreu-se a bases de dados onde *documentos* **JSON** podiam ser armazenados diretamente as: Document Databases

Introdução

Sistemas não-relacional e o NoSQL

- Entre 2008 e 2009 acontece uma explosão de novos sistemas de Bases de Dados, integrando os avanços tecnológicos
- Algumas persistem:
 - MongoDB, Cassandra, neo4j, ...
- Surge a designação para este tipo de Bases de dados: “***Distributed Non-Relational Database Management System (DNRDMS)***”, mas não era claro relativamente ao conceito subjacente
 - ✓ Surge então o termo **NoSQL** a referir sistemas de bases de dados que quebravam com as tradicionais bases de dados relacionais
- ✓ O termo NoSQL é, refere-se a Bases de Dados que priorizam a **escalabilidade** e **disponibilidade**, em oposição à **atomicidade** e **consistência transacionais**

Bases de Dados NoSQL

Modelos de armazenamento

- Key-value
- Column-oriented
- Graph
- Document

Bases de Dados NoSQL

Modelos de armazenamento

- Key-value

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Bases de Datos NoSQL

Modelos de armazenamento

- Column-oriented

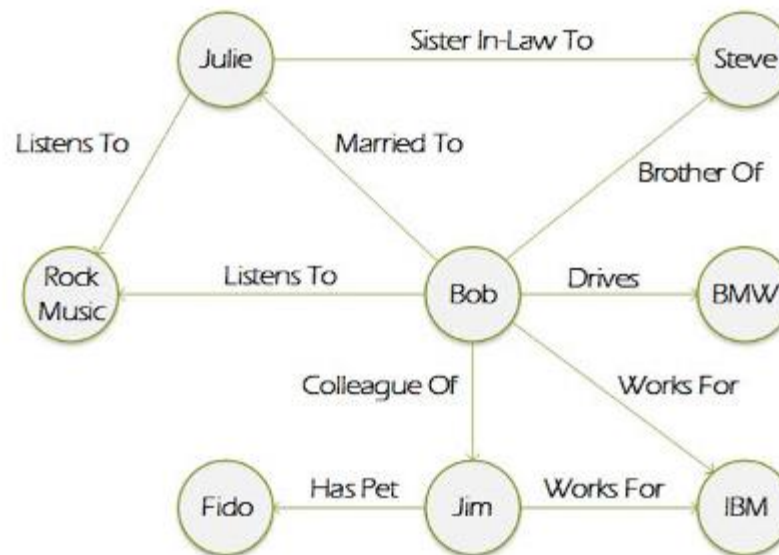
Row-oriented			
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented					
Name	ID	Grade	ID	GPA	ID
John	001	Senior	001	4.00	001
Karen	002	Freshman	002	3.67	002
Bill	003	Junior	003	3.33	003

Bases de Dados NoSQL

Modelos de armazenamento

- Graph



Bases de Datos NoSQL

Modelos de armazenamento

- Document

Relational

ID	first_name	last_name	cell	city	year_of_birth	location_x	location_y
1	'Mary'	'Jones'	'516-555-2048'	'Long Island'	1986	'-73.9876'	'40.7574'

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'









ID	user_id	name	version
20	1	'MyApp'	1.0.4
21	1	'DocFinder'	2.5.7

ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

MongoDB

```
{
  first_name: "Mary",
  last_name: "Jones",
  cell: "516-555-2048",
  city: "Long Island",
  year_of_birth: 1986,
  location: {
    type: "Point",
    coordinates: [-73.9876, 40.7574]
  },
  profession: ["Developer", "Engineer"],
  apps: [
    { name: "MyApp",
      version: 1.0.4 },
    { name: "DocFinder",
      version: 2.5.7 }
  ],
  cars: [
    { make: "Bentley",
      year: 1973 },
    { make: "Rolls Royce",
      year: 1965 }
  ]
}
```

Bases de Datos NoSQL

Type	Example	
Key-Value Store	 redis	 riak
Wide Column Store	 H-BASE	 cassandra
Document Store	 mongoDB	 CouchDB relax
Graph Store	 Neo4j	 The Distributed Graph Database

Relacional vs NoSQL

(<https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data>)

Consider a NoSQL datastore when:

You have high volume workloads that require large scale

Your workloads don't require ACID guarantees

Your data is dynamic and frequently changes

Data can be expressed without relationships

You need fast writes and write safety isn't critical

Data retrieval is simple and tends to be flat

Your data requires a wide geographic distribution

Your application will be deployed to commodity hardware, such as with public clouds

Consider a relational database when:

Your workload volume is consistent and requires medium to large scale

ACID guarantees are required

Your data is predictable and highly structured

Data is best expressed relationally

Write safety is a requirement

You work with complex queries and reports

Your users are more centralized

Your application will be deployed to large, high-end hardware

mongoDB

- Document Database

- Em MongoDB um registro é um documento, que é uma estrutura de dados composta por pares campo/valor. Os documentos do MongoDB são semelhantes aos objetos JSON. Os valores dos campos podem incluir outros documentos, arrays e arrays de documentos.

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```



← field: value
← field: value
← field: value
← field: value

mongoDB

- **Collections/Views/On-Demand Materialized Views**
 - O MongoDB armazena documentos em coleções (*collections*). As coleções são análogas às tabelas em bases de dados relacionais.
 - Para além de coleções, o MongoDB oferece suporte a:
 - Read-only Views (Starting in MongoDB 3.4)
 - On-Demand Materialized Views (Starting in MongoDB 4.2).

mongoDB

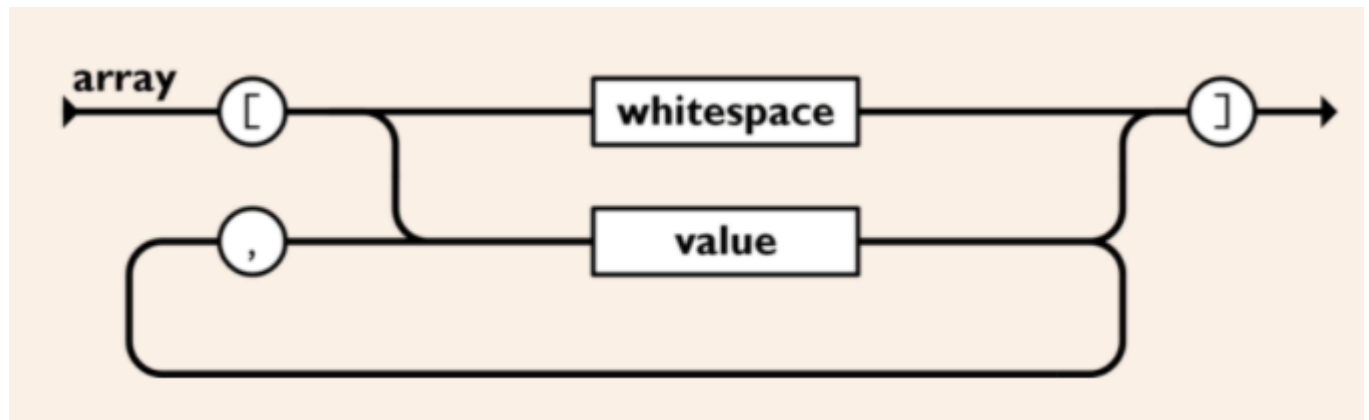
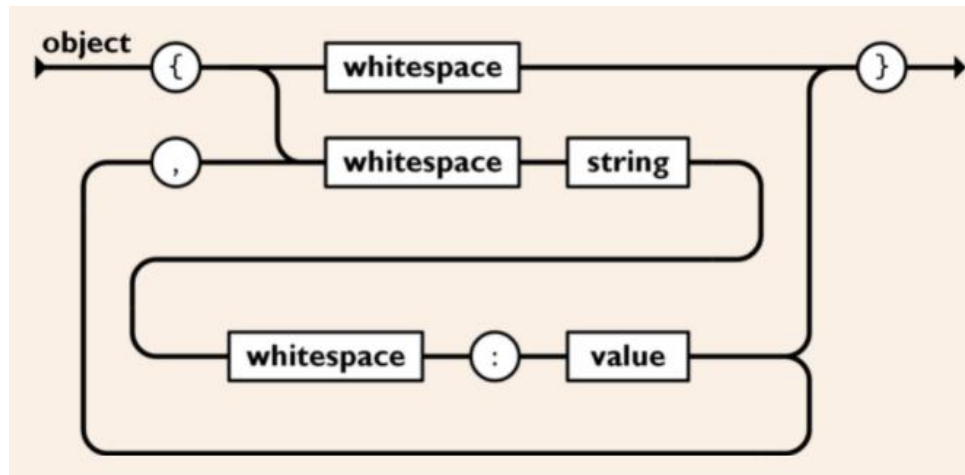
- **Rich Query Language**
 - MongoDB disponibiliza uma linguagem de consulta avançada, de suporte às operações de escrita e leitura (CRUD), assim como:
 - Data Aggregation
 - Text Search and Geospatial Queries.
 - O MongoDB armazena os registos de dados como documentos (especificamente documentos BSON) que são reunidos em coleções.
 - BSON é uma representação binária de documentos JSON.

JSON

- Sintaxe
 - Informação em pares nome/valor
 - Informação separada por “ “
 - As chavetas para representar objetos {..}
 - Parênteses retos para representar arrays [...]

JSON

- Sintaxe



mongoDB

- Criar uma base de dados

use *myNewDB*

- Se a base de dados não existe, esta é criada no momento em que pela primeira vez se insere informação-

mongoDB

- Criar uma *collection*

`db.myNewCollection.insertOne({ x: 1 })`

– Se a collection não existir é criada, se não é adicionada a informação à já existente.

– Criação explícita

`db.createCollection("myNewCollection")`

mongoDB

- Popular uma *collection* (Insert)

`db.collection.insertOne()`

`db.collection.insertMany()`

```
db.inventory.insertMany( [  
  { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },  
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  
  { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },  
  { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },  
  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },  
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },  
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },  
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },  
  { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  
  { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }  
] );
```

mongoDB

- **Atualizar um documento (Update)**

`db.collection.updateOne(<filter>, <update>, <options>)`

`db.collection.updateMany(<filter>, <update>, <options>)`

`db.collection.replaceOne(<filter>, <update>, <options>)`

mongoDB

- Update document: *exemplos*

```
db.inventory.updateOne(  
  { item: "paper" },  
  {  
    $set: { "size.uom": "cm", status: "P" }  
  }  
)
```

```
db.inventory.updateMany(  
  { "qty": { $lt: 50 } },  
  {  
    $set: { "size.uom": "in", status: "P" }  
  }  
)
```

```
db.inventory.replaceOne(  
  { item: "paper" },  
  { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }  
)
```

mongoDB

- **Remover documentos (delete)**

`db.collection.deleteMany()`

`db.collection.deleteOne()`

```
db.inventory.deleteOne( { status: "D" } )
```

```
db.inventory.deleteMany({ status : "A" })
```

mongoDB

- **Remover *collections***
`db.collection.drop()`

mongoDB

- Query Documents






```
// SELECT * FROM inventory
db.inventory.find({})
```

```
// SELECT * FROM inventory WHERE status = 'D'
db.inventory.find({status:"D"})
```

```
// SELECT * FROM inventory WHERE status = 'A' AND qty < 30
db.inventory.find({status:"A", qty: {$lt:30}})
```






```
// SELECT * FROM inventory WHERE status = "D" OR qty < 30
db.inventory.find({ $or: [ { status: "D" }, { qty: { $lt: 30 } } ] })
```

```
// SELECT * FROM inventory WHERE status = "A" AND ( qty < 30 OR item LIKE "p%")
db.inventory.find({ status: "A", $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ] })
```

 _id	61cb2ac58efbfd23e472f33a	ObjectId
 item	canvas	String
 qty	100	Double
 size	{ h : 28, w : 35.5, uom : "cm" }	Object
 status	A	String

mongoDB

- Query on Embedded/Nested Documents

 _id	61cb2ac58efbfd23e472f33a	ObjectId
 item	canvas	String
 qty	100	Double
 size	{ h : 28, w : 35.5, uom : "cm" }	Object
 status	A	String

```
db.inventory.find({ size: { h: 14, w: 21, uom: "cm" } })
```

```
db.inventory.find({ "size.uom": "in" })
```

```
db.inventory.find({ "size.h": { $lt: 15 } })
```

```
db.inventory.find({ "size.h": { $lt: 15 }, "size.uom": "in", status: "D" })
```

mongoDB

- Query an Array

 _id	61cb32708efbfd23e472f344	ObjectId
 item	journal	String
 qty	25	Double
▷  tags	["blank", "red"]	Array
▷  dim_cm	[14, 21]	Array

```
db.inventory.find( { tags: ["red", "blank"] } )
```

```
db.inventory.find( { tags: { $all: ["red", "blank"] } } )
```






```
// Query an Array with Compound Filter Conditions on the Array Elements  
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } )
```

```
//Query for an Array Element that Meets Multiple Criteria  
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )
```

```
//Query for an Element by the Array Index Position  
db.inventory.find( { "dim_cm.1": { $gt: 25 } } )
```

mongoDB

- Query an Array of Embedded Documents

 _id	61cb34d98efbfd23e472f349	ObjectId
 item	journal	String
 instock	[{ warehouse : "A", qty : 5 }, { warehouse : "C", qty : 15 }]	Array
▶  0	{ warehouse : "A", qty : 5 }	Object
▶  1	{ warehouse : "C", qty : 15 }	Object

```
db.inventory.find( { "instock": { warehouse: "A", qty: 5 } } )
```

```
db.inventory.find( { 'instock.qty': { $lte: 20 } } )
```

```
db.inventory.find( { "instock": { $elemMatch: { qty: 5, warehouse: "A" } } } )
```

```
db.inventory.find( { "instock.qty": 5, "instock.warehouse": "A" } )
```

mongoDB

- Project Fields to Return from Query

```
// SELECT _id, item, status from inventory
db.inventory.find( {}, { item: 1, status: 1 } )
```

▷ (1) 61cb2ac58efbfd23e472f33a	{ item : "canvas", status : "A" }	Document
▷ (2) 61cb2ac58efbfd23e472f33b	{ item : "journal", status : "A" }	Document
▷ (3) 61cb2ac58efbfd23e472f33c	{ item : "mat", status : "A" }	Document
▷ (4) 61cb2ac58efbfd23e472f33d	{ item : "mousepad", status : "P" }	Document

```
// SELECT item, status from inventory WHERE status = "A"
db.inventory.find( { status: "A" }, { item: 1, status: 1, _id: 0 } )
```

Key	Value	Type
▷ (1)	{ item : "canvas", status : "A" }	Object
▷ (2)	{ item : "journal", status : "A" }	Object

Nota: O `_id` é apresentado por defeito

mongoDB

- **Project Specific Fields in Embedded Documents**

```
db.inventory.find( { status: "A" }, { item: 1, status: 1, "size.uom": 1 })  
db.inventory.find( { }, { item: 1, "instock.qty": 1 } )
```

mongoDB

- Query for Null or Missing Fields

```
db.inventory.find( { status: {$exists:false} } )
```

```
db.inventory.find( { "instock.qty": {$exists:false} } )
```

mongoDB

- **Result Set Processing**

```
db.inventory.find().sort({item:-1}) // 1 asc, -1 desc
```

```
db.inventory.find().count()
```

```
db.inventory.find().limit(5) // first #
```

```
db.inventory.find().sort({item:-1}).limit(1)
```

mongoDB

- **Aggregation**

- As operações de agregação processam vários documentos e retornam resultados computados. Você pode usar operações de agregação para:
 - Agrupamento de valores de vários documentos.
 - Operações sobre os dados agrupados que retornam um único resultado.
- Para realizar operações de agregação, pode ser utilizado:
 - *Aggregation pipelines*
 - *Single purpose aggregation methods*

mongoDB

- **Aggregation pipeline**
 - consiste em uma ou mais etapas que processam documentos:
 - Cada etapa executa uma operação nos documentos de entrada. Por exemplo, uma etapa pode filtrar documentos, agrupar documentos e calcular valores.
 - Os documentos que são produzidos numa etapa são inseridos na próxima etapa.
 - Um pipeline de agregação pode retornar resultados para grupos de documentos. Por exemplo, retornar os valores total, médio, máximo e mínimo.

mongoDB

- **Aggregation pipeline**
 - Etapas

Name	Description
\$addField	Adds new fields to documents. Outputs documents that contain all existing fields from the input documents and newly added fields.
\$count	Returns a count of the number of documents at this stage of the aggregation pipeline.
\$group	Groups input documents by a specified identifier expression and applies the accumulator expression(s), if specified, to each group. Consumes all input documents and outputs one document per each distinct group. The output documents only contain the identifier field and, if specified, accumulated fields.
\$limit	Passes the first n documents unmodified to the pipeline where n is the specified limit. For each input document, outputs either one document (for the first n documents) or zero documents (after the first n documents).
\$lookup	Performs a left outer join to another collection in the same database to filter in documents from the "joined" collection for processing.

mongoDB

- **Aggregation pipeline**

- Etapas

Name	Description
\$match	Filters the document stream to allow only matching documents to pass unmodified into the next pipeline stage. \$match uses standard MongoDB queries. For each input document, outputs either one document (a match) or zero documents (no match).
\$out	Writes the resulting documents of the aggregation pipeline to a collection. To use the \$out stage, it must be the last stage in the pipeline.
\$project	Reshapes each document in the stream, such as by adding new fields or removing existing fields. For each input document, outputs one document.
\$set	Adds new fields to documents. Outputs documents that contain all existing fields from the input documents and newly added fields.
\$sort	Reorders the document stream by a specified sort key. Only the order changes; the documents remain unmodified. For each input document, outputs one document.

mongoDB

- **Aggregation pipeline**

- Lista completa de etapas em:

- <https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/#std-label-aggregation-pipeline-operator-reference>

mongoDB

- Aggregation pipeline: **\$addFields**

scores 0.411 s 2 Docs	
Key	Value
▶ (1) 2	{ student: "Ryan", homework: [5, 6, 5], quiz: [8, 8], extraCredit: 8 } (5 fields)
▶ (2) 1	{ student: "Maya", homework: [10, 5, 10], quiz: [10, 8], extraCredit: 0 } (5 fields)

```
db.scores.aggregate( [  
  {  
    $addFields: {  
      totalHomework: { $sum: "$homework" },  
      totalQuiz: { $sum: "$quiz" }  
    },  
  },  
  {  
    $addFields: { totalScore:  
      { $add: [ "$totalHomework", "$totalQuiz", "$extraCredit" ] } }  
  }  
] )
```

(1) 1	{8 fields}
_id	1
student	Maya
homework	[10, 5, 10]
quiz	[10, 8]
extraCredit	0
totalHomework	25
totalQuiz	18
totalScore	43

mongoDB

- Aggregation pipeline: **\$match**

scores		0.411 s	2 Docs
Key	Value		
(1) 2	{ student: "Ryan", homework: [5, 6, 5], quiz: [8, 8], extraCredit: 8 } (5 fields)		
(2) 1	{ student: "Maya", homework: [10, 5, 10], quiz: [10, 8], extraCredit: 0 } (5 fields)		



```
db.scores.aggregate( [  
  {  
    $match: { extraCredit: { $gt: 0 } }  
  }  
)
```



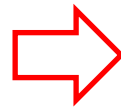
(1) 2	{ student: "Ryan", ... }
_id	2
student	Ryan
homework	[5, 6, 5]
quiz	[8, 8]
extraCredit	8

mongoDB

- Aggregation pipeline: **\$count**

inventory 0.397 s | 10 Docs

Key	Value
(1) 61cc735b8efbfd23e	{ item : "sketch pad", qty : 95, size : { h : 22.9, w : 14, v : 10 }
(2) 61cc735b8efbfd23e	{ item : "sketchbook", qty : 80, size : { h : 27.9, w : 14, v : 10 }
(3) 61cc735b8efbfd23e	{ item : "postcard", qty : 45, size : { h : 10, w : 14, v : 10 }
(4) 61cc735b8efbfd23e	{ item : "planner", qty : 75, size : { h : 22.9, w : 14, v : 10 }
(5) 61cc735b8efbfd23e	{ item : "paper", qty : 100, size : { h : 8.5, w : 14, v : 10 }
(6) 61cc735b8efbfd23e	{ item : "notebook", qty : 50, size : { h : 8.5, w : 14, v : 10 }
(7) 61cc735b8efbfd23e	{ item : "mousepad", qty : 25, size : { h : 1, w : 14, v : 10 }
(8) 61cc735b8efbfd23e	{ item : "mat", qty : 85, size : { h : 27.9, w : 14, v : 10 }
(9) 61cc735b8efbfd23e	{ item : "journal", qty : 25, size : { h : 14, w : 14, v : 10 }
(10) 61cc735b8efbfd23	{ item : "canvas", qty : 100, size : { h : 28, w : 14, v : 10 }



```
db.inventory.aggregate( [  
  {  
    $count: "number_of_items"  
  }  
)
```



Key	Value
(1)	{ number_of_items : 10 }
number_of_items	10

mongoDB

- Aggregation pipeline: **\$group**

inventory 0.397 s | 10 Docs

Key	Value
(1) 61cc735b8efbfd23e	{ item : "sketch pad", qty : 95, size : { h : 10, w : 14, v : 14.5 } }
(2) 61cc735b8efbfd23e	{ item : "sketchbook", qty : 80, size : { h : 10, w : 14, v : 14.5 } }
(3) 61cc735b8efbfd23e	{ item : "postcard", qty : 45, size : { h : 10, w : 14, v : 14.5 } }
(4) 61cc735b8efbfd23e	{ item : "planner", qty : 75, size : { h : 22.9, w : 14, v : 14.5 } }
(5) 61cc735b8efbfd23e	{ item : "paper", qty : 100, size : { h : 8.5, w : 14, v : 14.5 } }
(6) 61cc735b8efbfd23e	{ item : "notebook", qty : 50, size : { h : 8.5, w : 14, v : 14.5 } }
(7) 61cc735b8efbfd23e	{ item : "mousepad", qty : 25, size : { h : 14, w : 14, v : 14.5 } }
(8) 61cc735b8efbfd23e	{ item : "mat", qty : 85, size : { h : 27.9, w : 14, v : 14.5 } }
(9) 61cc735b8efbfd23e	{ item : "journal", qty : 25, size : { h : 14, w : 14, v : 14.5 } }
(10) 61cc735b8efbfd23	{ item : "canvas", qty : 100, size : { h : 28, w : 14, v : 14.5 } }

```
// SELECT COUNT(*) AS count FROM inventory
```

```
db.inventory.aggregate( [  
  {  
    $group: {  
      _id: null,  
      count: { $sum: 1 }  
    }  
  }  
)
```

OU

```
db.inventory.aggregate( [  
  {  
    $group: {  
      _id: null,  
      count: { $count: {} } //version 5.0  
    }  
  }  
)
```

(1) null { count : 10 }

key	value
_id	null
count	10

mongoDB

- Aggregation pipeline: **\$group**

inventory 0.397 s | 10 Docs

Key	Value
(1) 61cc735b8efbfd23e	{ item : "sketch pad", qty : 95, size : { h : 2, w : 10, t : 10 } }
(2) 61cc735b8efbfd23e	{ item : "sketchbook", qty : 80, size : { h : 10, w : 10, t : 10 } }
(3) 61cc735b8efbfd23e	{ item : "postcard", qty : 45, size : { h : 10, w : 10, t : 10 } }
(4) 61cc735b8efbfd23e	{ item : "planner", qty : 75, size : { h : 22.9, w : 10, t : 10 } }
(5) 61cc735b8efbfd23e	{ item : "paper", qty : 100, size : { h : 8.5, w : 10, t : 10 } }
(6) 61cc735b8efbfd23e	{ item : "notebook", qty : 50, size : { h : 8.5, w : 10, t : 10 } }
(7) 61cc735b8efbfd23e	{ item : "mousepad", qty : 25, size : { h : 1, w : 10, t : 10 } }
(8) 61cc735b8efbfd23e	{ item : "mat", qty : 85, size : { h : 27.9, w : 10, t : 10 } }
(9) 61cc735b8efbfd23e	{ item : "journal", qty : 25, size : { h : 14, w : 10, t : 10 } }
(10) 61cc735b8efbfd23	{ item : "canvas", qty : 100, size : { h : 28, w : 10, t : 10 } }



```
// SELECT SUM(qty) as "QtyTotal"
// FROM inventory
// GROUP BY status

db.inventory.aggregate( [
  {
    $group: {
      _id: "$status",
      QtyTotal: { $sum: "$qty" }
    }
  }
] )
```



(1) D	{ QtyTotal : 175 }
(2) P	{ QtyTotal : 75 }
(3) A	{ QtyTotal : 430 }

mongoDB

- Aggregation pipeline: **\$group**

inventory 0.397 s 10 Docs	
Key	Value
(1) 61cc735b8efbfd23e	{ item : "sketch pad", qty : 95, size : { h : 22.9, w : 14, v : 100 } }
(2) 61cc735b8efbfd23e	{ item : "sketchbook", qty : 80, size : { h : 22.9, w : 14, v : 100 } }
(3) 61cc735b8efbfd23e	{ item : "postcard", qty : 45, size : { h : 10, w : 14, v : 100 } }
(4) 61cc735b8efbfd23e	{ item : "planner", qty : 75, size : { h : 22.9, w : 14, v : 100 } }
(5) 61cc735b8efbfd23e	{ item : "paper", qty : 100, size : { h : 8.5, w : 14, v : 100 } }
(6) 61cc735b8efbfd23e	{ item : "notebook", qty : 50, size : { h : 8.5, w : 14, v : 100 } }
(7) 61cc735b8efbfd23e	{ item : "mousepad", qty : 25, size : { h : 14, w : 14, v : 100 } }
(8) 61cc735b8efbfd23e	{ item : "mat", qty : 85, size : { h : 27.9, w : 14, v : 100 } }
(9) 61cc735b8efbfd23e	{ item : "journal", qty : 25, size : { h : 14, w : 14, v : 100 } }
(10) 61cc735b8efbfd23	{ item : "canvas", qty : 100, size : { h : 28, w : 14, v : 100 } }



```
// SELECT SUM(qty) as "QtyTotal"
// FROM inventory
// GROUP BY status
// HAVING QtyTotal > 200

db.inventory.aggregate( [
  //First Stage
  {
    $group: {
      _id: "$status",
      QtyTotal: { $sum: "$qty" }
    }
  },
  //Second Stage
  {
    $match: {
      "QtyTotal": { $gt: 200 }
    }
  }
] )
```



(1) A	{ QtyTotal : 430 }
_id	A
QtyTotal	430

<https://docs.mongodb.com/manual/reference/operator/aggregation/group/#mongodb-pipeline-pipe.-group>

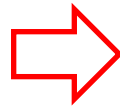
mongoDB

- Aggregation pipeline: **\$unwind**

scores 0.411 s 2 Docs	
Key	Value
▶ (1) 2	{ student : "Ryan", homework : [5, 6, 5], quiz : [8, 8], extraCredit : 8 } (5 fields)
▶ (2) 1	{ student : "Maya", homework : [10, 5, 10], quiz : [10, 8], extraCredit : 0 } (5 fields)



```
db.scores.aggregate( [  
  {  
    $unwind: "$homework"  
  }  
)
```



```
db.scores.aggregate( [  
  {  
    $unwind: "$homework"  
  },  
  {  
    $match: { homework: 10 }  
  }  
)
```

Key	Value
▶ (1) 1	{ student : "Maya", homework : 10, quiz : [10, 8], extraCredit : 0 } (5 fields)
▶ (2) 1	{ student : "Maya", homework : 5, quiz : [10, 8], extraCredit : 0 } (5 fields)
▶ (3) 1	{ student : "Maya", homework : 10, quiz : [10, 8], extraCredit : 0 } (5 fields)
▶ (4) 2	{ student : "Ryan", homework : 5, quiz : [8, 8], extraCredit : 8 } (5 fields)
▶ (5) 2	{ student : "Ryan", homework : 6, quiz : [8, 8], extraCredit : 8 } (5 fields)
▶ (6) 2	{ student : "Ryan", homework : 5, quiz : [8, 8], extraCredit : 8 } (5 fields)

mongoDB

- Aggregation pipeline: **\$out**

books 0.392 s 5 Docs

Key	Value
(1) 8752	{ title : "Divine Comedy", author : "Dante", copies : 1 } (4 fields)
(2) 8751	{ title : "The Banquet", author : "Dante", copies : 2 } (4 fields)
(3) 8645	{ title : "Eclogues", author : "Dante", copies : 2 } (4 fields)
(4) 7020	{ title : "Iliad", author : "Homer", copies : 10 } (4 fields)
(5) 7000	{ title : "The Odyssey", author : "Homer", copies : 10 } (4 fields)



```
db.books.aggregate( [
  { $group : { _id : "$author", books: { $push: "$title" } } },
  { $out : "authors" }
] )
```

```
db.authors.find()
```








authors 0.430 s 2 Docs

Key	Value
(1) Homer	{ books : ["The Odyssey", "Iliad"] }
(2) Dante	{ books : ["The Banquet", "Divine Comedy", "Eclogues"] }

mongoDB

- Aggregation pipeline: **\$lookup**

 orders  0.343 s 2 Docs	
Key	Value 
 (1) 2	{ item : "journal", price : 20, quantity : 1 } (4 fields)
 (2) 1	{ item : "canvas", price : 12, quantity : 2 } (4 fields)

Key	Value
(1) 61cc735b8efbfd23e472f35c	{ item : "sketch pad", qty : 95, size : { h : 22.85, w : 35.5, uom : "cm" } }
(2) 61cc735b8efbfd23e472f35b	{ item : "sketchbook", qty : 80, size : { h : 14, w : 21, uom : "cm" } }
(3) 61cc735b8efbfd23e472f35a	{ item : "postcard", qty : 45, size : { h : 10, w : 19, uom : "cm" } }
(4) 61cc735b8efbfd23e472f359	{ item : "planner", qty : 75, size : { h : 22.85, w : 35.5, uom : "cm" } }
(5) 61cc735b8efbfd23e472f358	{ item : "paper", qty : 100, size : { h : 8.5, w : 11, uom : "cm" } }
(6) 61cc735b8efbfd23e472f357	{ item : "notebook", qty : 50, size : { h : 8.5, w : 11, uom : "cm" } }
(7) 61cc735b8efbfd23e472f356	{ item : "mousepad", qty : 25, size : { h : 19, w : 27.9, uom : "cm" } }
(8) 61cc735b8efbfd23e472f355	{ item : "mat", qty : 85, size : { h : 27.9, w : 35.5, uom : "cm" } }
(9) 61cc735b8efbfd23e472f354	{ item : "journal", qty : 25, size : { h : 14, w : 21, uom : "cm" } }
(10) 61cc735b8efbfd23e472f353	{ item : "canvas", qty : 100, size : { h : 28, w : 35.5, uom : "cm" } }

```
db.inventory.aggregate( [
  {
    $lookup:
    {
      from: "orders",
      localField: "item",
      foreignField: "item",
      as: "inventory_orders"
    }
  }
] )
```

Key	Value
(1) 61cc735b8efbfd23e472f353	{ status : "A" } (6 fields)
_id	61cc735b8efbfd23e472f353
item	canvas
qty	100
size	{ h : 28, w : 35.5, uom : "cm" }
status	A
inventory_orders	[{ _id : 1, item : "canvas", price : 12, quantity : 2 }]

mongoDB

- Os exemplos apresentados foram baseados no manual do MongoDB

<https://docs.mongodb.com/manual/tutorial/getting-started/>

MS-SQL Server

- Exportar informação em JSON
 - SELECT ... FROM ... **FOR JSON AUTO**

<https://docs.microsoft.com/en-us/sql/relational-databases/json/format-query-results-as-json-with-for-json-sql-server?view=sql-server-ver15>

Complementos de Bases de Dados – NoSQL– (*mongoDB*)

Engenharia Informática
2º Ano / 1º Semestre

Cláudio Miguel Sapateiro – claudio.sapateiro@estsetubal.ips.pt
João Portelinha Santos - joao.portelinha@estsetubal.ips.pt