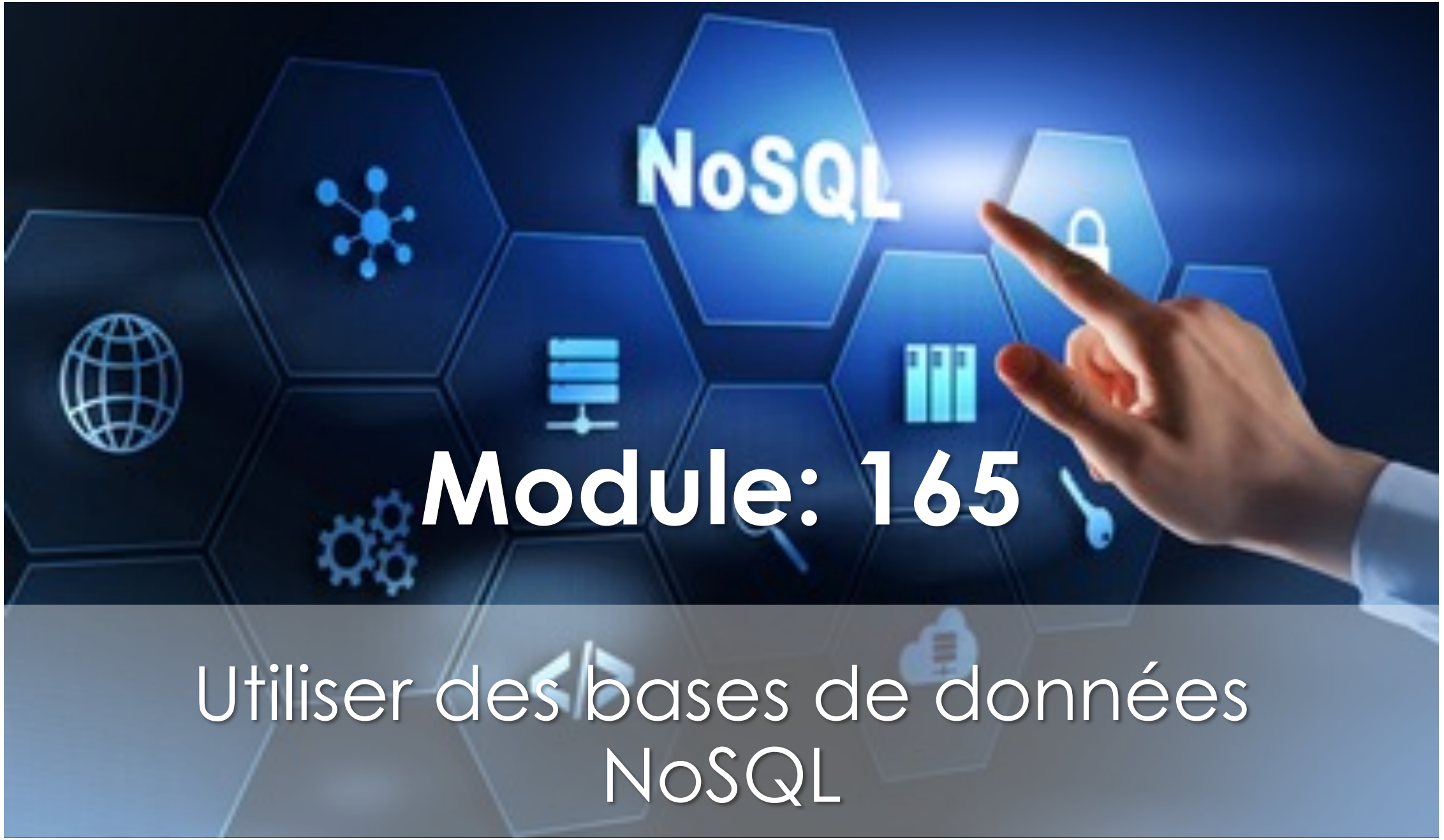
ETML Informatique



Module: 165

Utiliser des bases de données NoSQL

ETML Informatique

MongoDB
Les agrégations

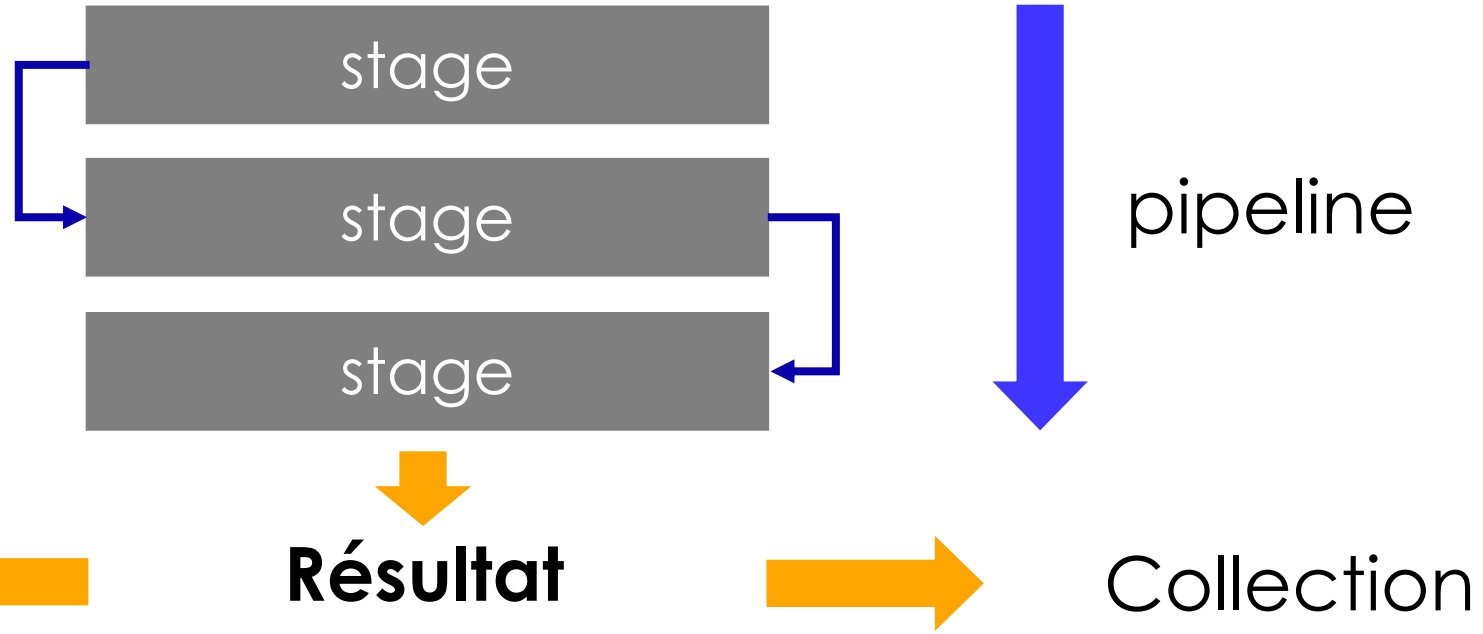
MongoDB
Les agrégations

Les agrégations fonctionnent par plusieurs étapes. Elles permettent:

- De filtrer
- Définir l'ordre
- De grouper
- De transformer

MongoDB
Les agrégations

db.collection.aggregation(aggregation)



MongoDB
Les agrégations : stages

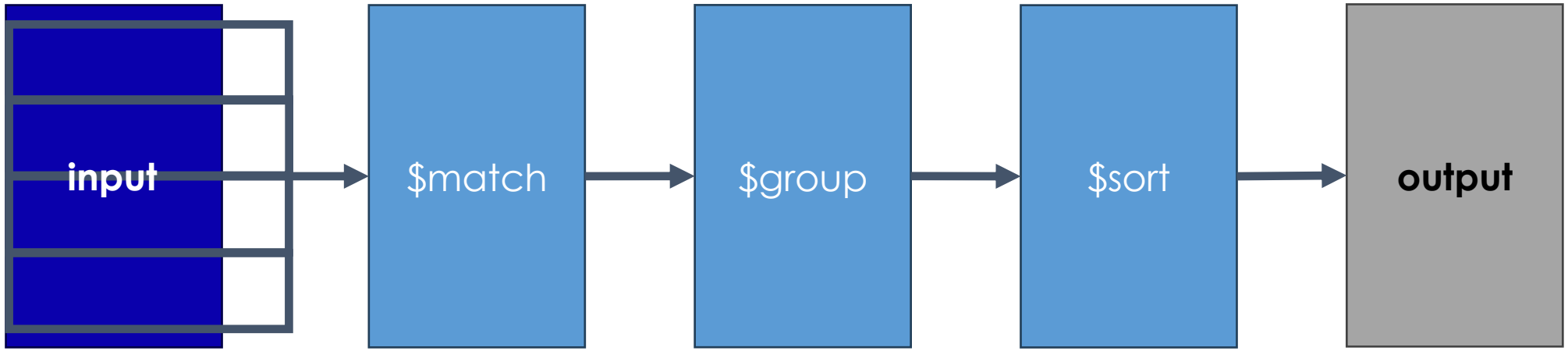
$match est la première étape de ces différant stage.
Il ressemble à **find()** en filtrant les documents désirer.

$group est la seconde étape qui permet de regroupé des documents selon des critère définit.

$sort est la dernière étape, permettant de retourner les documents dans l'ordre désiré.

MongoDB
Les agrégations : stages

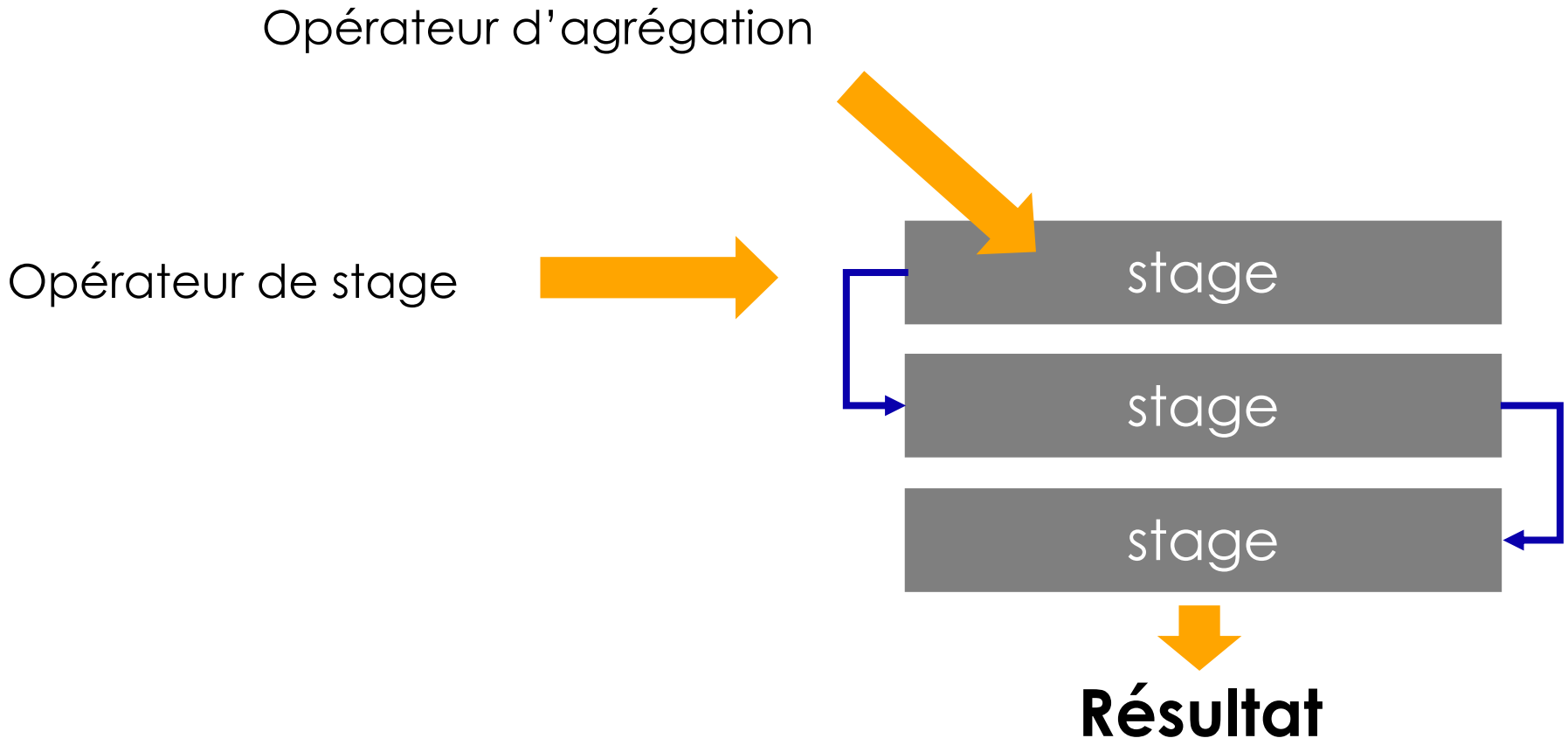
Voici un diagramme pour illustrer un pipeline d'agrégation MongoDB typique.



MongoDB
Les agrégations : syntaxe

```
db.collection.aggregate([
    {
        $stage1: {
            { expression1 },
            { expression2 }...
        },
        $stage2: {
            { expression1 }...
        }
    }
])
```

MongoDB
Les agrégations : opérateurs



MongoDB
Les opérateurs de stage

$match	$project	$group	$set
$sort	$count	$limit	$out

MongoDB
Les opérateurs de stage

$match	$project	$group	$set
$sort	$count	$limit	$out

MongoDB
L'opérateurs - atelier

Veuillez restaurer dans votre serveur MongoDB la base de données « **db_bird** » depuis l'archive « **db_bird.gz** » transmise par votre enseignant:

```
docker exec -i mongo mongorestore
    --uri=mongodb://root:admin@localhost:27017
    --authenticationDatabase=admin
    --drop --gzip --db=db_birds
    --archive=/backupdb/db_bird.gz
```

MongoDB
L'opérateurs: $match

L'étape **$match** filtre les documents qui correspondent aux conditions spécifiées. Voici le code pour **$match** :

```
{
  $match: {
     "field_name": "value"
  }
}
```

MongoDB
L'opérateurs: $group

L'étape **$group** regroupe les documents par une clé de groupe.

```
{
  $group:
    {
      _id: <expression>, // Group key
      <field>: { <accumulator> : <expression> }
    }
 }
```

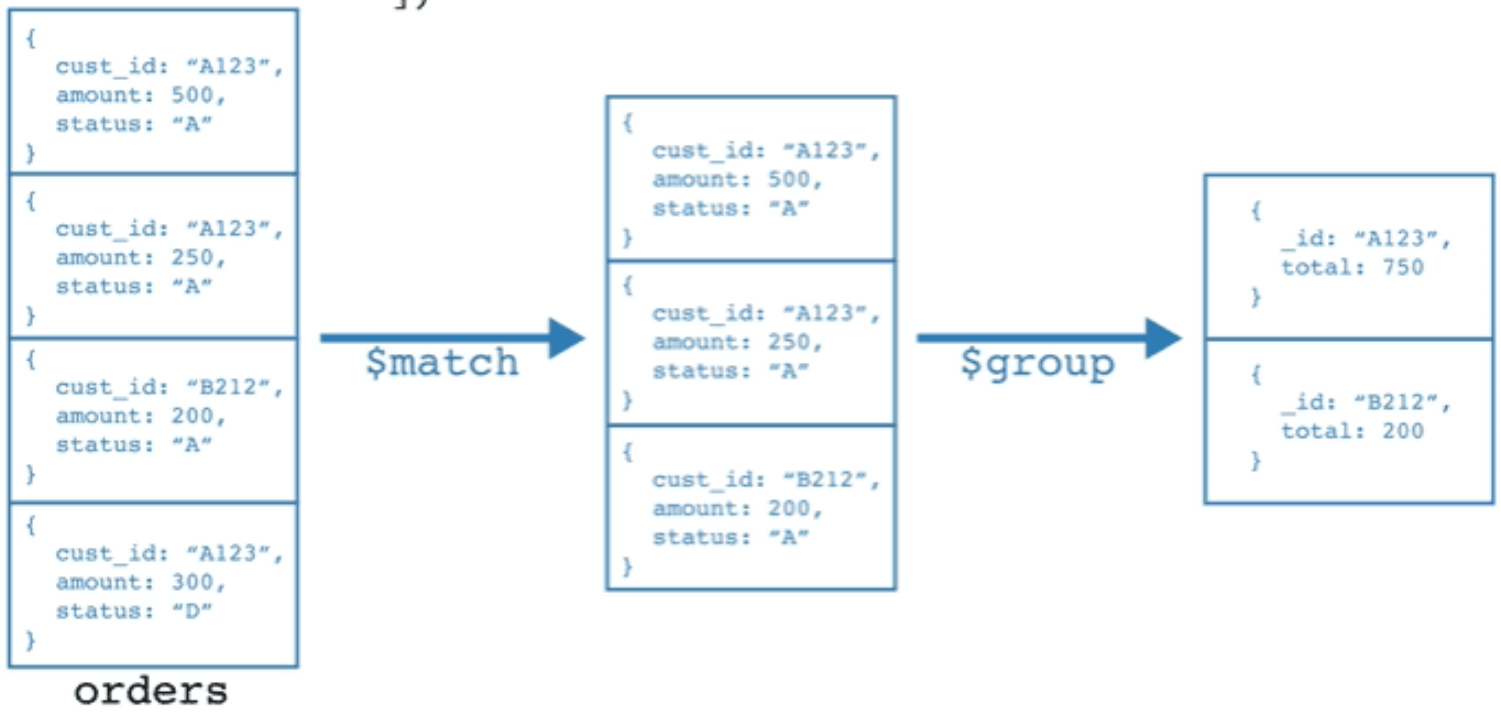
MongoDB
L'opérateur : $group

Opérateur	Signification
$count	Calcule la quantité de documents dans le groupe donné.
$max	Affiche la valeur maximale du champ d'un document dans la collection.
$min	Affiche la valeur minimale du champ d'un document dans la collection.
$avg	Affiche la valeur moyenne du champ d'un document dans la collection.
$sum	Résume les valeurs spécifiées de tous les documents de la collection.
$push	Ajoute des valeurs supplémentaires dans le tableau du document résultant.

MongoDB
Exemple : $match and $group

Collection

```
db.orders.aggregate( [
  $match stage  → { $match: { status: "A" } },
  $group stage  → { $group: { _id: "$cust_id",total:  {$sum: "$amount" } } }
                  ])
```



MongoDB
Exercice - $match et $group

Vous disposez d'une base de données appelée « **db_bird** » avec la collection « **sightings** ».

Nous souhaitons utiliser ces données pour savoir où nous devrions aller voir notre oiseau préféré, le Merlebleu de l'Est « Eastern Bluebird ».

Nous utiliserons les coordonnées du lieu (latitude et longitude) et le nombre d'observations à chaque endroit pour identifier les meilleurs endroits pour observer les Merlebleus de l'Est.

MongoDB
L'opérateurs: $sort

L'étape **$sort** trie tous les documents d'entrée et les renvoie au pipeline dans l'ordre trié. Nous utilisons **1** pour représenter l'ordre croissant et **-1** pour représenter l'ordre décroissant.

```
{
    $sort: {
        "field_name": 1
    }
}
```

MongoDB
L'opérateurs: $limit

L'étape **$limit** renvoie uniquement un nombre spécifié d'enregistrements.

```
db.zips.aggregate([
{
  $sort: {
    pop: -1
  }
},
{
  $limit:  5
}
])
```

MongoDB
Exercice - $sort et $limit

Vous disposez d'une base de données appelée « **db_bird** » avec la collection « **sightings** ».

Nous souhaitons trouver les oiseaux observés le plus au nord.

Dans ce cas, « location.coordinates.1 » sera le champ par lequel nous trierons. Dans un tableau de coordonnées, le premier élément (0) est la longitude et le deuxième élément (1) est la latitude. Les valeurs latitudinales sont plus grandes plus au nord, nous les trions donc de la plus élevée à la plus basse.

MongoDB
L'opérateurs: $project

L'étape **$project** spécifie les champs des documents de sortie. 1 signifie que le champ doit être inclus et 0 signifie que le champ doit être supprimé. Le champ peut également se voir attribuer une nouvelle valeur.

```
{
    $project: {
        state:1,
        zip:1,
        population:"$pop",
        _id:0
    }
}
```

MongoDB
Exercice - $projet

Vous disposez d'une base de données appelée « **db_bird** » avec la collection « **sightings** ».

Il y a beaucoup de données dans chaque document, mais nous souhaitons renvoyer uniquement une liste de l'heure « date » de l'observation et le nom commun « species_common » de l'oiseau observé

MongoDB
L'opérateurs: $set

L'étape **$set** crée de nouveaux champs ou modifie la valeur des champs existants, puis génère les documents avec les nouveaux champs.

```
{
    $set: {
        place: {
            $concat:["$city",",","$state"]
        },
        pop:10000
    }
}
```

MongoDB
Exercice - $set

Vous disposez d'une base de données appelée « **db_bird** » avec une collection d'informations sur les espèces d'oiseaux « **birds** ».

À l'avenir, nous ajouterons de nouveaux animaux à la base de données et à la collection. Avant de faire cela, nous devons ajouter le champ « classe » et définir ce champ à « bird » dans tous les documents existants de la collection .

MongoDB
L'opérateurs: $count

L'étape **$count** crée un nouveau document, avec le nombre de documents à cette étape dans le pipeline d'agrégation attribué au nom de champ spécifié.

```
{
  $count: "total_zips"
}
```

MongoDB
Exercice - $count

Vous disposez d'une base de données appelée « **db_bird** » avec une collection d'informations sur les espèces d'oiseaux « **sightings** ».

Nous souhaitons connaître le nombre total d'observations de Merlebleus de l'Est en 2022.

MongoDB
L'opérateurs: $out

L'étape **$out** crée une nouvelle collection, selon le résultat des étapes spécifiées dans le pipeline d'agrégation avec **$match** et **$group**. L'on peut aussi lui spécifié une autre base de données.

```
{
  $out: {
    db: "<db>",
    coll: "<newcollection>"
  }
}
```

MongoDB
D'autres opérateurs de stage

$unwind

$lookup

$skip

$geoNear

$addFields

MongoDB
Les opérateurs d'agrégations

Expression Arithmétique

$multiply	$divide	$add
$substract	$roud	

Tableau

$filter	$concatArray	$first
$last	$reduce	$map
$slice	$size	

Booléen

$and	$nor	$or

Comparaison

$eq	$ne	$lt
$lte	$gt	$gte

MongoDB
Les opérateurs d'agrégations

Condition

$cond $ifNull

Date

$toDate $month $week $year

Chaîne de caractères

$concat $toUpper $toLower $toString $substrCP

Accumulation (stage groupe)

$addToSet $push $avg $sum $min $max

MongoDB
Exemple 1

Opérations mathématiques

```
use('sample_mflix');

db.movies.aggregate([
  {
    $match: { genres: 'Fantasy', 'imdb.rating': { $ne: '' } }
  },
  {
    $project: {
      title: 1,
      averageRating: {
        $avg: [
          [ '$imdb.rating',
          { $multiply: [ '$tomatoes.viewer.rating', 2 ] } ]
        ]
      }
    }
  },
  { $sort: { averageRating: -1 } },
])
```

MongoDB
Exemple 2

Opérations de conversion, formatage, extraction de dates.

```
use('sample_mflix');

db.movies.aggregate(
  [
    {
      $project: {
        conversionDate: { $toDate: "2021-03-07T10:16:37.929Z" },
        year: { $year: "$released" },
        month: { $month: "$released" },
        dayOfMonth: { $dayOfMonth: "$released" },
        dayOfWeek: { $dayOfWeek: "$released" },
        weekNumber: {
          $isoWeek: {
            date: "$released",
            timezone: "Europe/Paris"
          }
        },
      }
    }
  ]
)
```

MongoDB
Exemple 3

Manipulation de conditions.

```
use('sample_mflix');

db.movies.aggregate(
  [
    {
      $addFields: {
        isGoodMovie: {
          $cond: {
            if: {
              $and: [
                { $gt: [ "$imdb.rating", 7 ] },
                { $gt: [ "$tomatoes.viewer.rating", 3.5 ] },
              ]
            },
            then: true,
            else: false
          }
        },
      }
    }
  ]
)
```

MongoDB
Exemple 4

Grouper et appliquer des opérations d'accumulation

```
use('sample_mflix');

db.movies.aggregate(
  [
    {
      $match: {
        year: { $type: 'int' },
        "imdb.rating": { $ne: "" }
      }
    },
    { $sort: { "imdb.rating": -1 } },
    {
      $group: {
        _id: '$year',
        nbrOfMovies: { $sum: 1 },
        bestMovieTitle: { $first: "$title" },
        worstMovieTitle: { $last: "$title" },
        bestMovieRating: { $max: "$imdb.rating" },
        worstMovieRating: { $min: "$imdb.rating" },
        averageMovieRating: { $avg: "$imdb.rating" }
      }
    },
    { $addFields:
      {
        averageMovieRating: { $round: ["$averageMovieRating", 2] }
      }
    },
    { $sort: { _id: -1 } },
  ]
)
```

MongoDB
Exemple 5

Les étapes **$sample**
et **$unwind**

```
use('sample_mflix');

db.movies.aggregate(
  [
    {
      $sample: { size: 5 }
    }
  ]
)
```

```
use('sample_mflix');

db.movies.aggregate(
  [
    {
      $unwind: "$genres"
    },
    {
      $group: {
        _id: "$genres",
        count: { $sum: 1 },
        avgRating: { $avg: "$imdb.rating" }
      }
    },
    {
      $sort: {
        avgRating: -1
      }
    }
  ]
)
```

MongoDB
Exemple 6

L'opérateur d'étape **$lookup** permet de récupérer le ou les documents d'une même base de données, mais dans une autre collection, correspondant à des identifiants.

C'est très utile quand on associe des documents de collections différentes par **référence**.

```
use('sample_mflix');

db.comments.aggregate(
  [
    {
      $lookup: {
        from: "movies",
        localField: "movie_id",
        foreignField: "_id",
        as: "movie"
      }
    },
  ]
)
)
```

MongoDB
Agrégation: plus

Références

Tous sur les étapes / stages

Tableau de comparaison avec le language SQL