GENERALITAT VALENCIANA ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

UD8: BASES DE DATOS NOSQL

Prácticas no evaluables
Parte 3. Consultas Avanzadas

Bases de Datos (BD)
CFGS DAM/DAW

Abelardo Martínez y Pau Miñana.

Curso 2023-2024

Aspectos a tener en cuenta

Estas actividades son opcionales y no evaluables pero es recomendable hacerlas para un mejor aprendizaje de la asignatura.

> 🚫 **Si buscas las soluciones por Internet o preguntas al oráculo de ChatGPT, te estarás engañando a ti mismo**. Ten en cuenta que ChatGPT no es infalible ni todopoderoso. Es una gran herramienta para agilizar el trabajo una vez se domina una materia, pero usarlo como atajo en el momento de adquirir habilidades y conocimientos básicos perjudica gravemente tu aprendizaje.
>
> Si lo utilizas para obtener soluciones o asesoramiento respecto a las tuyas, revisa cuidadosamente las soluciones propuestas igualmente. Intenta resolver las actividades utilizando los recursos que hemos visto y la documentación extendida que encontrarás en el "Aula Virtual".

> 📖 **Recomendaciones**:
>
> - **No uses NUNCA tildes, ni eñes, ni espacios, ni caracteres no alfanuméricos** (salvo el guión bajo) **en los metadatos** (nombres de elementos de una base de datos) y **usa letras minúsculas**.
> - Sé coherente con el uso de mayúsculas/minúsculas.

ÍNDICE

1. Base Imperial

Vamos a modelar los pilotos y misiones de una base imperial, teniendo en cuenta el número de naves enemigas abatidas.

Para ello, necesitamos una base de datos en MongoDB llamada *imperialbaseMDB*, con una colección llamada *pilotos* que incluya los siguientes campos con los siguientes valores:

piloto_id	nombre	apellidos	mision	abatidos
pr01	Lord	Darth Vader	ARRAY	10
pr02	Moff	Tarkin		
pr03	Almirante	Thrawn	ARRAY	
pr04	Biggs	Darklighter	ARRAY	5
pr05	Barón	Soontir Fel	ARRAY	2
pr06	Maarek	Stele	ARRAY	3
pr07	Mulchive	Wermis	ARRAY	6
pr08	Loka	Hask	ARRAY	7
pr09	Tycho	Celchu	ARRAY	4
pr10	PT001	Piloto DS-61-1		3

El array de *mision* contendrá documentos con las misiones realizadas por cada piloto, la nave usada y su coste:

idpiloto	cod	nave	coste
pr01			
	MA001	Crucero imperial nodriza	5
	MI003	Crucero imperial nodriza	7.16
	ME004	Tiefighter Mustafar	2.33
pr03			
	MT002	Crucero logístico	6.58
pr04			
	MI003	Tiefighter	10.24
	ME004	Crucero de transporte	7.89
pr05			
	MA001	Tiefighter	3.25
	ME004	Crucero de transporte	1.76
pr06			
	MA001	Tiefighter	5.01
	ME004	Crucero de transporte	2.43
pr07			
	MA001	Tiefighter	4
	ME004	Crucero de transporte	3.29
pr08			
	MA001	Tiefighter	3.29
	ME004	Crucero de transporte	4.05
pr09			
	MA001	Tiefighter	2.57
	ME004	Crucero de transporte	1.65

El script para crear esa base de datos es el siguiente:

```
use imperialbaseMDB
db.createCollection("pilotos")

var p1 = {_id: "pr01", nombre: "Lord", apellidos: "Darth Vader",
    mision:[
    {cod:"MA001", nave: "Crucero imperial nodriza", coste: 5},
    {cod:"MI003", nave: "Crucero imperial nodriza", coste: 7.16},
    {cod:"ME004", nave: "Tiefighter Mustafar", coste: 2.33}],
    abatidos: 10}
var p2 = {_id: "pr02", nombre: "Moff", apellidos: "Tarkin"}
var p3 = {_id: "pr03", nombre: "Almirante", apellidos: "Thrawn",
    mision:{cod:"MT002", nave: "Crucero logístico", coste:6.58}}
var p4 = {_id: "pr04", nombre: "Biggs", apellidos: "Darklighter",
    mision:[
    {cod:"MI003", nave: "Tiefighter", coste: 10.24},
    {cod:"ME004", nave: "Crucero de transporte", coste: 7.89}],
    abatidos: 5}
var p5 = {_id: "pr05", nombre: "Barón", apellidos: "Soontir Fel",
    mision:[
    {cod:"MA001", nave: "Tiefighter", coste: 3.25},
    {cod:"ME004", nave: "Crucero de transporte", coste: 1.76}],
    abatidos: 2}
var p6 = {_id: "pr06", nombre: "Maarek", apellidos: "Stele",
    mision:[
    {cod:"MA001", nave: "Tiefighter", coste: 5.01},
    {cod:"ME004", nave: "Crucero de transporte", coste: 2.43}],
    abatidos: 3}
var p7 = {_id: "pr07", nombre: "Mulchive", apellidos: "Wermis",
    mision:[
    {cod:"MA001", nave: "Tiefighter", coste: 4},
    {cod:"ME004", nave: "Crucero de transporte", coste: 3}],
    abatidos: 6}
var p8 = {_id: "pr08", nombre: "Loka", apellidos: "Hask",
    mision:[
    {cod:"MA001", nave: "Tiefighter", coste: 3.29},
    {cod:"ME004", nave: "Crucero de transporte", coste: 4.05}],
    abatidos: 7}
var p9 = {_id: "pr09", nombre: "Tycho", apellidos: "Celchu",
    mision:[
    {cod:"MA001", nave: "Tiefighter", coste: 2.57},
    {cod:"ME004", nave: "Crucero de transporte", coste: 1.65}],
    abatidos: 4}
var p10 = {_id: "pr10", nombre: "PT001", apellidos: "DS-61-1",
    abatidos: 3}

db.pilotos.insertMany([p1, p2, p3, p4, p5, p6, p7, p8, p9, p10])
```

Crea los scripts necesarios para realizar las siguientes tareas:

1.1. Script 1: Contar pilotos

Contar el número de pilotos que tienen *abatidos*, o bien su *nave* es "Tiefighter Mustafar" o han realizado alguna misión.

Solución

```
/* La primera condición se puede interpretar de 2 modos diferentes
   que exista el campo, o que sea mayor que 0 */
// var j_valor1 = {abatidos: {$exists: 1}}
var j_valor1 = {"abatidos": {"$gt": 0}}
var j_valor2 = {"mision.nave": "Tiefighter Mustafar"}
var j_valor3 = {"mision": {"$exists": 1}}
var j_filtro = {"$or": [j_valor1, j_valor2, j_valor3]}
db.pilotos.countDocuments(j_filtro)
/* db.pilotos.countDocuments({$or: [{abatidos: {$gt: 0}},
                        {mision: {$exists: 1}},
                        {"mision.nave": "Tiefighter Mustafar"}]}) */
```

```
imperialbaseMDB> db.pilotos.countDocuments(j_filtro)
9
```

1.2. Script 2: Datos misiones con coste mayor que 5

Listar los datos de las misiones (*nombre*, *apellidos* del piloto, *cod*, *nave* y *coste*) con *coste* superior a 5, ordenados por nombre y apellidos del piloto. Para los costes se consideran las misiones individualmente por cada piloto, no el total.

Solución

```
var st_unw   = {"$unwind": "$mision"}
var j_cond   = {"mision.coste": {"$gt": 5}}
var st_match = {"$match": j_cond}
var j_proy   = {"nombre": 1, "apellidos": 1, "mision": 1}
// Si queremos la consulta a prueba de cambios en los documentos
/* var j_proy = {"nombre": 1, "apellidos": 1, "mision.cod": 1,
                 "mision.nave": 1, "mision.coste": 1} */
var st_proy  = {"$project": j_proy}
var j_orden = {"nombre": 1, "apellidos": 1}
var st_sort = {"$sort": j_orden}
db.pilotos.aggregate([st_unw, st_match, st_proy, st_sort])
/* db.pilotos.aggregate([{$unwind: "$mision"},
                        {$match: {"mision.coste": {$gt: 5}}},
                        {$project: {nombre: 1, apellidos: 1,
                                    "mision": 1}},
                        {$sort: {nombre: 1, apellidos: 1}}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_unw, st_match, st_proy, st_sort])
[
  {
    _id: 'pr03',
    nombre: 'Almirante',
    apellidos: 'Thrawn',
    mision: { cod: 'MT002', nave: 'Crucero logístico', coste: 6.58 }
  },
  {
    _id: 'pr04',
    nombre: 'Biggs',
    apellidos: 'Darklighter',
    mision: { cod: 'MI003', nave: 'Tiefighter', coste: 10.24 }
  },
  {
    _id: 'pr04',
    nombre: 'Biggs',
    apellidos: 'Darklighter',
    mision: { cod: 'ME004', nave: 'Crucero de transporte', coste: 7.89 }
  },
  {
    _id: 'pr01',
    nombre: 'Lord',
    apellidos: 'Darth Vader',
    mision: { cod: 'MI003', nave: 'Crucero imperial nodriza', coste: 7.16 }
  },
  {
    _id: 'pr06',
    nombre: 'Maarek',
    apellidos: 'Stele',
    mision: { cod: 'MA001', nave: 'Tiefighter', coste: 5.01 }
  }
]
```

> 📖 Como se puede observar en las anteriores soluciones, se pueden usar comillas o no para los nombres de los campos y fases. En general, es recomendable usarlas para evitar problemas de interpretación; además en caso de objetos embebidos, es obligatorio usarlas, con lo se evitan errores.

Por cuestión del resaltado automático de la sintaxis, se ha optado por no usarlas a partir de aquí cuando no sea estrictamente necesario de modo que se vea mejor la estructura de las consultas, pero no es la opción más recomendable.

1.3. Script 3: Datos de las misiones con coste total mayor que 20

Se requiere listar las misiones(*cod*) y el coste total que cada una ha tenido, para las misiones con coste total mayor que 20, ordenadas por coste total de mayor a menor.

Solución

```
var st_unw   = {$unwind: "$mision"}
var j_group  = {_id: "$mision.cod",
                coste_total: {$sum: "$mision.coste"}}
var st_group = {$group: j_group}
var j_cond   = {coste_total: {$gt: 20}}
var st_match = {$match: j_cond}
var j_proy   = {mision:"$_id", coste_total: 1, _id: 0}
var st_proy  = {$project: j_proy}
var j_orden  = {coste_total: -1}
var st_sort  = {$sort: j_orden}
db.pilotos.aggregate([st_unw, st_group, st_match, st_proy, st_sort])
/* db.pilotos.aggregate([{$unwind: "$mision"},
                         {$group: {_id: "$mision.cod",
                                   tot: {$sum: "$mision.coste"}}},
                         {$match: {tot: {$gt: 20}}},
                         {$project: {mision:"$_id", tot: 1, _id: 0}},
                         {$sort: {coste_total: -1}}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_unw, st_group, st_match, st_proy, st_sort])
[
  { coste_total: 23.12, mision: 'MA001' },
  { coste_total: 23.11, mision: 'ME004' }
]
```

1.4. Script 4: Costes por piloto

Se requiere listar los pilotos (*nombre* y *apellidos*) y los costes totales que han generado entre todas sus misiones, ordenados de menor a mayor coste.

Solución

```
/* Se puede añadir un filtro para eliminar los pilotos sin misiones
    y que no salgan con coste 0 */
// var st_match = {$match: {mision: {$exists: 1}}}
var j_proy   = {_id: 0, nombre: 1, apellidos: 1,
                coste_total: {$sum: "$mision.coste"}}
var st_proy  = {$project: j_proy}
var j_orden  = {coste_total: 1}
var st_sort  = {$sort: j_orden}
db.pilotos.aggregate([st_proy, st_sort])
/* db.pilotos.aggregate([{$project: {_id: 0, nombre: 1, apellidos: 1,
                                     tot: {$sum: "$mision.coste"}}},
                         {$sort: {tot: 1}}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_proy, st_sort])
[
  { nombre: 'Moff', apellidos: 'Tarkin', coste_total: 0 },
  { nombre: 'PT001', apellidos: 'DS-61-1', coste_total: 0 },
  { nombre: 'Tycho', apellidos: 'Celchu', coste_total: 4.22 },
  { nombre: 'Barón', apellidos: 'Soontir Fel', coste_total: 5.01 },
  { nombre: 'Almirante', apellidos: 'Thrawn', coste_total: 6.58 },
  { nombre: 'Mulchive', apellidos: 'Wermis', coste_total: 7 },
  { nombre: 'Loka', apellidos: 'Hask', coste_total: 7.34 },
  {
    nombre: 'Maarek',
    apellidos: 'Stele',
    coste_total: 7.4399999999999995
  },
  { nombre: 'Lord', apellidos: 'Darth Vader', coste_total: 14.49 },
  { nombre: 'Biggs', apellidos: 'Darklighter', coste_total: 18.13 }
]
```

En esta unidad no se ha considerado el tipo de dato en las inserciones numéricas, y los números con decimales se insertan por defecto como *double* en MongoDB. Esto ha provocado un error de las operaciones en coma flotante para los costes de "Maarek Stele". Si usamos Compass para cambiar gráficamente el tipo de dato a *decimal128*, se solucionaría el problema.

1.5. Script 5: Piloto que más costes ha generado

Se requiere listar el *nombre*, *apellidos* y coste total del piloto que más ha costado, considerando todas las misiones.

Solución

```
// Usando la proyección de la consulta anterior
var j_orden  = {coste_total: -1}
var st_sort  = {$sort: j_orden}
var st_limit = {$limit: 1}
db.pilotos.aggregate([st_proy, st_sort, st_limit])
/* db.pilotos.aggregate([{$project: {_id: 0, nombre: 1, apellidos: 1,
                                     tot: {$sum: "$mision.coste"}}},
                         {$sort: {tot: -1}},
                         {$limit: 1}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_proy, st_sort, st_limit])
[ { nombre: 'Biggs', apellidos: 'Darklighter', coste_total: 18.13 } ]
```

1.6. Script 6: Abatidos

a) Se requiere contar el número total de *abatidos* por todas las tropas imperiales

Solución

```
var j_group = {_id: null, total: {$sum: "$abatidos"}}
var st_group = {$group: j_group}
var j_proy  = {_id: 0}
var st_proy = {$project: j_proy}
db.pilotos.aggregate([st_group, st_proy])
/* db.pilotos.aggregate([{$group: {_id: null,
                                   total: {$sum: "$abatidos"}}},
                         {$project: {_id: 0}}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_group, st_proy])
[ { total: 40 } ]
```

b) Se requiere el *nombre*,*apellidos* y *abatidos* del piloto que más naves enemigas ha abatido.

Solución

```
var j_proy   = {_id: 0, nombre: 1, apellidos: 1, abatidos: 1}
var j_orden  = {abatidos: -1}
db.pilotos.find({},j_proy).sort(j_orden).limit(1)
/* db.pilotos.find({}, {nombre: 1, apellidos: 1,
   abatidos: 1}).sort({abatidos: -1}).limit(1) */
/* Con aggregate
var st_proy  = {$project: j_proy}
var st_sort  = {$sort: j_orden}
var st_limit = {$limit: 1}
db.pilotos.aggregate([st_proy, st_sort, st_limit])
o
db.pilotos.aggregate([{$project: {_id: 0, nombre: 1, apellidos: 1,
                                  abatidos: 1}},
                      {$sort: {abatidos: -1}},
                      {$limit: 1}]) */
```

```
imperialbaseMDB> db.pilotos.find({},j_proy).sort(j_orden).limit(1)
[ { nombre: 'Lord', apellidos: 'Darth Vader', abatidos: 10 } ]
```

1.7. Script 7: Abatidos por tipo nave

Se requiere listar el nombre de cada tipo de *nave* y el total de *abatidos* que se han conseguido con la misma, independientemente del piloto/misión. ¿Qué error contiene esta consulta en su planteamiento?

Solución

```
var st_unw   = {$unwind: "$mision"}
var j_group  = {_id: "$mision.nave",
                total: {$sum: "$abatidos"}}
var st_group = {$group: j_group}
var j_proy   = {_id: 0, nave: "$_id", total: 1}
var st_proy  = {$project: j_proy}
db.pilotos.aggregate([st_unw, st_group, st_proy])
/* db.pilotos.aggregate([{$unwind: "$mision"},
                         {$group: {_id: "$mision.nave",
                                   total: {$sum: "$abatidos"}}},
                         {$project: {_id: 0, nave: "$_id",
                                     total: 1}}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_unw, st_group, st_proy])
[
  { total: 10, nave: 'Tiefighter Mustafar' },
  { total: 20, nave: 'Crucero imperial nodriza' },
  { total: 27, nave: 'Tiefighter' },
  { total: 0, nave: 'Crucero logístico' },
  { total: 27, nave: 'Crucero de transporte' }
]
```

El error de la consulta es que se está sumando el total de abatidos por cada piloto, no por cada nave. Con lo que el total de abatidos es superior al real. Los datos en el estado en el que están no permiten saber cuántos abatidos ha tenido cada nave, ya que no se detalla en qué misiones se han conseguido esos abatidos.

Que una consulta devuelva resultados y estos respondan a alguna lógica no significa que sean correctos.

1.8. Script 8: Costes por tipo nave

Se requiere listar el nombre de cada tipo de *nave* y el total de costes generados con la misma, independientemente del piloto/misión.

Solución

```
var st_unw   = {$unwind: "$mision"}
var j_group  = {_id: "$mision.nave",
                total: {$sum: "$mision.coste"}}
var st_group = {$group: j_group}
var j_proy   = {_id: 0, nave: "$_id", total: 1}
var st_proy  = {$project: j_proy}
db.pilotos.aggregate([st_unw, st_group, st_proy])
/* db.pilotos.aggregate([{$unwind: "$mision"},
                         {$group: {_id: "$mision.nave",
                                   total: {$sum: "$mision.coste"}}},
                         {$project: {_id: 0, nave: "$_id",
                                     total: 1}}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_unw, st_group, st_proy])
[
  { total: Decimal128('6.58'), nave: 'Crucero logístico' },
  { total: Decimal128('20.78'), nave: 'Crucero de transporte' },
  { total: Decimal128('12.16'), nave: 'Crucero imperial nodriza' },
  { total: Decimal128('2.33'), nave: 'Tiefighter Mustafar' },
  { total: Decimal128('28.36'), nave: 'Tiefighter' }
]
```

Los resultados pueden diferir ligeramente si no se han cambiado los costes a *decimal128* por el problema de los decimales en coma flotante.

1.9. Script 9: Tipo de nave con más costes

Se requiere listar el nombre del tipo de *nave* que ha generado más costes entre todas sus misiones.

Solución

```
/* Con las fases de la consulta anterior, no se usa la proyección
   esta vez, pero sería perfectamente válido */
var j_orden  = {total: -1}
var st_sort  = {$sort: j_orden}
var st_limit = {$limit: 1}
db.pilotos.aggregate([st_unw, st_group, st_sort, st_limit])
/* db.pilotos.aggregate([{$unwind: "$mision"},
                         {$group: {_id: "$mision.nave",
                                   total: {$sum: "$mision.coste"}}},
                         {$sort: {total: -1}},
                         {$limit: 1}]) */
```

```
imperialbaseMDB> db.pilotos.aggregate([st_unw, st_group, st_sort, st_limit])
[ { _id: 'Tiefighter', total: 28.36 } ]
```

1.10. Script 10: Misiones en planetas boscosos

Considerando la siguiente colección de planetas:

```
use imperialbaseMDB
db.createCollection("planetas")

var j_plan1 = {_id: "PL01", nombre: "Endor", tipo: "boscoso",
    misiones: ["MA001"]}
var j_plan2 = {_id: "PL02", nombre: "Tatooine", tipo: "desértico"}
var j_plan3 = {_id: "PL03", nombre: "Yavin IV", tipo: "boscoso",
    misiones: ["MI003", "ME004"]}
var j_plan4 = {_id: "PL04", nombre: "Wayland", tipo: "montañoso",
    misiones: ["MT002"]}

db.planetas.insertMany([j_plan1, j_plan2, j_plan3, j_plan4])
```

Se requiere listar el *nombre*,*apellidos* y *cod* de misión de los pilotos que han realizado misiones en planetas boscosos, así como el nombre del *planeta*.

Solución

```
var st_unw   = {$unwind: "$mision"}
var  j_cond  = {"mision.planeta.tipo": "boscoso"}
var st_match = {$match: j_cond}
var j_join   = {from: "planetas", localField: "mision.cod",
               foreignField: "misiones", as: "mision.planeta"}
var st_join  = {$lookup: j_join}
var j_proy   = {_id: 0, nombre: 1, apellidos: 1, "mision.cod": 1,
                "mision.planeta.nombre": 1}
var st_proy  = {$project: j_proy}
db.pilotos.aggregate([st_unw, st_join, st_match, st_proy])
/* db.pilotos.aggregate([{$unwind: "$mision"},
                         {$lookup: {from: "planetas",
                                    localField: "mision.cod",
                                    foreignField: "misiones",
                                    as: "mision.planeta"}},
                         {$match: {"mision.planeta.tipo":"boscoso"}},
                         {$project: {_id: 0, nombre: 1, apellidos: 1,
                                     "mision.cod": 1,
                                     "mision.planeta.nombre": 1}}]) *
```

```
imperialbaseMDB> db.pilotos.aggregate([st_unw, st_join, st_match, st_proy])
[
  {
    nombre: 'Lord',
    apellidos: 'Darth Vader',
    mision: { cod: 'MA001', planeta: [ { nombre: 'Endor' } ] }
  },
  {
    nombre: 'Lord',
    apellidos: 'Darth Vader',
    mision: { cod: 'MI003', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Lord',
    apellidos: 'Darth Vader',
    mision: { cod: 'ME004', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Biggs',
    apellidos: 'Darklighter',
    mision: { cod: 'MI003', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Biggs',
    apellidos: 'Darklighter',
    mision: { cod: 'ME004', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Barón',
    apellidos: 'Soontir Fel',
    mision: { cod: 'MA001', planeta: [ { nombre: 'Endor' } ] }
  },
  {
    nombre: 'Barón',
    apellidos: 'Soontir Fel',
    mision: { cod: 'ME004', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Maarek',
    apellidos: 'Stele',
    mision: { cod: 'MA001', planeta: [ { nombre: 'Endor' } ] }
  },
  {
    nombre: 'Maarek',
    apellidos: 'Stele',
    mision: { cod: 'ME004', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Mulchive',
    apellidos: 'Wermis',
    mision: { cod: 'MA001', planeta: [ { nombre: 'Endor' } ] }
  },
  {
    nombre: 'Mulchive',
    apellidos: 'Wermis',
    mision: { cod: 'ME004', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Loka',
    apellidos: 'Hask',
    mision: { cod: 'MA001', planeta: [ { nombre: 'Endor' } ] }
  },
  {
    nombre: 'Loka',
    apellidos: 'Hask',
    mision: { cod: 'ME004', planeta: [ { nombre: 'Yavin IV' } ] }
  },
  {
    nombre: 'Tycho',
    apellidos: 'Celchu',
    mision: { cod: 'MA001', planeta: [ { nombre: 'Endor' } ] }
  },
  {
    nombre: 'Tycho',
    apellidos: 'Celchu',
    mision: { cod: 'ME004', planeta: [ { nombre: 'Yavin IV' } ] }
  }
]
```