Bases de No Datos Relacionales

- **ARRAYS**:

  Operaciones : insert, update y delete

  Proyección : find

- **Operadores sobre Arrays**
  - **$push** Nos permite **insertar** elementos *repetidos* en un array. Si el array no existe lo crea
  - **$addtoSet** Nos permite **insertar** elementos *únicos* en un array. Si el array no existe lo crea
  - **$pop** **Elimina** un elemento de un Array por sus extremos, permitiendo eliminar el primer elemento (-1) o el último (1).
  - **$pull** **Elimina** todos los elementos de un Array que coincidan con una expresión de filtro o condición.
  - **$pullAll** **Elimina** todos los elementos de un Array que contengan alguno de los valores indicados.
  - **$[<identifier>]** Identifica los elementos de un array que coinciden con las condiciones establecidas en el parámetro *arrayFilters* para realizar una operación de **modificación**.

- **Modificadores de operadores sobre arrays**

$each	Disponible para su uso con la operador **$addToSet** y el operador **$push**. Permite agregar múltiples valores en un array.
$slice	Limita el número de elementos del array durante una operación **$push**. Para usar el modificador $slice, debe aparecer con el modificador $each. Se puede pasar un array vacío [] al modificador $each de modo que solo el modificador $slice tenga efecto.
$sort	El modificador $sort ordena los elementos de un array durante una operación **$push**.Para usar el modificador $sort, debe aparecer con el modificador $each. Puede pasar un array vacío [] al modificador $each de modo que solo el modificador $sort tenga efecto.
$position	Especifica la ubicación del array en la que el operador **$push** inserta elementos. Sin el modificador $position, el operador $push inserta elementos al final del array.

- **Sintaxis y ejemplos**

- **Sin modificadores** { **$push**: { <field1>: <value1>, ... } }
- **Con modificadores** { **$push**: { <field1>: { **<modifier1>**: <value1>, **<modifier2>**: <value2>,... }, ... } }

Supongamos que trabajamos sobre estas dos colecciones con estos documentos

```
db.students.insertMany( [
{  id: 1, scores: [ 45, 72, 36, 87 ] } ,
{  id: 2, scores: [ 45, 78, 38, 80, 89 ] } ,
{  id: 3, scores: [ 47, 78, 38, 80, 89 ] }
 ])
db.students1.insertMany( [
{  id: 4, scores: [{Nota:5,Concepto:"Muy Bueno"},{Nota:10,Concepto:"Excelente"},{Nota:9,Concepto:"Muy Bueno"}]},
{ _id: 5, scores: [{Nota:6,Concepto:"Bueno" },{Nota:2,Concepto:"Malo"},{Nota:4,Concepto:"Bueno"}]},
{ _id: 6, scores: [{Nota:3,Concepto:"Muy Bueno"},{Nota:5,Concepto:"Muy Bueno"}]}
])
```

- **Sintaxis y ejemplos**

Luego las operaciones siguientes darán como resultado:

```
db.students.updateMany({ },{ $push: { scores: 95}})
```
Agregar el valor 95 en scores en los tres documentos

```
db.students.updateOne({ id: 1},{ $push: {scores:{$each:[90,92,85]}}})
```
Agregar los tres valores al documento con _id:1

```
db.students.updateOne({ id: 2},{$addToSet: {scores:{$each:[45,92,85]}}})
```
Insertar solo 92 y 85, al _id:2 ya que el 45 ya existe.

```
db.students.updateOne({ id: 3},{$push: {scores: {$each: [50,60,70],$position: 0}}})
```
Insertar los valores 50,60,70 al inicio del array scores, al _id:3.

- **Sintaxis y ejemplos**

```
db.students.updateOne
(
     {  id: 5 },
     {$push: {scores: {
          $each: [{Nota: 5, Concepto: "Muy Bueno" }, {Nota: 10, Concepto: "Excelente"]},
          $sort: { Nota: -1 },
          $slice: 3
     }}}
)
```

Agrega los dos valores al documento con _id:5, luego ordena scores de manera descendente por Nota y deja solo las tres primeras notas.

- **Sintaxis y ejemplos**
  - { **$pop:** { <field>: <-1 | 1>, ... } }

Ejemplos de pop

```
db.students.updateMany({_id:1 },{$pop: {scores: 1 }})
db.students.updateMany({_id:1 },{$pop: {scores: -1}})
```

  - { **$pull**: { <field1>: <value|condition>, <field2>: <value|condition>, ... } }

Ejemplos de pull con valor y con condiciones

Sobre un array de elementos…

```
db.students.updateMany({_id:3 },{$pull: {scores: 78 }})
db.students.updateMany({_id:2 },{$pull: {scores: {$in: [38,58] }}})
db.students.updateMany({_id:1 },{$pull: { scores: { $gte: 90 }}})
```

Sobre un array de documentos…

```
db.students.updateMany({_id:4},{$pull: {scores: {Nota: 6, Concepto:"Muy Bueno"}}})
```

- ## Sintaxis y ejemplos

  - { **$pullAll**: { <field1>: [<value1>, <value2> ...], ... } }

Ejemplos

Sobre un array de elementos…

```
db.students.updateMany({_id:3 },{$pullAll: {scores: [47,78,90]}})
```

Sobre un array de documentos…

```
db.students.updateMany({_id:4},{$pullAll: {scores: [{Nota: 5, Concepto:"Muy Bueno"}]}})
```

- ## Sintaxis y ejemplos
  - ### $[<identifier>] **

```
{ <update operator>: { "<array>.$[<identifier>]" : value } },
{ arrayFilters: [ { <identifier>: <condition> } ] }
```

Ejemplos:

Sobre un array de elementos…

```
db.students.updateMany(
   { },
   { $set: { "scores.$[identificador]" : 90}},
   { arrayFilters: [{"identificador": {$gt: 90}}]}
)
```

Sobre un array de documentos…

```
db.students.updateMany(
{},
{$set:{ "scores.$[elem].Concepto" : "Excelente" } },
{arrayFilters:[{"elem.Nota": { $eq: 10}}]}
)
```

** <identifier> Debe comenzar con una letra minúscula y contener sólo caracteres alfanuméricos.

- **Operadores de proyección sobre arrays** (se utilizan con el ***find***)

- **$elementMatch** Para especificar varios criterios en un array de documentos embebidos, de modo que al menos **un documento satisfaga todos los criterios especificados**.

¿Qué consulta trae el resultado correcto, si estoy buscando un estudiante con Nota 5 y Excelente concepto?

```
db.students.find({"scores.Nota":5, "scores.Concepto":"Excelente" })

db.students.find({scores:{$elemMatch:{Nota:5,Concepto:"Excelente"}}})

db.students.find({scores:{$elemMatch:{Nota:5,Concepto:"Muy Bueno"}}})
```

- **Operadores de proyección sobre arrays** (se utilizan con el ***find***)

- **$slice** El operador $slice, se utiliza para proyectar especificando el número de elementos de un array que se va a devolver en el resultado de una consulta.

```
db.students.find({},{ scores:{$slice: 1}})Muestra la primera nota de cada estudiante
```

- **$** Nos permite hacer la **proyección** del primer elemento del array que cumpla con la condición especificada, y también permite **modificar** el primer elemento del array que cumpla con la condición especificada. Es decir, el mismo que es mostrado al hacer la proyección

```
db.students.find({scores:{$elemMatch:{Nota:5,Concepto:"Muy Bueno"}}},{"scores.$":1})

db.students.update({scores:{$elemMatch:{Nota:5,Concepto:"Muy Bueno"}}},
                   {$set:{"scores.$.fecha":new date()}})
```

- **Operadores de proyección sobre arrays** (se utilizan con el ***find***)

- **$all** Para devolver documentos en los que el valor de un campo es una array que contiene **todos** los elementos especificados en el query (puede contener además otros).

```
db.students.find( { scores: { $all: [90,92]}})
```

- **dot notation y un numero:** Para consultar por una posición en un array

```
db.students.find({ 'scores.1': { $eq: 92 }})
```

- **$size** Para consultar por el tamaño de un array:

```
db.students.find({scores: { $size: 3 }})
```