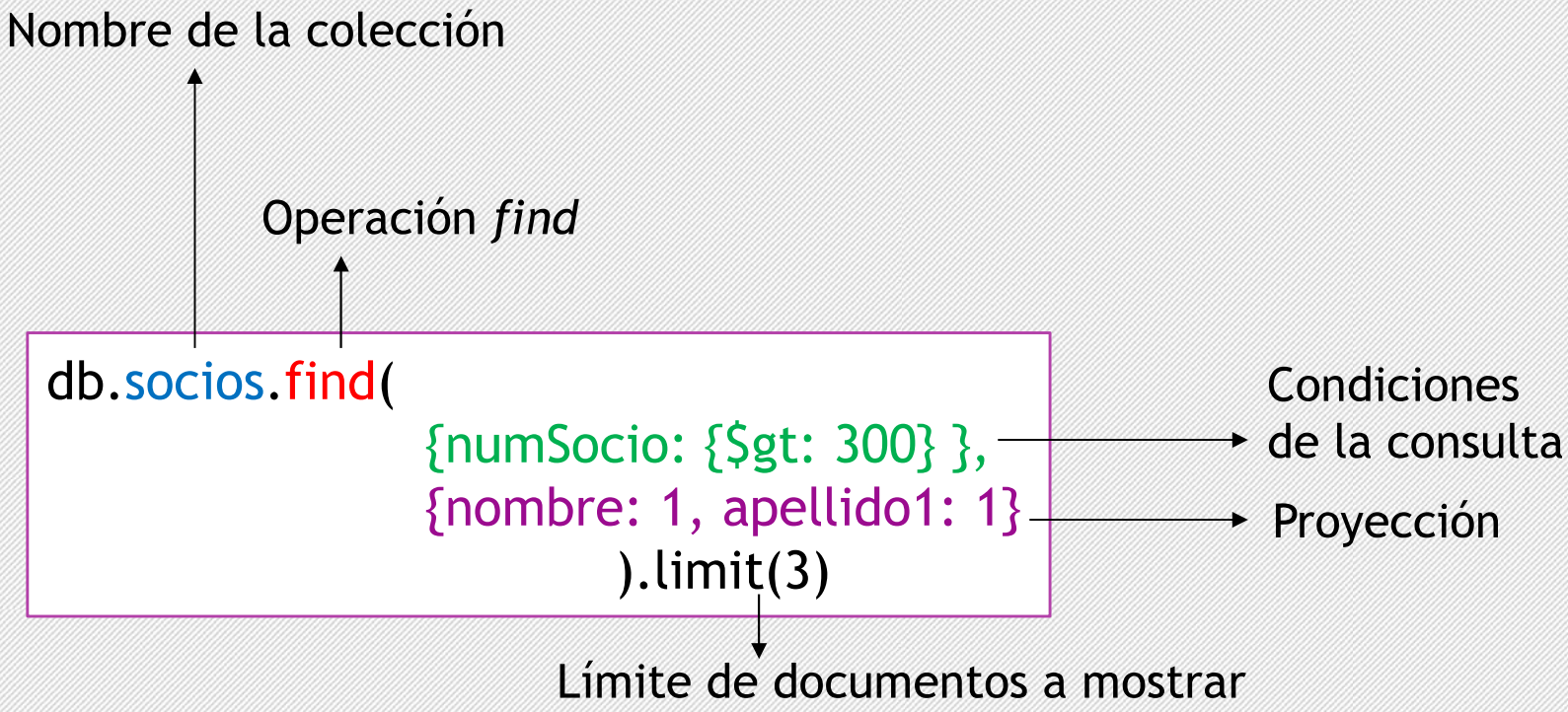
Consulta y modificación de datos: operaciones CRUD

- Los datos son accedidos mediante operaciones *CRUD* (*Create*, *Read*, *Update*, *Delete*):
  - Find: lectura (Read) de datos. Símil con el SELECT en SQL.
  - Insert: escritura (Create) de datos.
  - Update: actualización de valores en los datos.
  - Remove: borrado (Delete) de datos.
- La ejecución de cualquiera de estas instrucciones sigue el siguiente esquema:

```
db.<nombre_colección>.<nombre_operación>(<condiciones de la operación>)
```

Consultas: operación *find*

- Ejemplo de uso de *find*:



Más sobre la operación find en https://docs.mongodb.com/manual/tutorial/query-documents/

Consultas: operación *find*

- Ejemplo de uso de *find*: los campos consultados pueden ir opcionalmente entre comillas simples o dobles. Las tres consultas que se muestran a continuación funcionan exactamente igual:

```
db.socios.find({numSocio: {$gt: 300} })
```

```
db.socios.find({'numSocio': {$gt: 300} })
```

```
db.socios.find({"numSocio": {$gt: 300} })
```

Más sobre la operación find en https://docs.mongodb.com/manual/tutorial/query-documents/

Consultas: operación *find*

- Para acceder a campos de documentos embebidos, se utiliza la notación con punto. En este caso, el uso de las comillas es necesario:

```
db.socios.find(
                {numSocio: {$gt: 300} },
                {nombre: 1, apellido1: 1, "contacto.teléfono": 1}
                        ).limit(3)
```

Más sobre la operación find en https://docs.mongodb.com/manual/tutorial/query-documents/

Consultas: condiciones

M.Zorrilla - D.García

- Existen diferentes tipos de operaciones:

De igualdad (*$eq*):

```
db.socios.find({numSocio: {$eq: 300} })
```

=

```
db.socios.find({numSocio: 300 })
```

Forma implícita de la operación $eq

Mayor que ($gt):

```
db.socios.find({numSocio: {$gt: 300} })
```

Mayor o igual que ($gte):

```
db.socios.find({numSocio: {$gte: 300} })
```

Menor que ($lt):

```
db.socios.find({numSocio: {$lt: 300} })
```

Menor o igual que ($lte):

```
db.socios.find({numSocio: {$lte: 300} })
```

Diferente a ($ne):

```
db.socios.find({numSocio: {$ne: 300} })
```

Pertenece ($in) o no ($nin) a un conjunto

```
db.socios.find({numSocio: {$in: [300, 301]} })
db.socios.find({numSocio: {$nin: [300, 301]} })
```

Consultas: condiciones

- Operadores lógicos:

  Ambas condiciones han de cumplirse($and):

```
db.socios.find({$and: [{nombre: "Juan"}, {apellido1: "Galindo" }  ]})
```

  Alguna de las condiciones ha de cumplirse ($or):

```
db.socios.find({$or: [{nombre: "Juan"}, {apellido1: "Galindo" }  ]})
```

  No debe de cumplir la condición ($not):

```
db.socios.find({numSocio: { $not: {$eq: 300} }})
```

  Que no se cumpla ninguna condición ($nor):

```
db.socios.find({$nor: [{nombre: "Juan"}, {apellido1: "Galindo" }  ]})
```

Consultas: condiciones

- Los operadores lógicos puede anidarse:

```
db.socios.find({
                    $or: [
                        {$and: [{nombre: “Juan”}, {apellido1: “Galindo” }   ]},
                        {numSocio: 300}
                          ]

})
```

Consultas: condiciones

- Operadores lógicos sobre campos:

  Existe el campo ($exists):

```
db.socios.find({ apellido2: { $exists: true } })
```

  El campo es de un tipo concreto ($type):

```
db.socios.find({ apellido2: { $type: "string" } })
```

- Operadores lógicos sobre arrays:

  El array contiene todos los valores ($all), algún valor del array cumple las condiciones ($elemMatch), el array es de un tamaño concreto ($size):

```
db.serie.find({ tags: { $all: ["historia", "aventura"] } })
db.serie.find({ puntuacion: { $elemMatch: {$gte: 2, $lt: 5 }}})
db.serie.find({ tags: { $size : 3}})
```

Consultas: condiciones

M.Zorrilla - D.García

- Proyecciones:
  - Valor 1 si queremos mostrar el campo.
  - Valor 0 si no queremos mostrarlo.
  - Por defecto, si no se declara nada en la proyección, se muestran todos los campos:
    - Al poner un campo a 1, se muestra solamente ese campo.
      - ¡OJO!: el campo _id se muestra siempre por defecto aunque haya otros campos en la proyección con valor 1.
      - Si no se quiere mostrar el campo _id, hay que ponerlo a 0 explícitamente en la proyección.
  - Ejemplos de proyecciones:

```
db.socios.find({numSocio: {$eq: 300} }, { nombre: 1, apellido1: 1 })
```
→ Muestra nombre, apellido1 y _id

```
db.socios.find({numSocio: {$eq: 300} }, { nombre: 1, apellido1: 1, _id: 0 })
```
→ Muestra nombre y apellido1

```
db.socios.find({numSocio: {$eq: 300} }, { numSocio:0 })
```
→ Muestra todos los campos excepto numSocio

Consultas: modificadores

- *Sort*: ordena los resultados según los campos dados. Un valor de 1 indica ordenamiento ascendente, y un valor de -1 ordenamiento descendente:

```
db.socios.find({numSocio: {$eq: 300} }, { nombre: 1, apellido1: 1 }).sort({nombre: 1})
```

- *Limit*: limita el número de colecciones retornadas. El 0 indica que no hay límite:

```
db.socios.find({nombre: {$eq: "Juan"} }, { nombre: 1, apellido1: 1 }).limit(3)
```

Consultas: agrupamiento

- *Count*: cuenta el número de veces que aparece en los documentos de una colección un valor en un campo.

- Ejemplo: suponiendo que en una colección llamada "series" tenemos un campo puntuación, y los siguientes documentos:

```
{puntuacion: 5}, {puntuacion: 3}, {puntuacion: 5}, {puntuacion: 5}, {puntuacion: 2},
```

La siguiente instrucción retornaría **3** (3 veces que se repite el valor 5 en el campo puntuación:

```
db.series.count({puntuacion: 5})
```

Consultas: agrupamiento

- *Distinct*: Retorna un array con los diferentes valores que tiene un campo concreto en los documentos de una colección.

- Ejemplo: suponiendo que en una colección llamada "series" tenemos un campo puntuación, y los siguientes documentos:

```
{puntuacion: 5}, {puntuacion: 3}, {puntuacion: 5}, {puntuacion: 5}, {puntuacion: 2},
```

La siguiente instrucción retornaría el array **[5, 3, 2]** (los tres valores que toma el campo puntuación):

```
db.series.distinct("puntuacion")
```

Consultas: agrupamiento

M.Zorrilla - D.García

- *Group*: agrupa los elementos que cumplan una condición concreta (*cond*) bajo los distintos valores de un campo (*key*).
- Ejemplo: suponiendo que en una colección llamada "series" tenemos un campo puntuación y un campo numSerie, y los siguientes documentos:

```
{numSerie: 1, puntuacion: 5}, {numSerie: 1, puntuacion: 3}, {numSerie: 2, puntuacion: 5},
{numSerie: 1, puntuacion: 5}, {numSerie: 2, puntuacion: 2}, {numSerie: 3, puntuacion: 5}
```

- Si se ejecutase la siguiente instrucción:

```
db.series.group({ key: { numSerie: 1 }, cond: { numSerie: { $lte: 2} }, reduce: function(cur, result) {
result.puntuacion += cur.puntuacion }, initial: {puntuacion: 0 } })
```

- El resultado sería la suma de las puntuaciones agrupando por numSerie, siempre que el numSerie sea igual o menor que 2:

```
{numSerie: 1, puntuacion: 13}, {numSerie: 2, puntuacion: 7}
```

Consultas: agrupamiento

- *Mapreduce*: Utiliza el método *map/reduce* para hacer consultas y agrupamientos. La instrucción *group* utiliza *map/reduce* implícitamente. Sintaxis:

```
db.<nombre_colección>.mapreduce(
        function() ….,
        function(key, values)…,
        {
                query: {…},
                out:…
        }
)
```

Función *map*

Función *reduce*

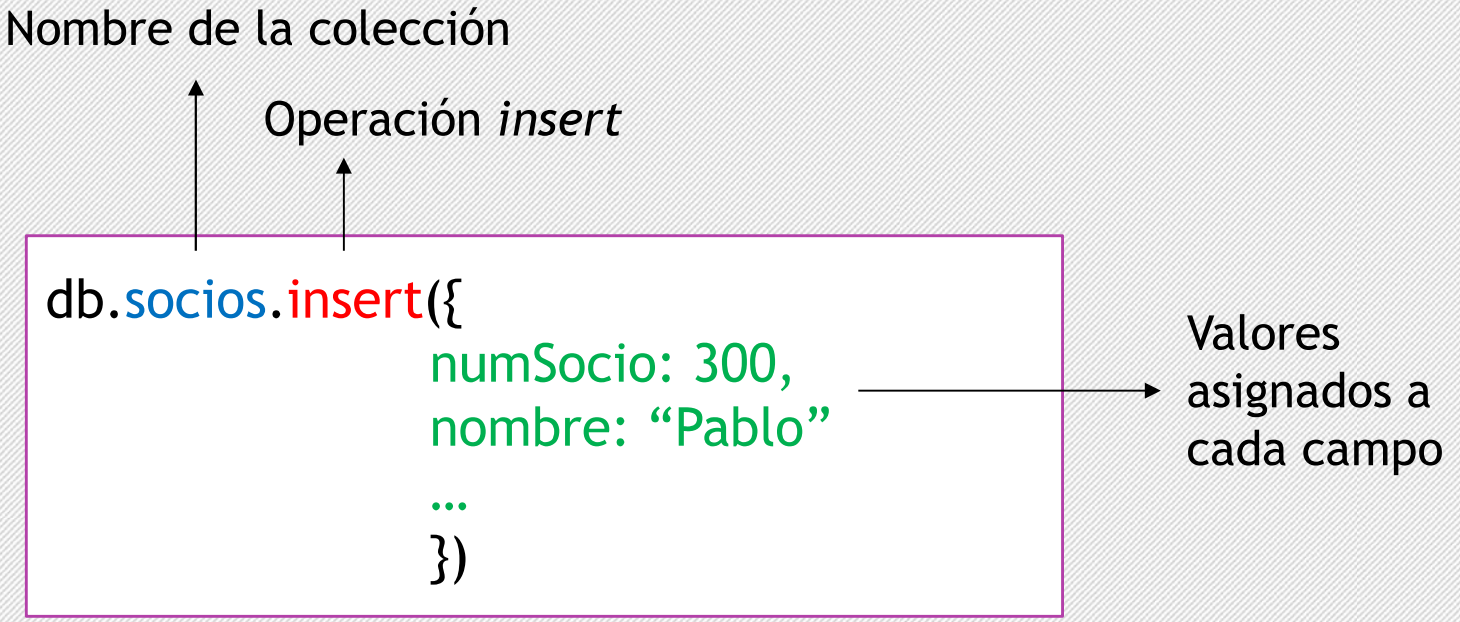
Condiciones de la consulta

Salida

Para más información sobre las funciones de agrupamiento, se recomienda visitar el siguiente enlace: https://docs.mongodb.com/manual/aggregation/

Insertado: operación *insert*

- Ejemplo de uso de *insert*:

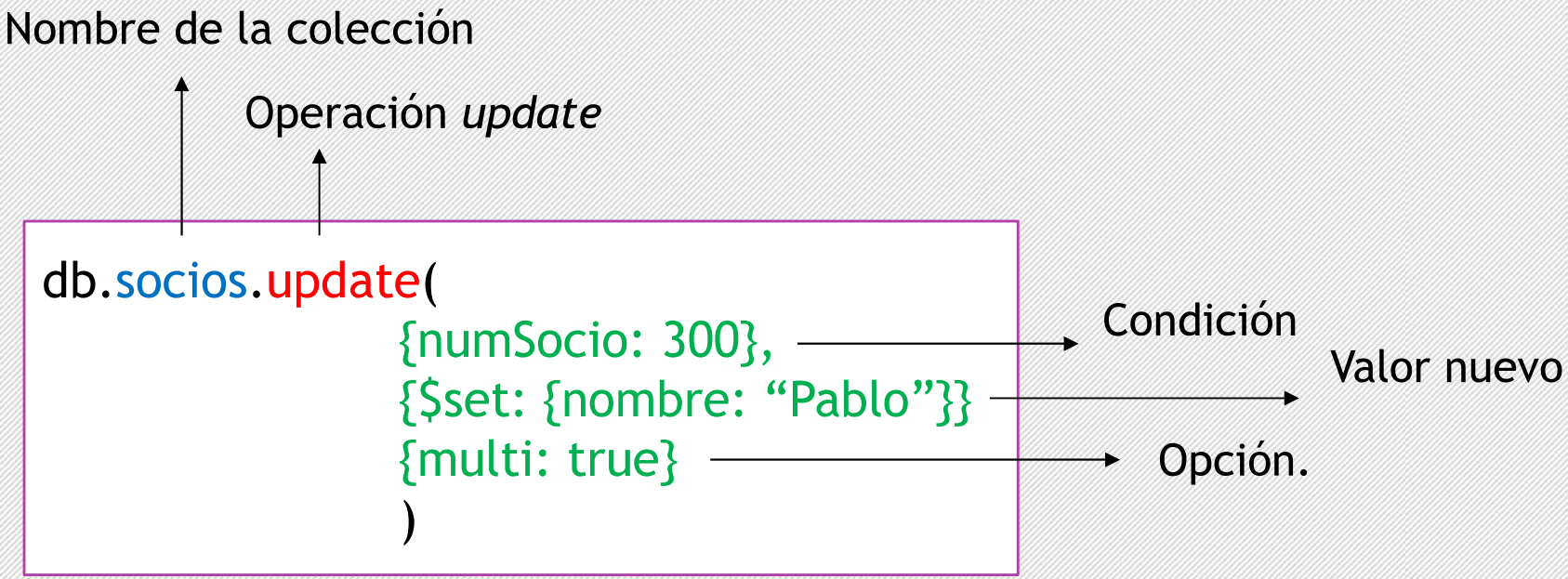


- Si no se inserta un valor para el campo _id, se genera automáticamente:
  - _id único, no hay peligro de *UPSERTS*: si se inserta un valor existente en la colección, devuelve error.

Más sobre la operación insert en https://docs.mongodb.com/manual/tutorial/insert-documents/

Actualizado: operación *update*

- Ejemplo de uso de *update*:



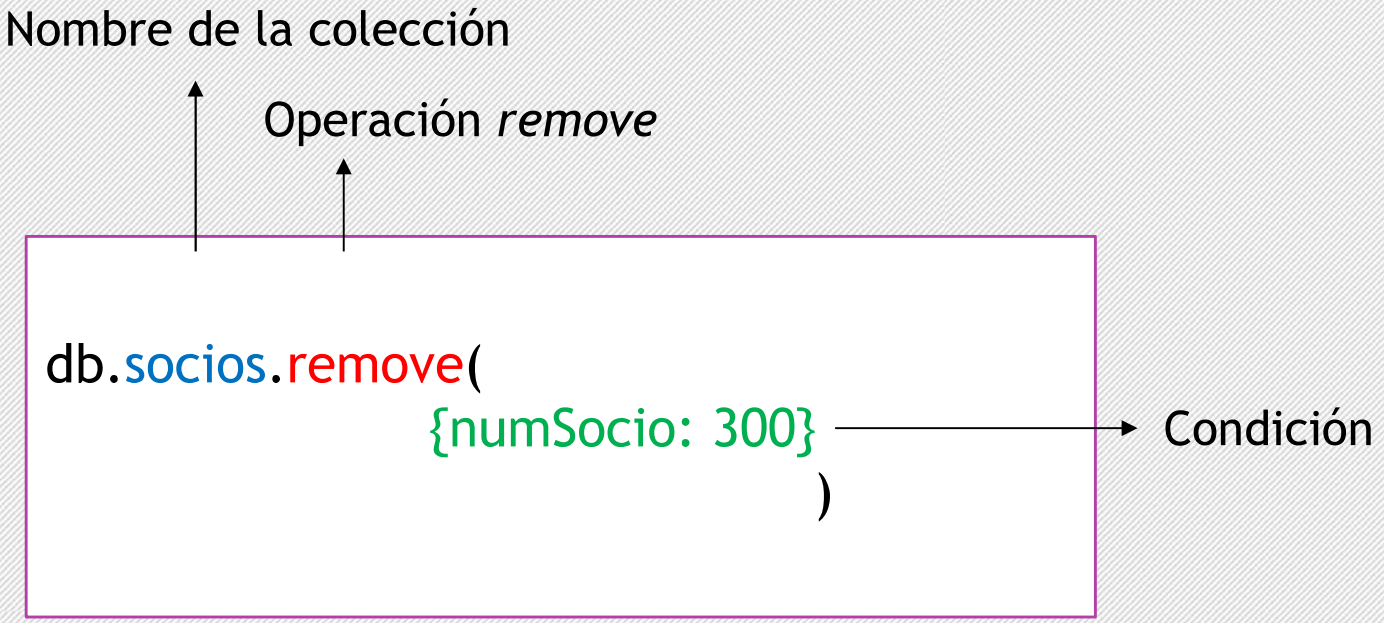
- La operación *update* admite opciones: multi:true indica que se pueden modificar varios documentos simultáneamente.
- UPSERT: si se establece *{upsert: true}* y no hay documentos que cumplan las condiciones, se crea uno nuevo con los valores proporcionados.

Más sobre la operación update en https://docs.mongodb.com/manual/tutorial/update-documents/

Actualizado: operación *remove*

M.Zorrilla - D.García

- Ejemplo de uso de *remove*:



Más sobre la operación remove en https://docs.mongodb.com/manual/tutorial/remove-documents/