

The slide features two thick black L-shaped brackets. One is on the left, with a horizontal bar at the top and a vertical bar extending downwards. The other is on the right, with a vertical bar at the top and a horizontal bar extending to the left. They frame the central text.

BASES DE DATOS NOSQL

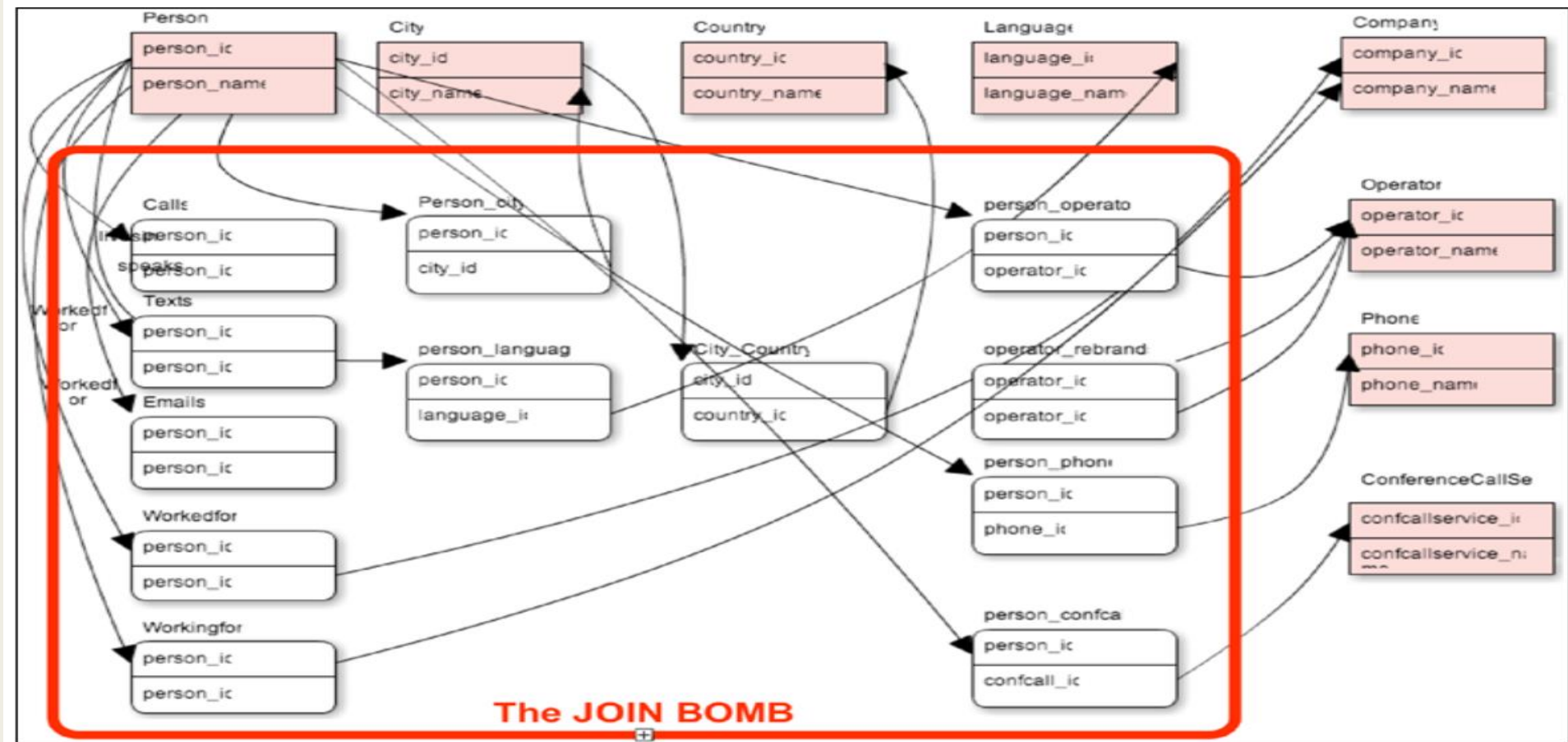
Mongo DB

NoSQL

- NoSQL viene del término “Not Only SQL”
- NoSQL. Son sistemas manejadores de bases de datos que no utilizan el esquema entidad-relación para almacenar la información
 - *Surgen como resultado del crecimiento exponencial de los datos al migrar servicios a la red (aplicaciones web)*
 - *Las bases de datos relacionales no estaban preparadas para almacenar los volúmenes de información de las redes sociales o de sitios multimedia como youtube*
- Surgen para resolver el problema de los JOIN en bases de datos relacionales
 - *JOIN es la operación para unir diferentes tablas (utilizando su llave primaria)*

NoSQL

- El problema de los JOIN en bases de datos relacionales



Ventajas de NoSQL

- Muy veloces al manipular grandes cantidades de información (gran velocidad para hacer consultas)
- Escalabilidad horizontal. Se pueden agregar más servidores para incrementar los recursos computacionales de la base de datos
 - *Es más barato agregar servidores que comprar más memoria o disco duro para incrementar la capacidad de un solo servidor*
- Se pueden ejecutar en máquinas con pocos recursos computacionales
- No hay JOIN

Desventajas de NoSQL

- No hay transacciones. Una transacción es el conjunto de operaciones necesarias para realizar una acción sobre la base de datos. Si alguna de las operaciones falla, entonces la todas las operaciones realizadas regresan a su estado original
- No son tan veloces para escribir grandes cantidades de información en la base de datos
- Puede haber redundancia de la información
- Puede haber inconsistencia de la información

Diferencias con Bases de Datos SQL

- No se utiliza SQL como lenguaje de consulta.
 - *SQL es un estándar para acceder a bases de datos relacionales*
 - *En NoSQL no hay un lenguaje estándar de consulta*
- Utilizan estructuras dinámicas para almacenar la información. Se pueden agregar campos sin necesidad de realizar operaciones costosas
 - *Las tablas en bases de datos relacionales son fijas (es fijo el número de campos o columnas) y es costoso hacer modificaciones*

NoSQL

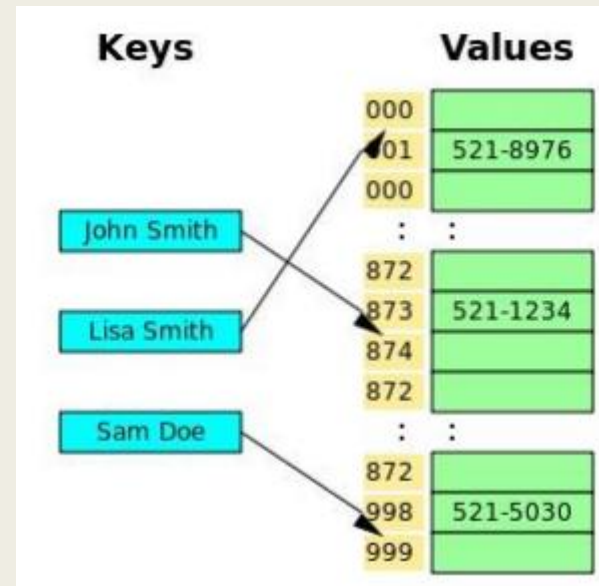
Hay 5 modelos de bases de datos NoSQL

- Key - Value stores. Se almacenan pares de valores <nombre:"Pedro">
- Column stores. La información se almacena por columnas, en lugar de por filas
- Graph stores. Utiliza un grafo que modela relaciones entre los datos
- Document stores. Se almacena la información en documentos.
- Multi-model. Bases de datos que combinan los modelos anteriores

Key-Value stores

- Representan la información de forma muy simple (pares de valores)
 - *<identificador, valor>*

- Su simplicidad los hace muy rápidos



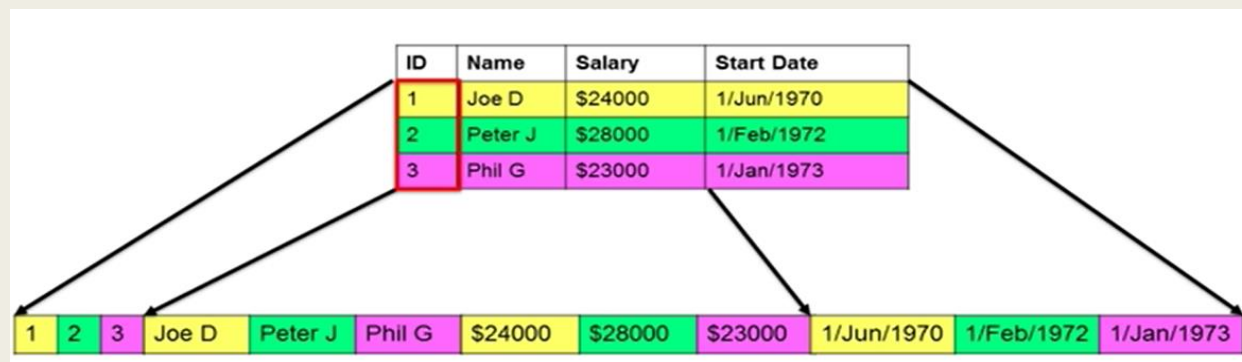
- Algunos ejemplos de sistemas manejadores de bases de datos son:
 - *Amazon Dynamo*
 - *MemcacheDB*

Bases de datos en columnas

- La información se almacena por columnas en lugar de por filas
- Es más fácil realizar operaciones de agregación (resumir la información)



← Almacenamiento relacional

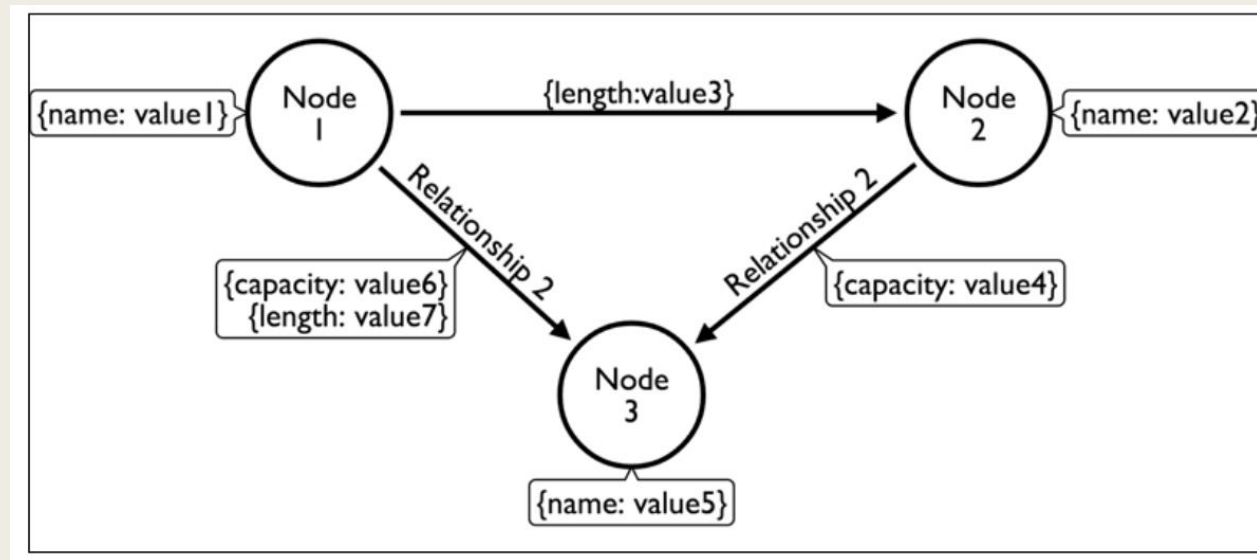


← Almacenamiento por columnas

- Algunos ejemplos de bases de datos en columnas son BigTable o Apache Cassandra

Bases de datos orientadas a grafos

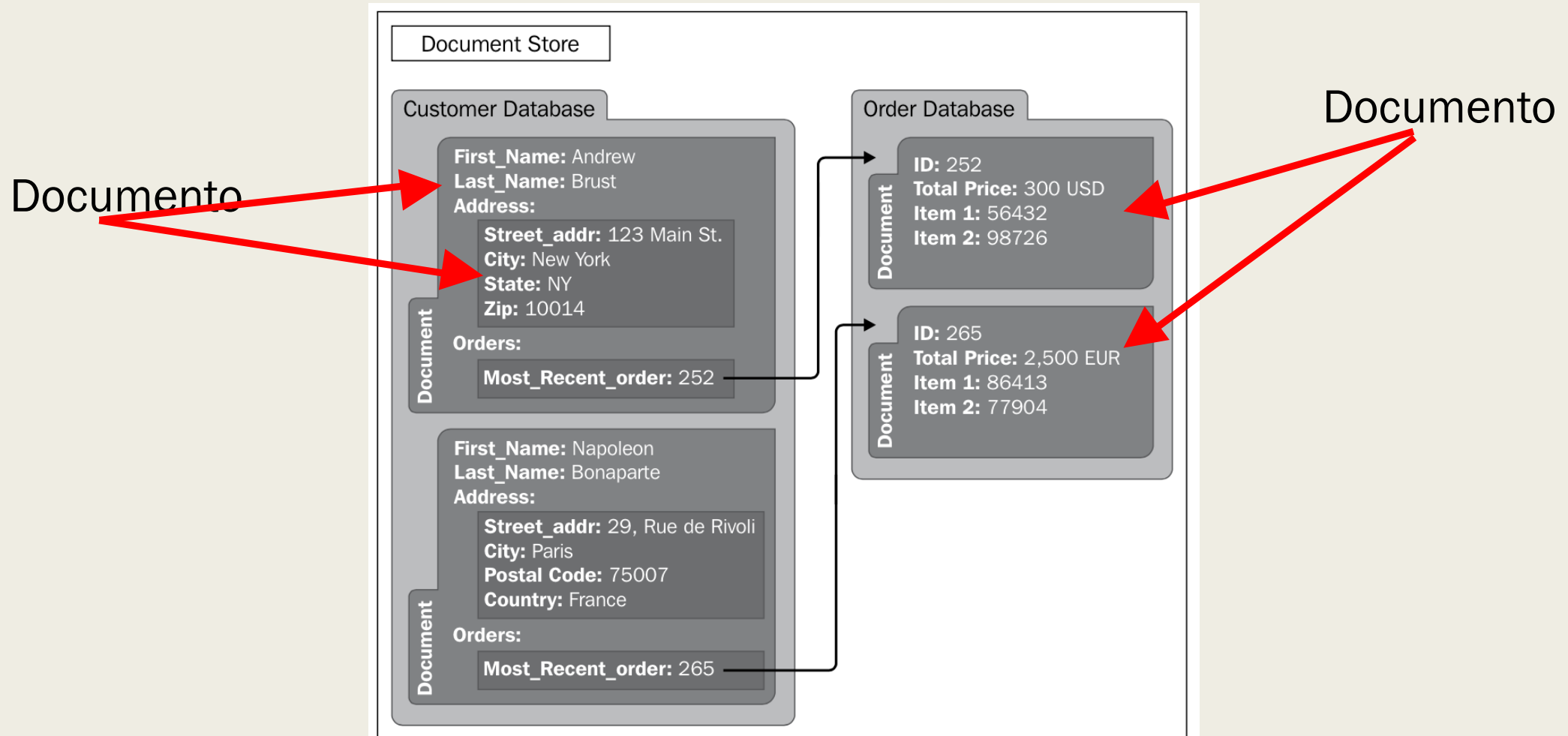
- La información se almacena utilizando un grafo
 - *Nodos.* Cada nodo es una abstracción del mundo real. Ej. Nodo ciudad. Nodo persona.
 - *Relaciones.* Representan relaciones entre objetos del mundo real. Ej. Matrimonio



- *Son muy efectivas para almacenar información de redes sociales*

Bases de datos orientadas a documentos

- Un documento es un conjunto de pares de valores



Bases de datos orientadas a documentos

- Los documentos se almacenan en formatos de texto simple
 - *XML* `<id>1</id><nombre>juan</nombre>`
 - *JSON* `{id:1, nombre:"juan"}`
- Un documento puede estar compuesto de otros documentos
 - *Documentos embebidos*
- Los documentos no tienen una estructura fija
 - *Cada documento puede tener diferentes campos*
- Ejemplos de bases de datos orientadas a documentos
 - *MongoDB*
 - *CouchDB*

MongoDB

- MongoDB es una base de datos orientada a Documentos (Document Store)
- MongoDB estructura la información de la siguiente manera:
 - *Base de datos*
 - *Colecciones*
 - *Documentos*
 - *Campos*

MongoDB

- Base de datos
 - *Es un contenedor físico para Colecciones.*
- Colección
 - *Es un grupo de Documentos. Es equivalente a una tabla en RDBMS (base de datos relaciona)*
 - *No tienen una estructura como en RDBMS (**id** int, **nombre** varchar, **dirección** varchar)*
- Documento
 - *Se puede pensar en un documento como en un registro en RDBMS (fila de datos)*
 - *Es un conjunto de valores KEY-VALUE → {nombre:"Juan"}*
 - *No tienen una estructura fija (cada documento puede tener diferentes campos (KEY-VALUE))*

MongoDB Comparativa

- Comparativa entre una RDBMS y un Document Store

| RDBMS | Document Store |
|------------------------------|---------------------------|
| Base de datos | Base de datos |
| Tabla | Collección |
| Registro (fila de datos) | Documento |
| Campo (columna de datos) | Campo |
| Table Join (unión de tablas) | Documentos embebidos |
| Llave primaria | Llave primaria automática |

MongoDB Representación de documentos

- Los documentos se escriben utilizando JSON (JavaScript Object Notation)
 - *JSON es un formato para almacenar e intercambiar información*
 - La información que se transmite a través de la red es TEXTO por lo tanto es necesaria una notación que permita estructurar y representar dicho texto
 - Ejemplo de un JSON

```
{ name: "John", age: 31, city: "New York" }
```
- De manera interna, en MongoDB, los documentos JSON se traducen a BSON (Binary JavaScript Object Notation)
 - *BSON es una extensión de JSON*
 - Permite almacenar arreglos de valores y estructuras complejas

MongoDB Documentos


■ Ejemplo de un Documento

```
{
  titulo: "Presentación MongoDB",
  descripcion: "MongoDB es un DBMS orientado a documentos",
  presenta: "Ariel García",
  keywords: ["diplomado", "Información", "Base de datos"],
  alumnos: 24,
  alumnos: [
    {
      nombre: "Enrique",
      matricula: 12345
    },
    {
      nombre: "Federico",
      matricula: 54321
    }
  ]
}
```

Arreglos de
valores



Documentos
embebidos



MongoDB. Comandos

- Crear una base de datos

- use databaseName*

- *El comando USE se utiliza tanto para crear una base de datos como para seleccionarla (utilizarla)*

- Revisar la base de datos actualmente seleccionada

- db*

- Mostrar las bases de datos en el sistema

- show dbs*

- *Las bases de datos se muestran hasta que haya al menos un documento o colección dentro de la misma*

- Eliminar una base de datos

- db.dropDatabase()*

MongoDB. Comandos

- Crear colecciones

- db.createCollection("collectionName")***

- *Ej. db.createCollection("users")*

- Mostrar las colecciones de la base de datos

- show collections***

- Eliminar una colección

- db.collection.drop()***

- *Ej. db.users.drop()*

- Insertar documentos a las colecciones

- db.collection.insert(document)***

- *Ej. db.users.insert({nombre:"Pedro", apaterno:"Ramírez", amaterno:"García"})*

MongoDB Comandos

- Insertar más de un documento en la misma consulta
 - *Las inserciones múltiples se hacen utilizando arreglos de documentos*
 - *Ej:*
`db.users.insert([
 {nombre:"Roy", apaterno:"Silva", amaterno:"Rosas", edad:30},
 {nombre:"Gerry", apaterno:"Moreno", amaterno:"González", edad:38}
])`

MongoDB Comandos

- Búsqueda de información
 - *Para buscar o mostrar la información dentro de las colecciones se utiliza el comando **find***
db.collection.find()
- El resultado de una búsqueda es un documento (conjunto de documentos) con formato JSON
 - *El JSON se muestra en una sola línea por lo que es difícil de leer*
- Para facilitar la lectura de la información resultante se utiliza el comando **pretty**
 - *Ej. **db.users.find().pretty()***
- Para mostrar un único documento se utiliza el comando **findOne**
 - *Ej. **Db.users.findOne().pretty()***

MongoDB Filtrar Búsquedas

- Algunas formas de filtrar la información en mongoDB

| Filtro | Instrucción | Ejemplo |
|-------------------|------------------------|--|
| Menor qué | {key : {\$lt: value}} | db.users.find({edad:{\$lt:38}}) |
| Menor o igual qué | {key : {\$lte: value}} | db.users.find({edad:{\$lte:30}}) |
| Mayor qué | {key : {\$gt: value}} | db.users.find({edad:{\$gt:30}}) |
| Mayor o igual qué | {key : {\$gte: value}} | db.users.find({edad:{\$gte:38}}) |
| Igual | {key : value} | db.users.find({nombre:"Pedro"}) |
| Diferente | {key : {\$ne: value}} | db.users.find({nombre:{\$ne:"Pedro"}}) |

MongoDB Filtrar Búsquedas

- Filtrar por más de un criterio (AND)
 - *La coma nos permite aplicar más de un filtro de búsqueda (más de un campo)*
 - *Ej. Buscar a los usuarios que tienen el mismo apellido paterno y apellido materno (hermanos)*

```
db.users.find({apaterno:"Silva", amaterno:"Rosas"})
```

MongoDB Filtrar Búsquedas

- Filtrar con el criterio OR

- *Es necesario utilizar la palabra reservada \$or*
- *Ej. Buscar a los usuarios cuyo apellido paterno es o Silva o Ramírez*

```
db.users.find({  
    $or : [ {apaterno:"Silva"},  
            {apaterno:"Moreno"}  
    ]  
})
```


MongoDB Filtrar Búsquedas

- Combinar AND y OR como criterios de búsqueda
 - *Ej. Buscar todos aquellos Usuarios que tienen menos de 38 años y cuyo apellido paterno o es Silva o es Ramírez*

```
db.users.find({edad: {$lt:38},  
              $or:[  
                {apaterno:"Silva"}, {apaterno:"Ramírez"}  
              ]  
            })
```

MongoDb Búsquedas con proyección

Proyección. Las proyecciones en MongoDB significan mostrar sólo aquellos campos en los que se está interesado al realizar una búsqueda

- A el comando find se le agrega un segundo parámetro en el que se indica qué con un 1 los campos se quiere mostrar y con un 0 los que no
- Ej. Mostrar todos los documentos, pero sólo el nombre y la edad de los usuarios

```
db.users.find({},  
                {nombre:1, edad:1, _id:0})
```

- Ej. Mostrar sólo la edad del usuario Juanito

```
db.users.find({nombre:"Juanito"},  
              {_id:0, edad:1})
```

MongoDB Limitar el número de documentos devueltos en una búsqueda

- Es muy importante limitar el resultado de una búsqueda para no saturar la base de datos cuando el número de documentos resultantes sea muy grande
- Hay dos formas de limitar el número de documentos devueltos por una búsqueda
 - ***limit.*** La función *limit* recibe como parámetro el número de documentos a mostrar
Ej. Mostrar sólo 3 documentos de la colección de usuarios
`db.users.find().limit(3)`
 - ***Skip.*** Recibe como parámetro el número de documentos a saltarse.
Suponiendo que la colección tiene 10 documentos y si aplicamos la función `skip(1)`, entonces sólo se mostrarán 9 documentos
`db.users.find().skip(1)`

MongoDB Actualizar Documentos

- Hay dos formas de actualizar un documento
 - **update.** *Actualiza los valores de un documento*
 - **save.** *Remplaza el documento existente con el nuevo documento*

- Update

db.collection.update(criterioBusqueda, nuevosDatos)

- Donde el criterio de búsqueda puede ser cualquiera de los antes vistos
- Ej. Actualizar el nombre del usuario Ivonne dado que está mal escrito (cambiar por Ivonn)

```
db.users.update(  
    {nombre:"Ivonne"},  
    {$set:{nombre:"Ivonn"}},  
    {multi:true}  
)
```

- La opción **multi:true** indica que va a reemplazar a todos los nombres que sean igual a Ivonne

MongoDB Actualizar Documentos

■ Save

- *El comando **save** puede realizar las siguientes acciones:*

- Reemplazar un documento completo. Se requiere identificar el documento a reemplazar con su `_id`)

Ej. Reemplazar el documento de Pedro Ramírez García

```
db.users.save({ _id:ObjectId("5aa186ed5bb651c5c851b6c2"),  
              nombre:"Juanito", apaterno:"Gómez", amaterno:"Gamboa", edad:50})
```

- Insertar un documento. Si no se identifica ningún documento a reemplazar, entonces hace una inserción

Ej. Tratar de modificar a Gustavo Lara, pero como no se identifica con su `_id`, entonces se inserta

```
db.users.save({name:"Gustavo", apaterno:"Lara", edad:26})
```

MongoDB Actualizar Documentos

- Actualizar el nombre de un campo

- *Para actualizar un campo se utiliza el comando **\$rename***
- *Ej. En el ejemplo de la diapositiva anterior se insertó al usuario Gustavo, pero el nombre del campo se puso como “name” cuando nombre correcto del campo es “nombre”. Actualiza el nombre del campo*

```
db.users.updateMany({},  
                    {$rename:{name:"nombre"}} )
```

MongoDB Eliminar documentos

- El comando para eliminar un documento es **remove**

remove(criterioBusqueda, justOne)

- *Donde el criterio de búsqueda indica que documento se va a eliminar y **justOne** permite especificar si se elimina sólo un documento o todos los documentos que cumplan el criterio de búsqueda*
- *Ej. Eliminar a todos los Silva Rosas*

db.users.remove({apaterno:"Silva",amaterno:"Rosas"}, 0)

1 → Sólo un documento

0 → Todos los documentos que hacen match con la búsqueda

- *Para eliminar todos los documentos de una colección se utiliza **remove** sin criterio de búsqueda*

db.users.remove()

MongoDB Ordenar Documentos

- Los documentos se pueden ordenar de mayor a menor o viceversa
 - *1 indica orden ascendente*
 - *-1 indica orden descendente*
- La función para ordenar es **sort**
 - *La función recibe como parámetro el o los campos por los que hay que ordenar los documentos*
 - *Ej. Muestra los usuarios de los más jóvenes a los más grandes*
db.users.find().sort({edad:1})
 - *Ej. Muestra quién es el usuario más grande*
db.users.find().sort({edad:-1}).limit(1)

MongoDB Operaciones de Agregación

Agregación. Significa resumir o agrupar la información mediante alguna operación

- La agregación permite agrupar valores de diferentes documentos y resume la información
- La agregación en MongoDB es mediante el método **aggregate**

Ej. `db.users.aggregate([`

`{$group:`

`{ _id:"$nombre",
 hay:{$sum : 1}
 }
 }`

El resultado es

`{ "_id" : "Ivonne", "hay" : 1 }
{ "_id" : "Gustavo", "hay" : 1 }
{ "_id" : "Gerardo", "hay" : 2 }
{ "_id" : "Gerry", "hay" : 1 }`

`])`

Donde: `$nombre` indica que se deben agrupar por el campo nombre, *hay sólo* es un título para el resultado y `$sum` indica que cada se van a sumar los resultados

MongoDB Operaciones de Agregación

- Ejemplo: agrupa la información de la colección de usuarios y cuenta cuántos hombres y cuántas mujeres hay

```
db.users.aggregate([
```

```
  {$group:
```

```
    {_id:"$genero",
```

```
    hay:{$sum:1}
```

```
  ]
```

```
])
```

resultado



```
{ "_id" : "F", "hay" : 1 }
```

```
{ "_id" : "M", "hay" : 4 }
```

MongoDB Operaciones de agregación

Expresiones de agregación. Son aquellas operaciones que nos permiten resumir la información

- Hay diferentes funciones para resumir la información. A continuación se muestran las más utilizadas

| Expresión de agregación | Descripción |
|-------------------------|----------------------------|
| \$sum | Suma los valores obtenidos |
| \$avg | Obtiene el promedio |
| \$min | Devuelve el mínimo |
| \$max | Devuelve el máximo |
| \$first | Obtiene el primer elemento |
| \$last | Obtiene el último elemento |

MongoDB Operaciones de Agregación

Ejemplos:

- \$sum

Suma todos los salarios (\$500)

```
db.users.aggregate([{$group: {_id: "nombre", totalSalarios: {$sum: "$salario"}}}])
```

- \$avg

Calcula la edad promedio de hombres y mujeres (32, 31)

```
db.users.aggregate([{$group: {_id: "$genero", edadPromedio: {$avg: "$edad"}}}])
```

- \$min

– Edades del hombre y la mujer más jóvenes (31,26)

```
db.users.aggregate([{$group: {_id: "$genero", masJoven: {$min: "$edad"}}}])
```