# Relaciones

- Se pueden incrustar, pero tambien se pueden pasar por referencia

Clientes

```
{
    "nombre": "Juan",
    "librosFav": ["id1", "id2"]
}
```

---

Libros

```
{
    "_id":"id1",
    "nombre": ["Lazarillo"]
}
```

---

Así si necesito aplicar cambios sería en un solo documento, no como pasaría con documentos incrustados

# Relación uno a uno (incrustado)

Ejemplo del hospital

- Paciente 1----> historial médico 1
- Paciente 2----> historial médico 2

Aqui no tienen relación ninguna paciente 1 y paciente 2

Puedo relacionar el historial con el paciente mediante un id

```
> db.pacientes.insertOne( {nombre: "Pedro", historial:"historial1"} )

> db.historial.insertOne( {_id:"histrorial1", enfermedad:"COVID"} )
```

Si ahora necesito consultar los datos de Pedro y el historial puedo extraer con findOne

```
> db.pacientes.findOne( {nombre:"Pedro} ).historial
```

> NOTA: solo se puede extraer con findOne

Lo puedo guardar en una variable

```
> var hist = db.pacientes.findOne( {nombre:"Pedro"} ).historial
> var   //OUTPUT: historial1
```

Para buscarlo

```
> db.historial.findOne( {_id: his} )
```

Podría hacer así

```
> db.historial.findOne( {_id: his} ).enfermedades
```

En una relación de 1 a 1 no tiene sentido hacer todos estos pasos.

> Es más práctico usar datos incrustados para este tipo de relación 1 a 1 y la notación de punto para extraerlo con findOne

```
> db.pacientes.findOne({nombre:"Pedro"}).historial.enfermedades
```

# Relación 1 a 1 - referencia

Creo la db nueva con use

```
> use coches
```

Y la coleccion personas

```
> db.personas.insertOne({nombre:'Marcos', sueldo:1000, coche:{modelo:
"Toyota", precio:12000}})
```

En este caso de relación 1 a 1 si puede interesar hacer uso de colecciones, para hacer analítica de cual es la clase de coche promedio, por ejemplo

Uso la estrategia anterior que aunque no tenia sentido con los pacientes, aquí si

# Relación 1 a muchos

Por ejemplo, en una aplicación de preguntas y respuestas

- Tienes una pregunta y hay varias respuestas posibles

```
  > use preguntasyrespuestas
  > db.pregunta.insertOne({nombre: "Juan", pregunta:"Cómo funciona esto",
respuestas:["preg1resp1","preg1resp2"]})
```

---

```
> db.respuesta.insertMany( [ {_id:"preg1resp1", texto:"Funciona así"}.
_id:"preg1resp2", texto:"Gracias" ] )
```

---

# Relación muchos a muchos

Un ejemplo muy claro es una tienda con una serie de clientes que compran varios productos

- Cliente 1: producto2, producto4, producto5
- Cliente 2: producto1, producto2, producto5

```
> db.productos.insertOne( {titulo:"un libro", precio: 9.99} )
> db.clientes.insertOne( {nombre:"Juan", edad: 25} )
```

- En pedidos uso el id que se generó con productos.insertOne "un libro"
- Y el id_cliente el id que se generó con clientes.insertOne "Juan"

```
> db.pedidos.insertOne( {idProducto:ObjectId(12233445453313),
idCliente=ObjectId(97268729362736)} )
```

- Si hago db.pedidos.find() me devuelve estos dos id's
- Esta es la forma de hacer SQL

> EN MONGO LO HARÍA DE OTRA FORMA

Pongo arreglo vacío pq solo hay uno, uso el atomic operator y el id del producto cuando se generó

```
    > db.clientes.updateOne({}, {$set:{pedidos:[
{idProducto:ObjectId(12233445453313), cantidad: 3} ] } } )
```

Parece una manera de incrustar otro documento, pero le he pasado el id de producto que sigue estando en otra colección

- La diferencia está en que así no hay duplicación de datos y la escalabilidad a la hora de manipularlos

# Muchos a muchos - ref

```
    > use libros
    > db.libro.insertOne({nombre: "un libro", autores:[ {nombre: "Juan"}, {nombre:
"Sebastián"} ] } )

    >db.autores.insertMany([ {nombre: "Juan", direccion: "Calle A"}, {nombre:
"Sebastián", direccion: "Calle B"}])
```

Ahora tengo id's de los autores en una colección específica, puedo añadírselos al libro Como libro solo tengo uno puedo ponerle por filtro el objeto vacío

```
    > db.libro.updateOne({}, {$set:{autores:[ObjectId:("687236872136"),
ObjectId("78623872632836")]}})

    > db.libro.find().pretty()
```

Esto me muestra un id del libro, el nombre y los autores con sus respectivos id's