Unir colecciones en MongoDB – De SQL a NoSQL

Objetivo

Aprender a realizar uniones entre colecciones en MongoDB utilizando `$lookup`, entendiendo su lógica, estructura y diferencias con los JOIN de SQL. Este apunte no solo explica cómo se usa `$lookup`, sino por qué se usa así, comparando con el modelo relacional.

¿Qué es una unión de datos?

Cuando tenemos información distribuida en **varias tablas** (o colecciones), muchas veces queremos combinarla para obtener una visión más completa. A eso se lo llama hacer un **join** (unión).

En SQL

Usamos `JOIN` para relacionar dos tablas según alguna columna en común.

```
SELECT empleados.nombre, departamentos.nombre
FROM empleados
JOIN departamentos
ON empleados.id_departamento = departamentos.id
```

En MongoDB

Usamos el operador `$lookup` dentro del método `.aggregate()` para unir colecciones.

```
db.empleados.aggregate([{
    $lookup: {
      from: "departamentos",
      localField: "id_departamento",
      foreignField: "id",
      as: "info_departamento"
    }
}])
```

Diferencia conceptual: tablas relacionadas vs documentos embebidos

- En SQL las relaciones se basan en claves foráneas (`FOREIGN KEY`) y los datos están separados.
- En MongoDB muchas veces **se recomienda** guardar los datos **dentro del mismo documento** si pertenecen al mismo contexto.
- Pero cuando los datos **no son embebibles**, usamos `$lookup`.

Estructura de un $lookup

```
{
  $lookup: {
    from: "departamentos",
    localField: "id_departamento",
    foreignField: "id",
    as: "departamento"
  }
}
```

`from: "departamentos"`

Este campo indica **el nombre de la colección que queremos unir**.
Equivale a decir en SQL: `JOIN departamentos ...`

En este ejemplo, queremos unir cada documento de `empleados` con la colección `departamentos`.

`localField: "id_departamento"`

Este campo representa **el nombre del campo en la colección actual** (en este caso, `empleados`) que **actúa como clave de relación**.

Es como en SQL decir: `ON empleados.id_departamento = ...`

Es decir, cada empleado tiene un valor en `id_departamento`, y queremos usarlo para encontrar el departamento correspondiente.

`foreignField: "id"`

Este campo representa **el nombre del campo en la otra colección (`departamentos`) que se va a comparar** con el `localField`.

Es el campo que, en la colección unida, debe coincidir con el campo local.

Traducido a SQL: `... = departamentos.id`

`as: "departamento"`

Este campo indica **el nombre del campo nuevo donde se va a guardar el resultado de la unión**.

MongoDB va a agregar un nuevo campo en cada documento llamado `"departamento"`, que va a contener un array con los datos coincidentes de `departamentos`.

Es como decir: "Guardá el resultado de este join bajo este nombre".

Podés llamarlo como quieras: `"info_departamento"`, `"detalle"`, `"union"`, etc.
Pero conviene usar un nombre descriptivo.

¿Y si no hay coincidencias?

Si no hay ningún documento en la colección `departamentos` que coincida con `id_departamento`, entonces el campo `"departamento"` será un **array vacío** (`[]`).
No da error: simplemente indica que no hubo relación.

Ejemplo concreto

Colección: empleados

```
{
  "nombre": "Ana",
  "apellido": "Pérez",
  "id_departamento": 2
}
```

Colección: departamentos

```
{
  "id": 2,
  "nombre": "Recursos Humanos"
}
```

Consulta con `$lookup`

```
db.empleados.aggregate([
  {
    $lookup: {
      from: "departamentos",
      localField: "id_departamento",
      foreignField: "id",
      as: "departamento"
    }
  }
])
```

Resultado

```
{
  "nombre": "Ana",
  "apellido": "Pérez",
  "id_departamento": 2,
  "departamento": [
    {
      "id": 2,
      "nombre": "Recursos Humanos"
    }
  ]
}
```

Atención: el resultado de `$lookup` siempre es un **array** (aunque solo haya una coincidencia).

Si estás seguro de que la relación es 1 a 1, podés usar `$unwind` para convertirlo en un solo objeto (ver más abajo).

Equivalencias SQL vs MongoDB con $lookup

Descripción	SQL	MongoDB
Traer empleados con nombre del departamento	`JOIN departamentos ON empleados.id_dep = departamentos.id`	`$lookup` con `from`, `localField`, `foreignField`, `as`
Campo relacionado se llama igual	`JOIN tabla ON tabla.id = tabla.id`	El campo puede tener distinto nombre; solo se necesita indicar cuál es cuál
Resultado plano (sin array)	Devuelve columnas combinadas	Usa `$unwind` para aplanar el array resultante

¿Qué es $unwind?

`$unwind` sirve para desarmar un array de resultados y trabajar con un solo documento por coincidencia.

```
{
  $unwind: "$departamento"
}
```

Este paso convierte:

```
"departamento": [
  { "id": 2, "nombre": "Recursos Humanos" }
]
```

En:

```
"departamento": { "id": 2, "nombre": "Recursos Humanos" }
```

Filtrar documentos después de la union

Para filtrar los documentos podés agregar un `$match` después para filtrar resultados.
Ejemplo:

```
db.empleados.aggregate([
  { $lookup: { ... } },
  { $unwind: "$departamento" },
  { $match: { "departamento.nombre": "Recursos Humanos" } }
])
```

¿Por qué .aggregate() y no .find()?

Porque `.find()` solo sirve para consultar una **única colección**.
Si queremos unir datos de **dos colecciones**, necesitamos usar `.aggregate()`, que permite operaciones más complejas como `$lookup`, `$unwind`, `$match`, etc.

¿Cuándo usar $lookup y cuándo embebido?

Situación	¿Embebido o `$lookup`?
Datos que siempre se consultan juntos	Embebido
Datos que cambian frecuentemente y están relacionados con muchos documentos	`$lookup` (evita duplicación)
Relación muchos a uno o uno a uno	`$lookup` con `$unwind`
Relación muchos a muchos	`$lookup` con arrays

Conclusiones clave

- `$lookup` es el equivalente al `JOIN` de SQL.
- Requiere trabajar con `.aggregate()` y se configura indicando los campos relacionados.
- Siempre devuelve un array con los documentos relacionados.
- Podés usar `$unwind` si querés convertir el array en un solo objeto.
- No abuses del `$lookup`: si podés embebir los datos sin duplicar demasiado, es más eficiente.