ScopeTreasure Courses

MONGODB COURSES · LESSON MG.8 · CODE SCOPE SERIES

Lookup + Relations Deep — Do Collections Ko Jorna

Join two collections the MongoDB way: embedding vs referencing, the $lookup stage, and the empty array that tells the truth.

Introduction

MG.6 left one thread hanging: fees live in their own collection, carrying `studentId` — so how do you ever see a student *with* his fee? MongoDB has two answers, and a real developer chooses between them per relationship. **Embedding**: keep the related data *inside* the document. **Referencing**: keep it in its own collection, pointing with an id — and use the aggregation stage `$lookup` when a question must cross the border. This lesson teaches both, with September's fees from DB.5/MG.6 as the test data.

(Roman Urdu: Andar rakhna = embedding; pata rakhna = referencing; aur $lookup = parosi ke ghar ja kar maloomat lena.)

Key points

- **Embed what travels together:** data that is read together, belongs to one parent, and stays small — a student's subjects, a product's sizes, an order's items. One fetch brings the whole story; the data dies with its parent, which is usually right.
- **Reference what is shared, big, or fast-growing:** data many parents share (a category, a teacher), data that grows without limit (comments, messages), or data often read alone (a fee record the office queries by itself). Store the `_id` of the other side — a `studentId` — and cross only when needed.
- **$lookup is MongoDB's JOIN:** an aggregation stage: `{ $lookup: { from: 'fees', localField: '_id', foreignField: 'studentId', as: 'fees' } }` — *from* the other collection, match my `_id` against its `studentId`, and attach the matches as a new array field named by `as`.
- **$lookup behaves like a LEFT JOIN:** every left document survives; matches arrive as an array — **and no match arrives as an empty array** `[]`, not an error and not a missing field. The empty array is MongoDB's NULL: it says *this student has no fee record* — Hira's story, again.
- **$unwind flattens an array:** it makes one document per array element. After a `$lookup`, unwind turns "student with fee list" into clean per-fee rows — and documents whose array was empty **disappear** (the INNER JOIN behaviour) unless you explicitly preserve them.
- **The decision is per relationship, not per app:** a blog post *embeds* its tags, *references* its author, and its comments may live either way. Ask three things: read together? belongs to one parent? bounded in size? Three yeses — embed. Otherwise — reference.

Code examples

Example 1 — The two shapes, side by side

```
db.students.findOne({ name: 'Ali Raza' })
db.fees.find()
```

What it does: First, embedding in the wild: Ali's document carries his `subjects` *inside* himself — an array that is read whenever Ali is read, and means nothing without him. Second, referencing: the fees collection — four independent documents (September, from MG.6), each pointing home with a `studentId`.

What you will see:

```
{
  _id: ObjectId('650000000000000000000001'),
  name: 'Ali Raza',
  class: '10-A',
  marks: 88,
  city: 'Sahiwal',
  subjects: [ 'Mathematics', 'Physics', 'English' ]
}
[
  { _id: ObjectId('650000000000000000000201'), studentId:
ObjectId('650000000000000000000001'), month: 'September', amount: 5000, status: 'Paid' },
  { _id: ObjectId('650000000000000000000202'), studentId:
ObjectId('650000000000000000000002'), month: 'September', amount: 5000, status: 'Paid' },
  { _id: ObjectId('650000000000000000000203'), studentId:
ObjectId('650000000000000000000003'), month: 'September', amount: 4500, status: 'Paid' },
  { _id: ObjectId('650000000000000000000204'), studentId:
ObjectId('650000000000000000000005'), month: 'September', amount: 5000, status: 'Pending' }
]
```

Subjects inside (small, personal, read together); fees outside (money records the office must query alone — by month, by status — across all students). Neither choice is a compromise; each matches how its data is *used*. That matching is the whole skill of MongoDB design.

Example 2 — $lookup: every student, with his fee record

```
db.students.aggregate([
  {
    $lookup: {
      from: 'fees',
      localField: '_id',
      foreignField: 'studentId',
      as: 'fees'
    }
  }
])
```

What it does: For each student, MongoDB searches `fees` for documents whose `studentId` equals the student's `_id`, and attaches them as a `fees` array. This is the LEFT JOIN of DB.5 rebuilt as a pipeline — all five students come home, matched or not.

What you will see:

```
[
  {
    _id: ObjectId('650000000000000000000001'),
    name: 'Ali Raza',
    class: '10-A',
    marks: 88,
    city: 'Sahiwal',
    fees: [
      { _id: ObjectId('650000000000000000000201'), studentId:
ObjectId('650000000000000000000001'), month: 'September', amount: 5000, status: 'Paid' }
    ]
  },
  {
    _id: ObjectId('650000000000000000000004'),
    name: 'Hira Shah',
    class: '10-B',
    marks: 95,
    city: 'Sahiwal',
    scholarship: true,
    fees: []
  }
]
```

(The other three students arrive the same way — Sara, Bilal and Usman with one fee document each.) Read Hira's entry twice: she is **present**, her facts intact, and her `fees: []` — the empty array — is the honest

MongoDB way of saying what SQL said with NULL: *no fee record exists for her.* Nothing crashed; the absence is data.

Example 3 — Flatten to a report: $lookup + $unwind + $project

```
db.students.aggregate([
  { $lookup: { from: 'fees', localField: '_id', foreignField: 'studentId', as: 'fees' } },
  { $unwind: '$fees' },
  { $project: { _id: 0, name: 1, month: '$fees.month', amount: '$fees.amount', status:
'$fees.status' } }
])
```

What it does: Three stages compose (MG.7's pipeline thinking): join, then unwind each student's `fees` array into separate rows, then project a clean report shape. Documents with an *empty* array have nothing to unwind — they drop out, exactly like DB.5's INNER JOIN.

What you will see:

```
[
  { name: 'Ali Raza', month: 'September', amount: 5000, status: 'Paid' },
  { name: 'Sara Khan', month: 'September', amount: 5000, status: 'Paid' },
  { name: 'Bilal Ahmed', month: 'September', amount: 4500, status: 'Paid' },
  { name: 'Usman Tariq', month: 'September', amount: 5000, status: 'Pending' }
]
```

Four rows — the office's fee report, shaped exactly like the SQL INNER JOIN answer. Hira is absent here, and now you can name *why* in both languages: no fee row (SQL) = empty array, dropped by unwind (MongoDB). Same truth; you are fluent in both.

Quick recap

- **Embed** what is read together, owned by one parent, and small (subjects, sizes, items). **Reference** what is shared, queried alone, or unbounded (fees, comments) — store the other side's `_id`.
- `$lookup` = the pipeline JOIN: `from` + `localField` + `foreignField` + `as`; matches attach as an **array**.
- No match → **empty array** (MongoDB's NULL). `$unwind` flattens arrays into rows — and drops the empties, like an INNER JOIN.
- The embed/reference choice is **per relationship**: read together? one parent? bounded? — three yeses embed.

Common mistakes

- **Referencing everything, SQL-style:** splitting every fact into its own collection means every screen pays a `$lookup` tax — the MG.6 warning in work clothes. If two things are *always* read together, the border between them is costing you on every single read.

- **Embedding the unbounded:** stuffing every comment a post will ever receive inside the post document works for ten comments and dies at ten thousand (documents have a 16 MB ceiling, and every read drags the whole history). Growth without limit is the loudest reference signal there is.

  (Roman Urdu: Jo cheez bina ruke barhti hai — comments, messages — use andar mat band karo; reference do.)

- **Reading `fees: []` as an error:** beginners "fix" the empty array with default fake data. The empty array is the database answering honestly; code should branch on it (`fees.length === 0` → "no fee record"), not paper over it.
- **Forgetting $lookup lives in aggregate:** `db.students.find({ $lookup ... })` does nothing — `$lookup` is a pipeline stage, not a find option. Relations in MongoDB are aggregation work; shape your pipeline accordingly.

Practice

1. Decide embed-or-reference, with one-line reasons: (a) a product's photo URLs, (b) a student's fee history over 5 years, (c) an order's delivery address snapshot, (d) a post's author. Which one was hardest, and what extra fact would settle it?
2. Extend Example 2 with a `$project` that shows only `name` and the fees array. Then add a `$match` *before* the lookup for class '10-B' only — predict exactly which documents arrive, and what their `fees` fields hold.
3. Rewrite Example 3 so Hira *survives* without a fee row (research: `$unwind` has an option for this — find its name) and her report row shows `amount: null`. What SQL join does the result now match?

Practice quiz

1. When do you embed data, and when do you reference it?

- Always embed everything
- Embed what is read together, belongs to one parent and stays small; reference what is shared, big, fast-growing, or often read alone, by storing the other side's _id
- Always reference everything, SQL-style
- Embed only numbers

Answer: Embed what is read together, belongs to one parent and stays small; reference what is shared, big, fast-growing, or often read alone, by storing the other side's _id

2. What does $lookup attach, and what arrives when there is no match?

- A single document, or an error when there is no match
- Matches attached as a new array field named by as, and no match arrives as an empty array [], not an error and not a missing field
- A string, or null
- Nothing, $lookup only filters

Answer: Matches attached as a new array field named by as, and no match arrives as an empty array [], not an error and not a missing field

3. What does $unwind do after a $lookup, and what happens to documents whose array was empty?

- It deletes the array
- It flattens the array into one document per element, and documents whose array was empty disappear, like an INNER JOIN, unless you explicitly preserve them
- It sorts the array
- It does nothing to empty arrays

Answer: It flattens the array into one document per element, and documents whose array was empty disappear, like an INNER JOIN, unless you explicitly preserve them

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/mongodb-lookup-relations · More free tools and notes at https://scopetreasure.com