Eloquent Relationships — hasMany, belongsTo Aur N+1 Ka Hal

Tables that know each other: hasMany and belongsTo, reading a teacher with his students, eager loading — and the famous N+1 problem, caught and cured.

Introduction

Real data is connected. A teacher has many students; every student belongs to one teacher. In the database this connection is one honest column — `teacher_id` on the students table. In your code, Eloquent turns that column into sentences: `$teacher->students`, `$student->teacher->name`. No joins typed by hand, no loops gluing tables together — the models *know* each other.

This lesson builds the teacher–student relation both ways, reads it from both ends, and then meets the most famous performance bug in Laravel: the **N+1 problem** — where an innocent loop quietly fires a hundred tiny queries. You will catch it in the query log and cure it with two words: **eager loading**.

(Roman Urdu: Table akeli nahi rehti — rishtay bandhay hotay hain. Eloquent un rishton ko aap ke code ki zubaan bana deta hai.)

Key points

- **A relationship starts as a foreign key:** the child table carries the parent's id — `students.teacher_id`. The migration adds it (`$table->foreignId('teacher_id')`), and from that column both directions of the story grow.
- **`hasMany` looks downward:** the Teacher model declares `students()` returning `$this->hasMany(Student::class)`. Read it in English: "a teacher HAS MANY students". `$teacher->students` then hands you the whole class as a collection.
- **`belongsTo` looks upward:** the Student model declares `teacher()` returning `$this->belongsTo(Teacher::class)`. `$student->teacher` is the one teacher; `$student->teacher->name` reads like a sentence because it is one.
- **Creating through a relation fills the key for you:** `$teacher->students()->create([...])` writes the new student WITH the correct `teacher_id` — you never type the foreign key by hand and never get it wrong.
- **The naming convention is the wiring, again:** method `students()` + model `Student` + column `teacher_id` — follow the names and Eloquent connects everything silently. Break a name, declare it openly (`hasMany(Student::class, 'mentor_id')`).

  (Roman Urdu: Naam ka usool phir kaam aata hai — teacher_id, students(), Student — teeno milte hain to rishta khud ban jata hai.)
- **N+1 is death by small talk:** fetch 5 students (1 query), then ask each for its teacher inside a loop (5 more queries) = **6 queries**. With 100 students it is 101. The page works — and crawls.

- **Eager loading is the cure:** `Student::with('teacher')->get()` fetches students in one query and ALL their teachers in one more — **2 queries total**, however long the list grows. Load relations you know you will show.

Code examples

Example 1 — Build the rishta: key, and both declarations

```
// a new migration adds the column to students
Schema::table('students', function (Blueprint $table) {
    $table->foreignId('teacher_id')->nullable();
});
```

```
// app/Models/Teacher.php
class Teacher extends Model
{
    public function students()
    {
        return $this->hasMany(Student::class);
    }
}
```

```
// app/Models/Student.php
class Student extends Model
{
    public function teacher()
    {
        return $this->belongsTo(Teacher::class);
    }
}
```

What it does: Teacher 1 is Imran Sheikh, teacher 2 is Fatima Noor. The five students carry keys: Ali, Sara and Bilal → 1; Hira and Usman → 2. Two small methods — one per direction — are the entire wiring.

What you will see: nothing visual yet — but the models now understand each other. Example 2 spends that understanding.

Example 2 — Reading both directions, and creating through the relation

```
$teacher = Teacher::find(1);

foreach ($teacher->students as $student) {
    echo $student->name . ' — ' . $student->marks . '<br>';
}

$student = Student::find(4);
echo $student->name . ' learns from ' . $student->teacher->name;

$teacher->students()->create([
    'name' => 'Zain Malik', 'city' => 'Sahiwal', 'marks' => 81,
]);
```

What it does: Downward first: Imran Sheikh's `students` collection loops like any list. Upward next: Hira (id 4) reaches her teacher in one breath. Finally a new student is created *through* the teacher — the relation writes `teacher_id = 1` itself.

What you will see:

```
Ali Raza — 88
Sara Khan — 92
Bilal Ahmed — 76
Hira Shah learns from Fatima Noor
```

And Zain's new row in the table carries `teacher_id` 1 — set by the relation, never typed.

Example 3 — The N+1 problem: caught in the log, cured with with()

```
// THE SLOW WAY — innocent looking, quietly expensive
$students = Student::all();
foreach ($students as $student) {
    echo $student->name . ' — ' . $student->teacher->name . '<br>';
}

// THE CURE — eager loading
$students = Student::with('teacher')->get();
foreach ($students as $student) {
    echo $student->name . ' — ' . $student->teacher->name . '<br>';
}
```

What it does: Both loops print the identical list. The difference hides in the query log. The slow way asks the database for the students once, then — surprised by each `$student->teacher` — goes back five more times,

once per student. The cured way tells Eloquent in advance: "teachers bhi saath lao". It fetches all teachers in ONE extra query and matches them in memory.

What you will see: same five lines on screen (Ali — Imran Sheikh, Sara — Imran Sheikh, Bilal — Imran Sheikh, Hira — Fatima Noor, Usman — Fatima Noor), but the query count collapses:

```
Slow way:  1 + 5 = 6 queries
With with(): 2 queries — and still 2 when the class grows to 100
```

Quick recap

- Relationships grow from a foreign key (`teacher_id`) + one method on each model: `hasMany` down, `belongsTo` up.
- `$teacher->students` and `$student->teacher` read like sentences; creating through a relation fills the key for you.
- N+1 = 1 query for the list + N queries inside the loop. The cure is eager loading: `with('teacher')` — 2 queries total.

Common mistakes

1. **Looping relations without `with()`** — the classic N+1: works in testing with 5 rows, dies in production with 5,000. Fix: whenever a loop touches a relation, the fetch above it gets `with('relation')`.
2. **Declaring only one direction** — `hasMany` without the matching `belongsTo` leaves half your sentences unwritable. Fix: relations are built in pairs, always, in the same sitting.
3. **Typing foreign keys by hand** — `'teacher_id' => $teacher->id` in create arrays invites silent mismatches. Fix: create through the relation (`$teacher->students()->create(...)`) and let it fill the key.
4. **Method named like the column** — a relation method called `teacher_id()` or a property clash confuses Eloquent's magic. Fix: relations are named for WHAT they return — `teacher()`, `students()` — never for the key column.

Practice

1. Give the school a second relation: a `ClassRoom` model where a room hasMany students and a student belongsTo a room (add `room_id`). Print each room with its students using eager loading, and count your queries both ways.
2. Grow the class to the five series students plus Zain, then print "teacher name → student count" for both teachers using relations only. Expected: Imran Sheikh 4 (with Zain), Fatima Noor 2.

Practice quiz

1. A teacher has many students, a student belongs to a teacher. Which methods declare each direction?

- Teacher belongsTo, Student hasMany
- Teacher hasMany(Student::class) looking downward, Student belongsTo(Teacher::class) looking upward
- Both use hasMany
- Both use belongsTo

Answer: Teacher hasMany(Student::class) looking downward, Student belongsTo(Teacher::class) looking upward

2. You fetch 5 students, then ask each for its teacher inside a loop. How many queries is that, and what is it called?

- 1 query, eager loading
- 6 queries, 1 for the list plus 5 in the loop, the N+1 problem; with 100 students it is 101
- 5 queries, the loop only
- 10 queries, each relation costs two

Answer: 6 queries, 1 for the list plus 5 in the loop, the N+1 problem; with 100 students it is 101

3. What is the cure for N+1, and how many queries does it take for any list length?

- Fetch teachers in a second loop
- Eager loading: Student::with('teacher')->get(), students in one query and all their teachers in one more, 2 queries total however long the list grows
- Cache the page
- Remove the relation

Answer: Eager loading: Student::with('teacher')->get(), students in one query and all their teachers in one more, 2 queries total however long the list grows

---

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/laravel-eloquent-relationships · More free tools and notes at https://scopetreasure.com