Understanding Relationships in MongoDB

In MongoDB, relationships help connect data in different collections (like tables in SQL). Since MongoDB is a **NoSQL** database, it doesn't use **joins** like SQL databases. Instead, we connect data using **references (IDs)** or **embedding (storing related data inside a document).**

There are **three types of relationships** in MongoDB:

1. **One-to-One** – One thing connects to **one** thing (e.g., one person has **one** passport).
2. **One-to-Many** – One thing connects to **many** things (e.g., one user can have **many** posts).
3. **Many-to-Many** – Many things connect to **many** things (e.g., many students can join **many** courses).

One-to-One Relationship (Example in Code)

Let's say we are building an app where each **user** has **one** profile.

Step 1: Create User & Profile Schemas

```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  profile: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Profile", // Reference to Profile model
  },
});

const profileSchema = new mongoose.Schema({
  age: Number,
  address: String,
  phone: String,
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User", // Reference back to User model (optional)
  },
});

const User = mongoose.model("User", userSchema);
const Profile = mongoose.model("Profile", profileSchema);

module.exports = { User, Profile };
```

Step 2: Creating a User and Profile

```
const { User, Profile } = require("./models"); // Import models
const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017/mydatabase");

const createUserAndProfile = async () => {
  const newUser = new User({ name: "Tunde", email: "tunde@example.com" });
  await newUser.save();

  const newProfile = new Profile({
    age: 25,
    address: "Lekki, Lagos",
    phone: "08012345678",
    user: newUser._id, // Linking profile to user
  });

  await newProfile.save();

  newUser.profile = newProfile._id; // Linking user to profile
  await newUser.save();

  console.log("User and Profile created successfully!");
};

createUserAndProfile();
```

Step 3: Fetching User with Profile

```
const getUserWithProfile = async (userId) => {
  const user = await User.findById(userId).populate("profile"); // Fetch user
with profile details
  console.log(user);
};

getUserWithProfile("userId_here"); // Replace with actual user ID
```

Summary (Version)

- **One-to-One** means one thing has exactly one related thing (e.g., **one user = one profile**).
- **We store IDs** in MongoDB to link documents together.
- **`.populate()`** helps to fetch the full profile when getting the user.
