Coding- Activity

MongoDB databases work differently to relational databases. This is also true of relationships.

In MongoDB, you can create a relationship using one of the following two methods:

- Embedded documents.
- Referenced documents.

The method you use will depend on the data, and how you intend to query that data.

Embedded Relationships

With MongoDB, you can embed documents within documents. Therefore, a single document can contain its own relationships.

In fact, we already created a relationship using this method back when we first created a document.

One-to-One Relationship

A *one-to-one relationship* is where the parent document has one child, and the child has one parent.

For example, a business rule might say that an artist can only have one address and that the address can only belong to one artist.

The following code creates a one-to-one relationship, embedded within the document.

```
db.artists.insert(
   {
      _id : 2,
      artistname : "Prince",
      address :   {
                  street : "Audubon Road",
                  city : "Chanhassen",
                  state : "Minnesota",
                  country : "United States"
               }
   }
)
```

Result:

```
WriteResult({ "nInserted" : 1 })
```

One-to-Many Relationship

A *one-to-many* relationship is where the parent document can have many child documents, but the child documents can only have one parent document.

So, another business rule might say that one artist can have many albums, but an album can only belong to one artist.

Running the following code will create a one-to-many relationship:

```
db.artists.insert(
   {
      _id : 3,
      artistname : "Moby",
      albums : [
               {
                  album : "Play",
                  year : 1999,
                  genre : "Electronica"
               },
               {
                  album : "Long Ambients 1: Calm. Sleep.",
                  year : 2016,
                  genre : "Ambient"
               }
            ]
   }
)
```

Result:

```
WriteResult({ "nInserted" : 1 })
```

Document Referenced Relationships

You can use a document reference to create a relationship. Rather than embedding the child document into the parent document (like we did above), you separate the child document out into to its own standalone document.

So we could do this:

Parent Document

```
db.artists.insert(
    {
        _id : 4,
        artistname : "Rush"
    }
)
```

Child Documents

We'll insert 3 child documents — one for each band member:

```
 db.musicians.insert(
    {
        _id : 9,
        name : "Geddy Lee",
        instrument : [ "Bass", "Vocals", "Keyboards" ],
        artist_id : 4
    }
)
```

```
db.musicians.insert(
   {
      _id : 10,
      name : "Alex Lifeson",
      instrument : [ "Guitar", "Backing Vocals" ],
      artist_id : 4
   }
)
 db.musicians.insert(
   {
      _id : 11,
      name : "Neil Peart",
      instrument : "Drums",
      artist_id : 4
   }
)
```

Querying the Relationship

After inserting the above two documents, you can use `$lookup` to perform a left outer join on the two collections.

This, in conjunction with the `aggregate()` method, and `$match` to specify the specific artist you're interested in, will return parent and child documents in one.

```
db.artists.aggregate([
   {
    $lookup:
      {
        from: "musicians",
        localField: "_id",
        foreignField: "artist_id",
        as: "band_members"
      }
  },
  { $match : { artistname : "Rush" } }
]).pretty()
```

Result:

```
{
        "_id" : 4,
        "artistname" : "Rush",
        "band_members" : [
                {
                        "_id" : 9,
                        "name" : "Geddy Lee",
                        "instrument" : [
                                "Bass",
                                "Vocals",
                                "Keyboards"
                        ],
                        "artist_id" : 4
                },
```

```
                {
                        "_id" : 10,
                        "name" : "Alex Lifeson",
                        "instrument" : [
                                "Guitar",
                                "Backing Vocals"
                        ],
                        "artist_id" : 4
                },
                {
                        "_id" : 11,
                        "name" : "Neil Peart",
                        "instrument" : "Drums",
                        "artist_id" : 4
                }
        ]
}
```

You can see that the first two fields are from the artists collection, and the rest of it is from the musicians collection.

So if you only query the artists collection by itself:

db.artists.find({ artistname : "Rush" })

You'd only get this:

{ "_id" : 4, "artistname" : "Rush" }

No related data is returned.

When to use Embedded Documents vs Referenced Documents

Both methods of creating relationships have their pros and cons. There are times you might use embedded documents, and other times you'll use referenced documents.

When to use Embedded Relationships

One of the main benefits of using the embedded relationship method is performance. When the relationship is embedded within the document, queries will run faster than if they were spread out over multiple documents. MongoDB only needs to return the one document, rather than joining multiple documents in order to retrieve the relationships. This can provide a major performance boost — especially when working with lots of data.

Embedded relationships also make queries easier to write. Rather than writing complex queries that join many documents via their unique identifier, you can return all related data within a single query.

Another consideration to keep in mind is that, MongoDB can only ensure atomicity at a document level. Document updates to a single document are always atomic, but not for multiple documents.

When multiple users are accessing the data, there's always a chance that two or more users will try to update the same document with different data. In this case, MongoDB will ensure that no conflict occurs and only one set of data is updated at a time. MongoDB cannot ensure this across multiple documents.

So in general, embedded relationships can be used in most cases, as long as the document remains within the size limit (16 megabytes at the time of writing), and/or its nesting limit (100 levels deep at the time of writing).

However, embedded relationships aren't appropriate for *all* occasions. There may be situations where it makes more sense to create a document referenced relationship.

When to use Referenced Relationships

For data that needs to be repeated across many documents, it can be helpful to have them in their own separate document. This can reduce errors and help in keeping the data consistent (while bearing in mind that multiple-document updates are not atomic).

Using the above example, one musician could be a member (or ex-member) of many bands. Some might also produce albums for other artists, teach students, run clinics, etc. Also, a lot of data could be stored against each musician. So having a separate document for each musician makes sense in this case.

Also, if you think your embedded documents might exceed the file size limit imposed by MongoDB, then you'll need to store some data in separate documents.