mongoDB. Documentation

Search Documentation

Data Models > Data Model Reference > Database References

Database References

On this page

For many use cases in MongoDB, the denormalized data model where related data is stored within a single document will be optimal. However, in some cases, it makes sense to store related information in separate documents, typically in different collections or databases.

IMPORTANT:

MongoDB 3.2 introduces `$lookup` pipeline stage to perform a left outer join to an unsharded collection in the same database. For more information and examples, see `$lookup`.

Starting in MongoDB 3.4, you can also use `$graphLookup` pipeline stage to join an unsharded collection to perform recursive search. For more information and examples, see `$graphLookup`.

This page outlines alternative procedures that predate the `$lookup` and `$graphLookup` pipeline stages.

MongoDB applications use one of two methods for relating documents:

- Manual references where you save the `_id` field of one document in another document as a reference. Then your application can run a second query to return the related data. These references are simple and sufficient for most use cases.
- DBRefs are references from one document to another using the value of the first document's `_id` field, collection name, and, optionally, its database name. By including these names, DBRefs allow documents located in multiple collections to be more easily linked with documents from a single collection.

To resolve DBRefs, your application mu
drivers have helper methods that form t
automatically resolve DBRefs into documents.
DBRefs provide a common format and type to represent relationships among documents. The DBRef format also provides common semantics for representing links between documents if your database must interact with multiple frameworks and tools.

Unless you have a compelling reason to use DBRefs, use manual references instead.

[1] Some community supported drivers may have alternate behavior and may resolve a DBRef into a document automatically.

Manual References

Background

Using manual references is the practice of including one document's `_id` field in another document. The application can then issue a second query to resolve the referenced fields as needed.

Process

Consider the following operation to insert two documents, using the `_id` field of the first document as a reference in the second document:

```
original_id = ObjectId()

db.places.insert({
    "_id": original_id,
    "name": "Broadway Center",
    "url": "bc.example.net"
})

db.people.insert({
    "name": "Erin",
    "places_id": original_id,
    "url":  "bc.example.net/Erin"
})
```

Then, when a query returns the document from the `people` collection you can, if needed, make a second query for the document referenced by the `places_id` field in the `places` collection.

Use

For nearly every case where you want to store a relationship between two documents, use manual references. The references are simple to create and your application can resolve references as needed.

The only limitation of manual linking is that these references do not convey the database and collection names. If you have documents in a single collection that relate to documents in more than one collection, you may need to consider using DBRefs.

DBRefs

Background

DBRefs are a convention for representing a document, rather than a specific reference type. They include the name of the collection, and in some cases the database name, in addition to the value from the `_id` field.

Format

DBRefs have the following fields:

`$ref`

The `$ref` field holds the name of the collection where the referenced document resides.

`$id`

The `$id` field contains the value of the `_id` field in the referenced document.

`$db`

Optional.

Contains the name of the database where the referenced document resides.

Only some drivers support `$db` references.

EXAMPLE:

DBRef documents resemble the following document:

```
{ "$ref" : <value>, "$id" : <value>, "$db" : <value> }
```

Consider a document from a collection that stored a DBRef in a `creator` field:

```
{
  "_id" : ObjectId("5126bbf64aed
  // .. application fields
  "creator" : {
                  "$ref" : "creators",
                  "$id" : ObjectId("5126bc054aed4daf9e2ab772"),
                  "$db" : "users"
               }
}
```

The DBRef in this example points to a document in the `creators` collection of the `users` database that has `ObjectId("5126bc054aed4daf9e2ab772")` in its `_id` field.

NOTE:
The order of fields in the DBRef matters, and you must use the above sequence when using a DBRef.

Driver Support for DBRefs

Driver	DBRef Support	Notes
C	Not Supported	You can traverse references manually.
C++	Not Supported	You can traverse references manually.
C#	Supported	Please see the C# driver page for more information.
Haskell	Not Supported	You can traverse references manually.
Java	Supported	Please see the Java driver page for more information.
Node.js	Supported	Please see the Node.js driver page for more information.

Driver	DBRef Support	Notes
Perl	Supported	Please see the Perl driver page for more information.
PHP	Not Supported	You can traverse references manually.
Python	Supported	Please see the PyMongo driver page for more information.
Ruby	Supported	Please see the Ruby driver page for more information.
Scala	Not Supported	You can traverse references manually.

Use

In most cases you should use the manual reference method for connecting two or more related documents. However, if you need to reference documents from multiple collections, consider using DBRefs.