MongoDB stores JSON documents in a binary representation called [BSON (Binary JSON)](http://bsonspec.org/). BSON encoding extends the popular JSON representation to include additional data types such as int, long, and floating point.

In addition to making it more natural to represent data at the database level, the document model also provides performance and scalability advantages:

* An aggregated document can be accessed with a single call to the database, rather than having to JOIN multiple tables to respond to a query. The MongoDB document is physically stored as a single object, requiring only a single read from memory or disk. On the other hand, RDBMS JOINs require multiple reads from multiple physical locations.
* As documents are self-contained, distributing the database across multiple nodes (a process called sharding) becomes simpler and makes it possible to achieve massive horizontal scalability on commodity hardware. The DBA no longer needs to worry about the performance penalty of executing cross-node JOINs (should they even be possible in the existing RDBMS) to collect data from different tables.

Modeling Relationships with Embedding and Referencing

Deciding when to embed a document or instead create a reference between separate documents in different collections is an application-specific consideration. There are, however, some general considerations to guide the decision during schema design.

Embedding

Data with a 1:1 or 1:Many relationship (where the “many” objects always appear with, or are viewed in the context of their parent documents) is a natural candidate for embedding the referenced information within the parent document. The concept of data ownership and containment can also be modeled with embedding. Using the product data example above, product pricing – both current and historical – should be embedded within the product document since it is owned by and contained within that specific product. If the product is deleted, the pricing becomes irrelevant.

DBAs should also embed fields that need to be modified together atomically. To learn more, refer to the section below on the MongoDB transaction model.

Not all 1:1 or 1:Many relationships should be embedded in a single document. Instead, referencing between documents in different collections should be used when:

* A document is frequently read, but contains an embedded document that is rarely accessed. An example might be a customer record that embeds copies of the annual general report. Embedding the report only increases the in-memory requirements (the working set) of the collection.
* One part of a document is frequently updated and constantly growing in size, while the remainder of the document is relatively static.
* The document size exceeds MongoDB’s current 16MB document limit.

Referencing

Referencing enables data normalization, and can give more flexibility than embedding. But the application will issue follow-up queries to resolve the reference, requiring additional round-trips to the server.

References are usually implemented by saving the [\_id field](https://docs.mongodb.org/manual/reference/object-id/) of one document in the related document as a reference. A second query is then executed by the application to return the referenced data.

Referencing should be used:

* When embedding would not provide sufficient read performance advantages to outweigh the implications of data duplication.
* Where the object is referenced from many different sources.
* To represent complex many-to-many relationships.
* To model large, hierarchical data sets.

Different Design Goals

Comparing these two design options – embedding sub-documents versus referencing between documents – highlights a fundamental difference between relational and document databases:

* The RDBMS optimizes data for storage efficiency (as it was conceived at a time when storage was the most expensive component of the system).
* MongoDB’s document model is optimized for how the application accesses data (as developer time and speed to market are now more expensive than storage).