* [Database](http://voidcanvas.com/category/database/)

Mongoose or mongodb native driver, which one to use? This is one of the initial queries for a node-mongo developer; and probably one of the most important ones. Because one use Object Relational Mapping (ORM) and another Object Document Mapping (ODM); so chainging the driver later on in your project can immensly increase your work. So you need to decide the driver very wisely.

|  | **Mongoose** | **Mongodb native** |
| --- | --- | --- |
| Object mapping | ORM | ODM |
| Schema | Mandatory | Not necessary |
| Performance / processing time | Not bad | Excellent |
| Development time | Fast | Average |
| Default promise | No | No |
| Maintainability | Easy | Little hard |
| Learning curve | Little high | Low |
| Community | Good | Good |

Below is a point by point comparison of the two drivers. Try to relate the scenario of your project with them, I hope you will be able to find out which one is the best for you.

ORM or ODM – What is good for you?

ORM or Object Relational Mapping is based on the principal of having strict models or schemas. In ORM, which mongoose is using, you have to define your schema structure. There has to be a fixed schema.  
ODM or Object Document Mapping is the core concept of mongo itself; and same goes with mongodb native driver. Mongodb doesn’t need any fixed schema. You can insert or update whatever and however you want.

Mongoose:

So, is it a disadvantage to have a fixed Schema? No. Certainly not. This even gives a structure and more maitainability to your application code. This doesn’t hamper the scalability feature of mongo; because if in future if your app grows and there is a need to add few more fields, you can modify the schema and work accordingly (which is certainly not a bis task).  
Eg: If you want to develop a website to sell movie tickets, to store the inventories it’s better to have a fixed schema. Cause you know that the inventories will have fixed properties.

Mongodb native:

There are situation when a fixed schema can become a curse. Suppose you have an online glossary shop. So will you create a schema for every different kind of product; cause all of them have different property sets? Probably this is where you would like to go with a document mapping way and select mongodb native for that.

Performance & Development time

Well, in terms of performace, the low level things are always better; no doubt. But you may face other overheads. Fast processing time may slow down your development time and also the vice versa. Lets see how mongoose and mongodb native stands against performance and development time.

Mongoose:

If you are developing an app for a client which wont be having millions of users or very high read/write concurrency, and you want to develop it fast, than there is no reason not to select mongoose. Development with mongoose is really fast. You don’t need to take the overhead of creating a connection, closing it in proper time. Optimizing the connection, making promises etc. Abstruction layer of mongoose does all these for you.

Mongodb native:

CRUD operations using mongodb native is really faster than mongoose. If you digg in the internet you will find stats also; something like this <http://codeandcodes.com/tag/mongoose-vs-mongodb-native/>  
But if you do a single db operation in each web request, and follow the conventional ways of mongodb (opening a db connection, finish your operation and close it) and you have millions of concurrent requests, your app will die. Cause the proper way would be opening and closing a db connection and handle each request in optimized way. You cant just open and close it for each request. So this optimized handling need to be coded by you if you are making a performant app using mongodb native; for what you have to spend a little extra development time (it’s not that hard thing to do).

Maintainability

For any long term project, maitainability is a big factor. Easy to maintain code reduces your project cost very much. When you have wrapers like mongoose which forces you to create a schema and do things with the help of models, definately gives your project a structure and thus easily maintainable. But this never means mongodb native has a disadvantage on maintainability. If you follow good coding structure, you abviously can make the app good. But the only thing is, you yourself have to write good piece of code.

Learning curve

Neither of them are hard to adopt. But in you have done operations in mongo client in your early days with mongo, than choosing mongodb native would be great for you. Cause it has the same syntax as mongo client and will take zero effort to understand the concept.

Conclusion

Till now I always prefered mongodb native. The reason is, if something is more performent, than I do not mind to put some extra effort to structure my code and optimize the behavior. But may be for any urgent deliverable project I will go with mongoose. You can choose your diver basing on the conditions and according to your situation; but the point I strongly believe is, nothing sould be more important than performance.