*By William Zola, Lead Technical Support Engineer at MongoDB*

这篇文章是系列的最后一篇。在第一篇文章里，我介绍了三种针对“一对多 ”关系建模的基础方案。在第二篇文章中，我介绍了对基础方案的扩展：双向关联和反范式化。

反范式可以让你避免一些应用层级别的join，但是这也会让更新变的更复杂，开销更大。不过冗余那些读取频率远远大于更新频率的字段还是值得的。

如果你还没有读过前两篇文章，欢迎一览。

让我们回顾下这些方案

你可以采取内嵌，或者建立one端或者N端的引用，也可以三者兼而有之。

你可以在one端或者N端冗余多个字段

下面这些是你需要谨记的：

1、优先考虑内嵌，除非有什么迫不得已的原因。

2、需要单独访问一个对象，那这个对象就不适合被内嵌到其他对象中。

3、数组不应该无限制增长。如果many端有数百个文档对象就不要去内嵌他们可以采用引用ObjectID的方案；如果有数千个文档对象，那么就不要内嵌ObjectID的数组。该采取哪些方案取决于数组的大小。

4、不要害怕应用层级别的join：如果索引建的正确并且通过投影条件（第二章提及）限制返回的结果，那么应用层级别的join并不会比关系数据库中join开销大多少。

5、在进行反范式设计时请先确认读写比。一个几乎不更改只是读取的字段才适合冗余到其他对象中。

6、在mongodb中如何对你的数据建模，取决于你的应用程序如何去访问它们。数据的结构要去适应你的程序的读写场景。

**设计指南**

当你在MongoDB中对“一对多”关系进行建模，你有很多的方案可供选择，所以你必须很谨慎的去考虑数据的结构。下面这些问题是你必须认真思考的：

关系中集合的规模有多大：是一对很少，很多，还是非常多？

对于一对多中”多“的那一端，是否需要单独的访问它们，还是说它们只会在父对象的上下文中被访问。

被冗余的字段的读写的比例是多少？

**数据建模设计指南**

在一对很少的情况下，你可以在父文档中内嵌数组。

在一对很多或者需要单独访问“N”端的数据时，你可以采用数组引用ObjectID的方式。如果可以加速你的访问也可以在“N”端使用父引用。

在一对非常多的情况下，可以在“N”端使用父引用。

如果你打算在你的设计中引入冗余等反范式设计，那么你必须确保那些冗余的数据读取的频率远远大于更新的频率。而且你也不需要很强的一致性。因为反范式化的设计会让你在更新冗余字段时付出一定的代价（更慢，非原子化）