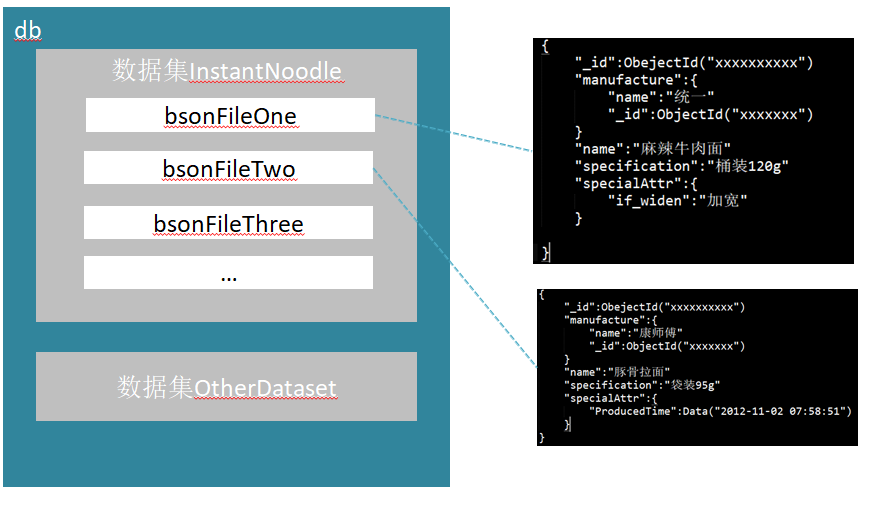
## 文档型数据库

顾名思义，文档型数据库面对的存储对象主要是文档数据或类似的半结构化数据。每个文档对象有自己独有的标识符ID，并以某个标准化格式存储，常见的格式如JSON, YAML, XML, protobuf ，文档以键值对的格式存储数据，同时支持嵌套状态，字段值可以包含其他文档，数组，甚至文档数组。当数据库中没有大量关系和标准化约束，可以分割为多个结构化文档时，使用文档型数据库是非常合适的选择。

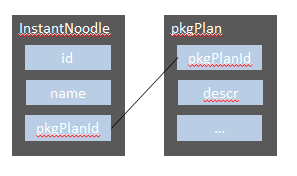
### Mongodb

MongoDB是一个基于C++的开源分布式，文档型数据库，发布于2009年，以提供适用于web的灵活的高性能数据存储方案为目的。它将数据存储为BSON格式（BSON是JSON的衍生类型）的文档。它是NoSQL数据库中最像关系型数据库的一种。这种相似主要体现在面向集合存储的基础上，数据被分开存储在数据集中，每个数据集有自己的独有标识id，包含多个嵌套键值对模式的bson文档，这里数据集和关系型数据库中的表概念相似。但它们的不同点在于MongoDB是模式自由（Schema-free）的，数据集内部的结构并非完全相同，一个数据集可能包括多个格式不同的文档。除此之外，它支持动态查询，支持完全索引，支持分布式。如图是一个示例，db数据库下InstantNoodle数据集，它包含了多个bson格式文件，每个文件的结构并不完全相同。



在本文场景下，测试员在给出属性数据时，可能出现测试属性不同或测试次数不同的情况，所以数据属性可能有细微的区别，结构不完全相同。为了解决这个问题，在传统的关系型数据库中我们有四种解决方案，一，为每种相同属性值的产品建立类型表；这种思路会导致大量冗余表，并且每个类型的查询语句都不一样，是不可取的。二：建立一个包含了所有不同属性的表；这样做消除了方法一查询语句的问题，可以用一个语句查询查询所有类型，但这样增大了存储空间开销，而且表结构会经常变化，需要大量的alter table操作。三，使用继承的思路，为产品中通用的属性建立单独的表，其他独有的属性建立具体的表，这样结合了方法一和二的优点，拥有更好的结构，但当查询时，我们需要进行昂贵的join操作。方法四：设置三元结构：标识id/属性名/属性值。这是最灵活但最冗余的结构，而且查询非常复杂。只能适用于存储数据，读操作很少的情况。

此时我们套用MongoDB模式自由，支持嵌套的思想，这个问题就迎刃而解。我们可以参考关系型中的方法三，在通用的属性外构建键为”specialAttr”的属性，将某类型独有的属性嵌套进去。这样我们就有了更灵活可扩展性更好的数据结构，方便了写操作。而局部嵌入后，读的性能也有所进步，考虑到数据库读数据前的磁盘寻道过程，当关系型数据库使用方法三时，查询数据需要进行大量join操作，也就是需要使用至少一次随机寻道，而MongoDB在磁盘上连续存储文档，在查找时避免了随机寻道，提升了速度和性能。同时，在原子性和独立性上局部嵌入也有优势，假如某方便面有多种包装,现在我们需要不再生产该方便面。如果使用关系型数据库方法三，如图一，我们需要执行一个包含了多个操作的事务：在产品表中删除该产品，在包装表中删除它对应的包装，处理该事务时数据库应该保证其原子性。而如果是MongoDB没有事务的概念，如图二，我们只需要删除这个嵌套对象即可，相比既保证了ACID，又优化了操作的速度。这个例子就体现了MongoDB作为NoSQL几个突出的特性：易扩展，高性能，自由方便。

 图一图二