(This is doc based on a forum post)

This document will discuss some of the different ways to store structured data like JSON objects in FDB. Let's assume one user is about to store data like the following struct:

```
type dbUser struct {
    ID    int64
    Name  string
    Bio string
}
```

There are basically two options: a.) store the whole object as a blob; b.) store each field at a separate key. Let's discuss them side by side.

Store the object as a blob

If you decide to store the object as a single blob, you will need to pay the cost of reading everything back, even just to read a single field. If you always read the entire document anyway (return it as JSON via a REST API) then that does not matter much. However if you have a few fields which are updated frequently (last accessed timestamp, etc...), this option will incur a large serialization overhead. What's more, because the limitation on the value size in FDB, you may need to split a large object into multiple parts.

On the other hand, storing it as a single blob, or in even more aggressive solutions where multiple objects got batched into one single object, provides the opportunity to compress the data on disk. FDB does not provide compression of the key value data so the layers need to take care of that.

Store the object as separate key value pairs

When stored in this way, random access to single field will be much faster and efficient. Implementing `GraphQL` becomes feasible.

However this comes with a cost: disk space. There is little or even zero opportunity to compress the data on disk.

Conclusion

If you're making a generic document database, it'll be hard for you to pick one strategy that works best for all of your customers. You might think about whether you can make the decision configurable, or even adaptive depending on the workload.

For example FDB Document Layer allows the data to be stored as fully expanded, or packed into 4k blocks.

To learn more about compressing data in this case, please see the discussion in the forum post