250-500

A future extension to your tool is to keep a log of the publishing that were performed with your internal DSL (exercise 3) by logging a link to the web page and the timestamp in an external databse.

Choose a relational or NoSQL data store for implementing such an extension of your tool and argue your choice regarding:

* interoperability with the tools used so far
* future maintenance
* performance and exploitation of the data, providing facilities for data analytics

With a key-value store, we can only access an aggregate by lookup based on its key. With a document database, we can submit queries to the database based on the fields in the aggregate, we can retrieve part of the aggregate rather than the whole thing, and the database can create indexes based on the contents of the aggregate.

### 8.4. When Not to Use

There are problem spaces where key-value stores are not the best solution.

#### 8.4.1. Relationships among Data

If you need to have relationships between different sets of data, or correlate the data between different sets of keys, key-value stores are not the best solution to use, even though some key-value stores provide link-walking features.

#### 8.4.2. Multioperation Transactions

If you’re saving multiple keys and there is a failure to save any one of them, and you want to revert or roll back the rest of the operations, key-value stores are not the best solution to be used.

#### 8.4.3. Query by Data

If you need to search the keys based on something found in the value part of the key-value pairs, then key-value stores are not going to perform well for you. There is no way to inspect the value on the database side, with the exception of some products like Riak Search or indexing engines like Lucene [[Lucene]](http://proquest.safaribooksonline.com.ezproxy4.lib.le.ac.uk/9780133036138/idp1249152#bib_41) or Solr [[Solr]](http://proquest.safaribooksonline.com.ezproxy4.lib.le.ac.uk/9780133036138/idp1249152#bib_62).

#### 8.4.4. Operations by Sets

Since operations are limited to one key at a time, there is no way to operate upon multiple keys at the same time. If you need to operate upon multiple keys, you have to handle this from the client side.

**9.3. Suitable Use Cases**

**9.3.1. Event Logging**

Applications have different event logging needs; within the enterprise, there are many different applications that want to log events. Document databases can store all these different types of events and can act as a central data store for event storage. This is especially true when the type of data being captured by the events keeps changing. Events can be sharded by the name of the application where the event originated or by the type of event such as order\_processed or customer\_logged.

**9.3.2. Content Management Systems, Blogging Platforms**

Since document databases have no predefined schemas and usually understand JSON documents, they work well in content management systems or applications for publishing websites, managing user comments, user registrations, profiles, web-facing documents.

**9.3.3. Web Analytics or Real-Time Analytics**

Document databases can store data for real-time analytics; since parts of the document can be updated, it’s very easy to store page views or unique visitors, and new metrics can be easily added without schema changes.

**9.3.4. E-Commerce Applications**

E-commerce applications often need to have flexible schema for products and orders, as well as the ability to evolve their data models without expensive database refactoring or data migration (“[Schema Changes in a NoSQL Data Store](http://proquest.safaribooksonline.com.ezproxy4.lib.le.ac.uk/9780133036138/idp1100000#ch12lev1sec3),” p. [128](http://proquest.safaribooksonline.com.ezproxy4.lib.le.ac.uk/9780133036138/idp1026512#page_128)).

Cassandra is one of the popular column-family databases; there are others, such as HBase, Hypertable, and Amazon DynamoDB [[Amazon DynamoDB]](http://proquest.safaribooksonline.com.ezproxy4.lib.le.ac.uk/9780133036138/idp1249152#bib_03). Cassandra can be described as fast and easily scalable with write operations spread across the cluster. The cluster does not have a master node, so any read and write can be handled by any node in the cluster.

I need to explain why I am using nosql rather than SQL and which nosql I am going to use and which technology of that nosql I am going to use.

My own notes

We are recording log data which needs to have good write performance since we will be constantly writing to the database after every post / articles. That’s where nosql is good. We won’t be reading much as far as the initial specs goes. Sql wont be able to do this. Constantly writing to disk will does require a lot of disk space since posts are recorded constantly. The cheapest and most effient way of expanding disk space with regards to data persistence is to scale it horizontally meaning we will have to increase the amount of clusters storing our log data instead of increase a single server’s hardware which is known as verticle scaling. This is more expensive and ultimately performance will still be an issue for various other reasons. That is why we have opted for NOSQL which can be used to scale horizontally.

We will use Cassandra since it’s the most widely used Column Family also known as Column Wide database. Cassandra is backed by Apache with regards to development even though it is an open source technology and people are constantly. There is cloud infrastructure for Cassandra which will help us using continues delivery in an Agile way. Cassandra