*So while a supposed Schema-less NoSQL data-store will in theory allow you to store any data you like (typically key value pairs, in a document) without prior knowledge of the keys, or data types, it will be pointless unless you have some mechanism to retrieve and use the data. So essentially the schema is partially moved from the RDBMS into the application code. I say partially as you'll have added Indexes to document collections and or partitioned the data for performance, so the NoSQL DBMS will have a partial schema defined locally, and possibly enforced via unique constraints.*

*As to adding additional attributes to a document/object in the store. Depending on how much padding is around the document (un-used space), in its physical data block, adding a few more key value pairs to the documents may result in the document having to be physically moved to a larger contiguous block of storage, and the associated indexes re-built. If you plan to use the new keys in a frequently utilized query then you'll be wanting to also add a suitable new index, which will obviously require some physical storage, take a while to initially build and possibly lead you to ask the sys admin to allocate more memory to the DBMS, to allow the new index(s) to be cached.*

*minimum number of column family:-*

*Where multiple Column Families exist in a single table, be aware of the cardinality (i.e., number of rows). If Column Family A has 1 million rows and Column Family B has 1 billion rows, Column Family A’s data will likely be spread across many, many regions . This makes mass scans for Column Family A less efficient.*

*Try to make do with one column family if you can in your schemas. Only introduce a second and third column family in the case where data access is usually column scoped; i.e. you query one column family or the other but usually not both at the one time.*

*HBase currently does not do well with anything above two or three column families so keep the number of column families in your schema low. Currently, flushing and compactions are done on a per Region basis so if one column family is carrying the bulk of the data bringing on flushes, the adjacent families will also be flushed even though the amount of data they carry is small. When many column families exist the flushing and compaction interaction can make for a bunch of needless i/o.*

*benefit of connection pool in Hbase:-*

* Connection pool in Hbase is used when we need to scan the row of a table very frequently in an hour. For example if we have the information about row id.we are creating row id for creating start row and end row which are essentially the same in all cases.
* Implementing connection pooling is also used for improving the performance.