

1. **HBase is a schema less database, what does it mean?**

Schema less database 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. HBase is one of a schema less database which is also a NoSql database. NoSql is Partial Schema enforced by the DBMS on Write, PLUS schema fully enforced by the Application on Read. Most of the time adjustments to the datatbase become transparent and atomic. For example, if we wish to add GPA to the student objects, we add the attribute, resave, and all is well – if we look up an existing student and reference GPA, we just get back null. Further, if we roll back our code, the new GPA fields in the existing objects are unlikely to cause problems if our code was well written.

Some Features of Schema less database.

* Generic data model

Heterogeneous containers, including sets, maps, and arrays

* + Dynamic type discovery and conversion

NoSQL analytics systems support runtime type identification and conversion so that custom business logic can be used to dictate analytic treatment of variation.

* + Non-relational and De-normalised  
     Data is stored in single tables as compared to joining multiple tables.
  + Commodity hardware

Adding more of the economical servers allows NoSQL databases to scale to handle more data.

* + Highly distributable

Distributed databases can store and process a set of information on more than one device.

1. **What is the minimum number of column family every HBase table should have?**

In HBase every table should have at least a minimum of one column family.

1. **What is the benefit of using connection pool in HBase?**

A cluster connection encapsulating lower level individual connections to actual servers and a connection to zookeeper. Connections are instantiated through the [ConnectionFactory](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/ConnectionFactory.html" \o "class in org.apache.hadoop.hbase.client) class. The lifecycle of the connection is managed by the caller, who has to [close()](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Connection.html#close--) the connection to release the resources.

The connection object contains logic to find the master, locate regions out on the cluster, keeps a cache of locations and then knows how to re-calibrate after they move. The individual connections to servers, meta cache, zookeeper connection, etc are all shared by the [Table](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Table.html) and [Admin](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Admin.html) instances obtained from this connection.

Connection creation is a heavy-weight operation. Connection implementations are thread-safe, so that the client can create a connection once, and share it with different threads. [Table](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Table.html) and [Admin](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Admin.html" \o "interface in org.apache.hadoop.hbase.client)instances, on the other hand, are light-weight and are not thread-safe. Typically, a single connection per client application is instantiated and every thread will obtain its own Table instance. Caching or pooling of [Table](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Table.html) and [Admin](https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Admin.html) is not recommended.