**Explain in brief**

**● Hbase is a schema less database, what does it mean?**

HBase is a wide column store and has column families which are roughly equivalent to tables. The column names can be completely variable and the number of columns can vary by row.

Without HBase you can’t do table joins – and so is definitely more of a “schema-less” nature.

HBase is a column-oriented database and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase:

Table is a collection of rows.

Row is a collection of column families.

Column family is a collection of columns.

Column is a collection of key value pairs.

What this really means is that the "schema" is stored with the record, not the table. In a RDBMS, the schema is defined and that table has the schema. In HBase (and other BigTable implementations) data is labeled with its types.

**● What is the minimum number of column family every Hbase table should have?**

There is a limit to the number of column families in HBase. There is one MemStore(Its a write cache which stores new data before writing it into Hfiles) per Column Family, when one is full, they all flush.

HBase currently does not do well with anything above two or three column families so keep the number of column families in the 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 though the amount of data they carry is small.

* **What is the benefit of using connection pool in Hbase?**

For applications which require high-end multithreaded access (e.g., web-servers or application servers that may serve many application threads in a single JVM), we can pre-create an HConnection, as shown in the following example:

Pre-Creating a HConnection

// Create a connection to the cluster.

HConnection connection = HConnectionManager.createConnection(Configuration);

HTableInterface table = connection.getTable("myTable");

// use table as needed, the table returned is lightweight

table.close();

// use the connection for other access to the cluster

connection.close();

Connections are "heavy" objects that are expensive to create, but thread-safe. Table, Admin, and BufferedMutatator objects are lightweight and not necessarily thread-safe. Connection objects don't necessarily translate to a single "connection", and can multiplex behind the scenes.