**Assignment 31.3**

**Problem Statement:**

Explain in brief:

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

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

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

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

**Answer:**

Apache Hbase is the Hadoop database, a distributed, column oriented, scalable, big datastore.Use Apache HBase when you need random, real-time read/write access to your Big Data.

Apache HBase is an open-source, distributed, versioned, non-relational database modelled after Google's Bigtable: A Distributed Storage System for Structured Data by Chang et al.Just as Bigtable leverages the distributed data storage provided by the Google FileSystem, Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.

A schema-less database does not require conformation to a rigid schema (database, schema, data types, tables etc.) that one is required to live up to through the life of a system.

Does not enforce data type limitations on individual values pertaining to one single column type

Models the business usage and not a database schema, application or product.

Can store structured and unstructured data.

Eliminates the need to introduce additional layers (ORM layer) to abstract the relational model and expose it in an object oriented format.

Schema-less databases store data as Key/Value pairs (also known as KV) or as JSON documents. Based on the use cases users have the choice to either store data as KV pairs or as JSON documents.

JSON documents are generally very rich in the way data is represented and allows users to very closely model the entity relationship model that we are all very familiar with and have found very useful.

For e.g.: An Account entity can be modeled in a JSON document with all the required attributes and the nested values that go with a typical Account object – the multiple address, emails, aliases etc.

JSON documents also provide the added advantage of being able to index individual values making the access much more performant, allowing pieces of data from different documents to be joined together.

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

**Answer:**

The minimum number of column family every Hbase table should have is 1.

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 though the amount of data they carry is small.

When many column families the flushing and compaction interaction can make for a bunch of needless I/O loading (To be addressed by changing flushing and compaction to work on a per column family basis).

**Cardinality of ColumnFamilies:**

Where multiple ColumnFamilies exist in a single table, be aware of the cardinality (i.e., number of rows). If ColumnFamilyA has 1 million rows and ColumnFamilyB has 1 billion rows, ColumnFamilyA's data will likely be spread across many, many regions (and RegionServers). This makes mass scans for ColumnFamilyA less efficient.

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

**Answer:**

The Connection Creation section described when connections are created. It described how several Context instances can share the same connection. In this type of sharing, operations from the different Context instances are multiplexed onto the same connection.

You can control the degree of sharing by deciding when to create a new initial context and when to obtain a derived Context instance from an existing Context instance. This type of connection sharing satisfies most applications.

Another type of connection sharing supported by the LDAP service provider is called connection pooling. In this type of sharing, the LDAP service provider maintains a pool of (possibly) previously used connections and assigns them to a Context instance as needed.

When a Context instance is done with a connection (closed or garbage collected), the connection is returned to the pool for future use. Note that this form of sharing is sequential: a connection is retrieved from the pool, used, returned to the pool, and then, retrieved again from the pool for another Context instance.

The pool of connections is maintained per Java runtime system. As such, it can be used by all of the ways in which a connection can be created, as described in the Connection Creation section.

For example, a connection required for processing a referral can use a pooled connection, as can a connection required for constructing an InitialContext. For some situations, this can improve performance significantly.

For example, only one connection is required for processing a search response that contains four referral references to the same LDAP server if connection pooling is used. Without connection pooling, such a scenario would require four separate connections.

A pooled connection can be shared by multiple Context instances, using the same model described in the Connection Creation section. The only difference is that the underlying connection might have come from a pool instead of being created fresh.