**Q-Explain in brief**

**● Sequence File Format**

**● NLine Input Format**

**● DB Input Format**

**● DB Output Format**

**Solution:**

**Sequence File Format:**

1. Sequence files are flat files containing key, value pairs.

2. A very common use case when designing ingestion systems is to use Sequence files as containers and store any file related metadata (filename, path, creation time etc.) as the key and the file contents as the value.

3. A sequence file has a header which contains information on the key/value class names, version, file format, metadata about the file and sync marker to denote the end of the header.

4. The header is followed by the records which constitute the key/value pairs and their respective lengths.

5. A Sequence file can be have three different formats: An Uncompressed format, a Record Compressed format where the value is compressed and a Block Compressed format where entire records are compressed.

6. There are sync markers for every few 100 bytes (approximately) that represent record boundaries.

 7. Sequence files are splittable with each map task processing a split, with one or many key/value pairs.

8. Each call to map() method in the Mapper would retrieve the next key and value in the corresponding split. Even when a sequence file split cuts off in the middle of a record, the sequence file reader will read until a sync marker is reached ensuring that a record is read in whole.

**NLine Input Format:**

1. NLineInputFormat which splits N lines of input as one split.

2. In many "pleasantly" parallel applications, each process/mapper processes the same input file (s), but with computations are controlled by different parameters.(Referred to as "parameter sweeps").

3. One way to achieve this, is to specify a set of parameters (one set per line) as input in a control file (which is the input path to the map-reduce application, where as the input dataset is specified via a config variable in JobConf.).

4. The NLineInputFormat can be used in such applications, that splits the input file such that by default, one line is fed as a value to one map task, and key is the offset. i.e. (k,v) is (LongWritable, Text). The location hints will span the whole mapred cluster.

**DB Input Format:**

1. The DBInputFormat component provided in Hadoop 0.19 finally allows easy import and export of data between Hadoop and many relational databases, allowing relational data to be more easily incorporated into your data processing pipeline.

2. DBInputFormat uses JDBC to connect to data sources. Because JDBC is widely implemented, DBInputFormat can work with MySQL, PostgreSQL, and several other database systems.

3. The DBInputFormat is an InputFormat class that allows you to read data from a database.

4. An InputFormat is Hadoop’s formalization of a data source; it can mean files formatted in a particular way, data read from a database, etc.

5. DBInputFormat provides a simple method of scanning entire tables from a database, as well as the means to read from arbitrary SQL queries performed against the database.

**DB Output Format:**

1. DBOutputFormat accepts <key,value> pairs, where key has a type extending DBWritable. Returned RecordWriter writes only the key to the database with a batch SQL query.

2. The DBOutputFormat writes to the database by generating a set of INSERT statements in each reducer.

3. The reducer’s close() method then executes them in a bulk transaction.

4. Performing a large number of these from several reduce tasks concurrently can swamp a database.

5. If you want to export a very large volume of data, you may be better off generating the INSERT statements into a text file, and then using a bulk data import tool provided by your database to do the database import.