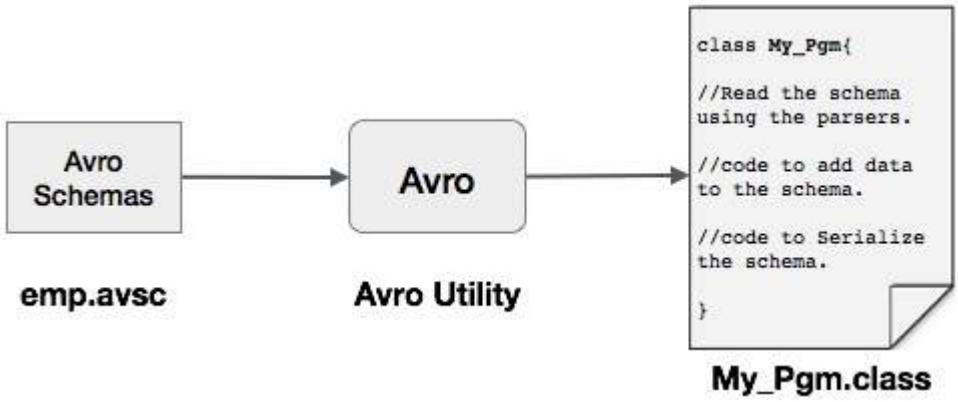
AVRO - SERIALIZATION USING PARSERS

One can read an Avro schema into a program either by generating a class corresponding to a schema or by using the parsers library. In Avro, data is always stored with its corresponding schema. Therefore, we can always read a schema without code generation.

This chapter describes how to read the schema **by using parsers library** and to **serialize** the data using Avro.

The following is a depiction of serializing the data with Avro using parser libraries. Here, emp.avsc is the schema file which we pass as input to Avro utility.



Serialization Using Parsers Library

To serialize the data, we need to read the schema, create data according to the schema, and serialize the schema using the Avro API. The following procedure serializes the data without generating any code −

Step 1

First of all, read the schema from the file. To do so, use **Schema.Parser** class. This class provides methods to parse the schema in different formats.

Instantiate the **Schema.Parser** class by passing the file path where the schema is stored.

```
Schema schema = new Schema.Parser().parse(new File("/path/to/emp.avsc"));
```

Step 2

Create the object of **GenericRecord** interface, by instantiating **GenericData.Record** class. This constructor accepts a parameter of type Schema. Pass the schema object created in step 1 to its constructor as shown below −

```
GenericRecord e1 = new GenericData.Record(schema);
```

Step 3

Insert the values in the schema using the **put** method of the **GenericData** class.

```
e1.put("name", "ramu");
e1.put("id", 001);
e1.put("salary",30000);
e1.put("age", 25);
e1.put("address", "chennai");
```

Step 4

Create an object of **DatumWriter** interface using the **SpecificDatumWriter** class. It converts Java objects into in-memory serialized format. The following example instantiates **SpecificDatumWriter** class object for **emp** class −

```
DatumWriter<emp> empDatumWriter = new SpecificDatumWriter<emp>(emp.class);
```

Step 5

Instantiate **DataFileWriter** for **emp** class. This class writes serialized records of data conforming to a schema, along with the schema itself, in a file. This class requires the **DatumWriter** object, as a parameter to the constructor.

```
DataFileWriter<emp> dataFileWriter = new DataFileWriter<emp>(empDatumWriter);
```

Step 6

Open a new file to store the data matching to the given schema using **create** method. This method requires two parameters −

- the schema,
- the path of the file where the data is to be stored.

In the example given below, schema is passed using **getSchema** method and the serialized data is stored in **emp.avro**.

```
empFileWriter.create(e1.getSchema(), new File("/path/to/emp.avro"));
```

Step 7

Add all the created records to the file using **append** method as shown below.

```
empFileWriter.append(e1);
empFileWriter.append(e2);
empFileWriter.append(e3);
```

Example – Serialization Using Parsers

The following complete program shows how to serialize the data using parsers −

```
import java.io.File;
import java.io.IOException;

import org.apache.avro.Schema;
import org.apache.avro.file.DataFileWriter;

import org.apache.avro.generic.GenericData;
import org.apache.avro.generic.GenericDatumWriter;
import org.apache.avro.generic.GenericRecord;

import org.apache.avro.io.DatumWriter;

public class Seriali {
   public static void main(String args[]) throws IOException{

      //Instantiating the Schema.Parser class.
      Schema schema = new Schema.Parser().parse(new
File("/home/Hadoop/Avro/schema/emp.avsc"));

      //Instantiating the GenericRecord class.
      GenericRecord e1 = new GenericData.Record(schema);

      //Insert data according to schema
      e1.put("name", "ramu");
```

```
      e1.put("id", 001);
      e1.put("salary",30000);
      e1.put("age", 25);
      e1.put("address", "chenni");

      GenericRecord e2 = new GenericData.Record(schema);

      e2.put("name", "rahman");
      e2.put("id", 002);
      e2.put("salary", 35000);
      e2.put("age", 30);
      e2.put("address", "Delhi");

      DatumWriter<GenericRecord> datumWriter = new
GenericDatumWriter<GenericRecord>(schema);

      DataFileWriter<GenericRecord> dataFileWriter = new
DataFileWriter<GenericRecord>(datumWriter);
      dataFileWriter.create(schema, new
File("/home/Hadoop/Avro_work/without_code_gen/mydata.txt"));

      dataFileWriter.append(e1);
      dataFileWriter.append(e2);
      dataFileWriter.close();

      System.out.println("data successfully serialized");
   }
}
```

Browse into the directory where the generated code is placed. In this case, at **home/Hadoop/Avro_work/without_code_gen**.

```
$ cd home/Hadoop/Avro_work/without_code_gen/
```

Now copy and save the above program in the file named **Serialize.java**. Compile and execute it as shown below −

```
$ javac Serialize.java
$ java Serialize
```

Output

```
data successfully serialized
```

If you verify the path given in the program, you can find the generated serialized file as shown below.