

# Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)

[👤 Membership ▾](#) [Your Career ▾](#)

[Home](#) › [bigdata-success.com](#) › [Tutorials - Big Data](#) › [TUT - File Formats](#) › 06: Avro

Schema evolution tutorial

## 06: Avro Schema evolution tutorial

 Posted on [December 18, 2016](#)

**Q1.** What do you understand by the term “AVRO schema evolution”?

**A1.** Schema evolution is the term used for how the store behaves when Avro schema is changed after data has been written to the store using an older version of that schema.

**Q2.** When does the schema evolution take place?

**A2.** During **deserialization**. In other words, when reading an avro file that was written with an older schema, and you can read it with a newer (i.e. evolved) schema provided you have defined the **default values** in your schema.

### 300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key Areas Q&As



150+ Java Architect FAQs



80+ Java Code Quality Q&As



150+ Java Coding Q&As



### 300+ Big Data Interview FAQs

300+ Big Data FAQs



Tutorials - Big Data



TUT -  Starting Big Data

TUT - Starting Spark & Scala

## employee-ver1.avsc

```
1 {
2   "type": "record",
3   "namespace": "com.myapp",
4   "name": "organization",
5   "fields": [
6     {
7       "name": "name",
8       "type": "string",
9       "doc": "The employee name",
10      "default": null
11    },
12    {
13      "name": "title",
14      "type": "string",
15      "doc": "employee title",
16      "default": null
17    }
18  ]
19 }
20 ]
21 }
22 }
```

## employee-ver2.avsc

The schema has evolved with an additional field named “salary”. It defines a “default” as “null”.

```
1 {
2   "type": "record",
3   "namespace": "com.myapp",
4   "name": "organization",
5   "fields": [
6     {
7       "name": "name",
8       "type": "string",
9       "doc": "The employee name",
10      "default": null
11    },
12    {
13      "name": "title",
14      "type": "string",
15      "doc": "employee title",
16      "default": null
17    },
18    {
19      "name": "salary",
20      "type": "double",
21      "doc": "employee salary",
22      "default": null
23    }
24  ]
25 }
```

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

## 800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



```

19         "default": null
20     }
21     ,
22     {
23         "name": "salary",
24         "type": ["null", "double" ],
25         "doc": "employee salary",
26         "default": null
27     }
28 ]
29 }
30

```

## Java code to write/read to/from sequence file

It initially writes and reads with “employee-ver1.avsc”, and then with evolved schema “employee-ver2.avsc”.

```

1
2 package com.mytutorial;
3
4 import java.io.IOException;
5 import java.net.URI;
6 import java.net.URL;
7
8 import org.apache.avro.Schema;
9 import org.apache.avro.file.DataFileReader;
10 import org.apache.avro.file.DataFileWriter;
11 import org.apache.avro.file.FileReader;
12 import org.apache.avro.file.SeekableInput;
13 import org.apache.avro.generic.GenericData;
14 import org.apache.avro.generic.GenericDatumReader;
15 import org.apache.avro.generic.GenericDatumWriter;
16 import org.apache.avro.generic.GenericRecord;
17 import org.apache.avro.io.DatumReader;
18 import org.apache.avro.mapred.FsInput;
19 import org.apache.hadoop.conf.Configuration;
20 import org.apache.hadoop.fs.FSDataOutputStream;
21 import org.apache.hadoop.fs.FileSystem;
22 import org.apache.hadoop.fs.Path;
23
24 public class CreateOrAppendToAvroOnHdfs {
25
26     public static void main(String[] args) {
27
28         Configuration hdfsConf = new Configuration();
29
30         try {
31             String uri = "hdfs://localhost:8020/

```

```
32     FileSystem hdfs = FileSystem.get(URI
33
34
35     Path filePath = new Path(uri);
36     hdfs.setReplication(filePath, (short)1);
37
38     URL avroUrlSchemaVersion1 = CreateOriginalSchema
39     Schema avroSchemaVersion1 = new Schema.Parser().par
40
41     writeToSequenceFile(hdfs, filePath, hdfsConfiguration);
42     readFromSequenceFile(filePath, hdfsConfiguration);
43
44
45     URL avroUrlSchemaVersion2 = CreateOriginalSchema
46     Schema avroSchemaVersion2 = new Schema.Parser().par
47
48     readFromSequenceFile(filePath, hdfsConfiguration);
49
50
51 } catch (IOException ex) {
52     ex.printStackTrace();
53 } finally {
54 }
55 }
56 }
57
58 private static void writeToSequenceFile(FileSystem hdfs, Path filePath, Configuration hdfsConfiguration) {
59
60     DataFileWriter<GenericRecord> fileWriter = null;
61     DataFileWriter<GenericRecord> appendFileWriter = null;
62     FSDataOutputStream out = null;
63
64     GenericDatumWriter<GenericRecord> datumWriter = new GenericDatumWriter<GenericRecord>();
65     fileWriter = new DataFileWriter<GenericRecord>(datumWriter);
66
67
68     if (!hdfs.exists(filePath)) {
69         // If the path doesn't exist, create it
70         out = hdfs.create(filePath);
71         appendFileWriter = fileWriter.create(out);
72
73     } else {
74         // Otherwise just append to the existing file
75         out = hdfs.append(filePath);
76         appendFileWriter = fileWriter.append(out);
77     }
78
79     GenericRecord myRecord = new GenericDatumWriter<GenericRecord>().toRecord(new GenericDatumWriter<GenericRecord>().toDatum("name", "John"), new GenericDatumWriter<GenericRecord>().toDatum("title", "VP"));
80     myRecord.put("name", "John");
81     myRecord.put("title", "VP");
82
83
84     appendFileWriter.append(myRecord);
85     appendFileWriter.close();
86     fileWriter.close();
```

```
87     }
88
89     private static void readFromSequenceFile(Path path) throws IOException {
90         SeekableInput input = new FsInput(filePath);
91         DatumReader<GenericRecord> reader = new GenericDatumReader<>();
92         FileReader<GenericRecord> fileReader = new FileReader<>(input, reader);
93
94         for (GenericRecord datum : fileReader) {
95             System.out.println("value = " + datum);
96         }
97
98         fileReader.close();
99     }
100 }
101
```

## Output

```
1
2 value = {"name": "John", "title": "VP"}
3 value = {"name": "John", "title": "VP", "salary": 100000}
4
```

**Q3.** What schema modifications can you safely perform?

**A3.**

- 1) A field with a default value is added, and a field that was previously defined with a default value is removed.
- 2) A field's doc or order attribute is changed, added or removed.
- 3) A field's default value is added, or changed.
- 4) Field or type aliases are added, or removed.
- 5) A non-union type may be changed to a union that contains only the original type, or vice-versa.

◀ 01b: Convert XML file To Sequence File – writing & reading – Hadoop File

System (i.e HDFS)

10: Q80 – Q87 HBase Schema Design Interview Q&As ▶

## Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#).

© 2022 [java-success.com](https://www.java-success.com)