Avro file stores data and its definition (schema) together. Schema is defined in JSON format

Schema evolution is a key feature of Avro in this case schema can change over time. For instance

* Adding new fields
* Removing old fields

Avro cleanly handles schema change like

* Missing fields
* Newly added fields
* Changed fields

Because of this

1. Old program can read data based on new schema

2. New program can read data based on old schema

Suppose we have department’s data like shown below

|  |
| --- |
| 1,Electronics  2,Fitness  3 ,Footwear  4 ,Apparel  5,Golf  6 ,Outdoors  7,Fan Shop |

This is avro schema file namely department.avsc that we need to specify at the time of table creation

|  |
| --- |
| {  "name": "departments\_information",  "type": "record",  "fields": [  {  "name":"department\_id",  "type":"int"  },  {  "name":"department\_name",  "type": ["string","null"],  "doc":"department name of retail store"  }  ]  } |

Create Avro table

|  |
| --- |
| CREATE TABLE departments\_avro  STORED AS AVRO  TBLPROPERTIES ('avro.schema.url'='hdfs://mac127:8020/user/test/retail\_stage/departments.avsc'); |

Or we can define property directly also

|  |
| --- |
| CREATE TABLE departments\_avro  STORED AS AVRO  TBLPROPERTIES ('avro.schema.literal'='{  "name": "departments\_information",  "type": "record",  "fields": [  {  "name":"department\_id",  "type":"int"  },  {  "name":"department\_name",  "type": ["string","null"],  "doc":"department name of retail store"  }  ]  }'); |

Create normal text table

|  |
| --- |
| create table departments\_text  (department\_id int,  department\_name string  ) ROW FORMAT DELIMITED  FIELDS TERMINATED BY ','  STORED AS TEXTFILE; |

Load data into text table

|  |
| --- |
| load data local inpath '/root/department.data' into table departments\_text; |

Now insert data from text to avro

|  |
| --- |
| insert into departments\_avro select \* from departments\_text; |

Now check data in department avro table

|  |
| --- |
| hive> select \* from departments\_avro;  OK  1 Electronics  2 Fitness  3 Footwear  4 Apparel  5 Golf  6 Outdoors  7 Fan Shop |

Now change avro schema, here we are adding two new fields

|  |
| --- |
| {  "name": "departments\_information",  "type": "record",  "fields": [  {  "name":"department\_id",  "type":"int"  },  {  "name":"department\_manager\_id",  "type":"int",  "default":-1  },  {  "name":"department\_name",  "type": ["string","null"],  "doc":"department name of retail store"  },  {  "name":"department\_location",  "type":"string",  "default":"null"  }  ]  } |

Now again put this file to same hdfs location

|  |
| --- |
| hdfs dfs -put -f departments.avsc hdfs://mac127:8020/user/test/retail\_stage/ |

Now again check the avro data

|  |
| --- |
| hive> select \* from departments\_avro;  OK  1 -1 Electronics null  2 -1 Fitness null  3 -1 Footwear null  4 -1 Apparel null  5 -1 Golf null  6 -1 Outdoors null  7 -1 Fan Shop null  Time taken: 0.191 seconds, Fetched: 7 row(s) |

When you check data file you can see that schema is stored with data itself

|  |
| --- |
| hive> dfs -cat /user/hive/warehouse/departments\_avro/000000\_0;  Objavro.schema▒{"type":"record","name":"departments\_information","fields":[{"name":"department\_id","type":"int"},{"name":"department\_name","type":["string","null"],"doc":"department name of retail store"}]}]▒;kp▒▒κ▒▒S▒▒ElectronicsFitnessFootweaApparel Golf |

Now we are inserting two new records that includes updated schema

|  |
| --- |
| insert into departments\_avro values (100,100,'testing1','pune'),(200,200,'testing2','mumbai'); |

|  |
| --- |
| hive> dfs -ls /user/hive/warehouse/departments\_avro/;  Found 2 items  -rwxrwxrwt 3 root hive 320 2017-04-07 16:13 /user/hive/warehouse/departments\_avro/000000\_0  -rwxrwxrwt 3 root hive 408 2017-04-07 16:32 /user/hive/warehouse/departments\_avro/000000\_0\_copy\_1 |

Now if you check new data file it has update schema

|  |
| --- |
| hive> dfs -cat /user/hive/warehouse/departments\_avro/000000\_0\_copy\_1;  Objavro.schema▒{"type":"record","name":"departments\_information","fields":[{"name":"department\_id","type":"int"},{"name":"department\_manager\_id","type":"int","default":-1},{"name":"department\_name","type":["string","null"],"doc":"department name of retail store"},{"name":"department\_location","type":"string","default":"null"}]}}▒▒k▒\*(▒?▒▒(▒▒▒P▒▒testingpune▒▒testing2 |

So basically here we have two data files one with old schema and other with new schema

Now again check avro data

|  |
| --- |
| hive> select \* from departments\_avro;  OK  1 -1 Electronics null  2 -1 Fitness null  3 -1 Footwear null  4 -1 Apparel null  5 -1 Golf null  6 -1 Outdoors null  7 -1 Fan Shop null  100 100 testing1 pune  200 200 testing2 mumbai  Time taken: 0.105 seconds, Fetched: 9 row(s) |

We can see that we get all data without any modification and that is example of schema evolution