Avro Intermediate Data Format

Title: **Avro Intermediate Data Format**

JIRA : https://issues.apache.org/jira/browse/SQOOP-1902

Summary

The connector-sdk package in sqoop currently supports CSVIDF and JSONIDF. Th goal of this ticket is to use avro `GenericRecord` to represent the sqoop data as it transfers from the "FROM" part of the sqoop job to the "TO" part of the sqoop job

How does the connectors use the AvroIDF?

By declaring in the connector implementation class as below. This will direct sqoop to store the data flowing from the FROM to the TO in the sqoop job in avro format. i.e the in memory intermediate representation will always be a avro record with its schema.

```
/**
  * Returns the {@linkplain IntermediateDataFormat} this connector
  * can return natively in. This will support retrieving the data as text
  * and an array of objects. This should never return null.
  *
  * @return {@linkplain IntermediateDataFormat} object
  */
 public Class<? extends IntermediateDataFormat<?>> getIntermediateDataFormat() {
   return AvroIntermediateDataFormat.class;
 }
```

Background

Read IDFAPI for more information on the core aspects of the IDF.

Requirements

- Create a IDF implementation that represents sqoop data in avro `GenericRecord`
- The source of truth stored in memory is the avro record, which is the native format, the remaining formats i.e text and object array are constructed "***lazily***" if and when invoked by the underlying code using the IDF.
- Provide a reliable way to convert from the sqoop schema to avro schema for all the 14 sqoop data types supported.

Design

- Extend the IDF API

```
/**
 * IDF representing the intermediate format in Avro object
 */
public class AvroIntermediateDataFormat extends IntermediateDataFormat<GenericRecord> {...}
```

- Sqoop schema is mandated, since we need a schema to construct a avro record

```
// convert the sqoop schema to avro schema
 public AvroIntermediateDataFormat(org.apache.sqoop.schema.Schema schema) {
   super.setSchema(schema);
 }
```

- Implement a method to convert csv text to avro GenericRecord
  - private GenericRecord toAvro(String csv) {..}
- Implement a method to convert the object array to avro GenericRecord
  - private GenericRecord toAvro(Object[] data) { ..}
- Conversely, implement a method to lazily construct the csv from avro GenericRecord when invoked
  - private String toCSV(GenericRecord record) { ..}
- implement a method to lazily construct the object arrat from avro GenericRecord when invoked
  - private Object[] toObject(GenericRecord data) {...}
- Implement methods to ser/ deser the avro record into a string - wire format

```
/**
   * {@inheritDoc}
   */
  @Override
  public void write(DataOutput out) throws IOException {
   // todo
  }
  /**
   * {@inheritDoc}
   */
  @Override
  public void read(DataInput in) throws IOException {
    // todo
  }
```

- Mappings from sqoop to avro types.

Column Type	Object Format	Avro Format / Feld Type
NULL value in the field	java null	UNION for any field that is nullable Schema.Type.NULL
ARRAY	java Object[]	Schema.Type.ARRAY
BINARY	java byte[]	Schema.Type.BYTES
BIT	java boolean	Schema.Type.BOOLEAN
DATE	org.joda.time.LocalDate	Schema.Type.LONG
DATE_TIME	org.joda.time. DateTime or org.joda.time. LocalDateTime (depends on timezone attribute)	Schema.Type.LONG
DECIMAL	java BigDecimal	Schema.Type.FIXED ???
ENUM	java String	Schema.Type.ENUM

FIXED_POINT	java Integer
	or
	java Long
	(depends on
	byteSize attribute)
		if (((org.apache.sqoop.schema.type.FixedPoint) column).getByteSize() <= Integer.SIZE) {
		return Schema.Type.INT;
		} else {
		return Schema.Type.LONG;
		}

FLOATING_POINT	java Double
	or
	java Float
	(depends on
	byteSize attribute)
		if (((org.apache.sqoop.schema.type.FloatingPoint) column).getByteSize() <= Float.SIZE) {
		return Schema.Type.FLOAT;
		} else {
		return Schema.Type.DOUBLE;
		}

MAP	java.util.Map<Object, Object>	Schema.Type.MAP
SET	java Object[]	Schema.Type.ARRAY
TEXT	java String	Schema.Type.STRING
TIME	org.joda.time.LocalTime (No Timezone)	Schema.Type.LONG
UNKNOWN	same as java byte[]	Schema.Type.BYTES

External Jar Dependencies added?

- Yes, avro 1.7 dependency added to the connector-sdk package

Testing

The unit tests should cover the following use cases for the 14 ColumnTypes supported by sqoop, including the null representation.

```
// convert from avro to other formats
setDataGetCSV
setDataGetObjectArray
setDataGetData

// convert from csv to other forms
setCSVGetData
setCSVGetObjectArray
setCSVGetCSV

// convert from object array to other formats
setObjectArrayGetData
setObjectArrayGetCSV
setObjectArrayGetObjectArray
```

Open Questions

- Avro 1.7 that we use does not yet support the date/dateTime/Time as a first class primitive types. Until 1.8 represent date as long
- How to represent Decimal? Should we use the "FIXED" type in avro
- Handling nulls via the union type, is this ok?