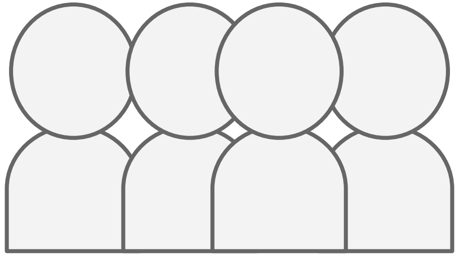
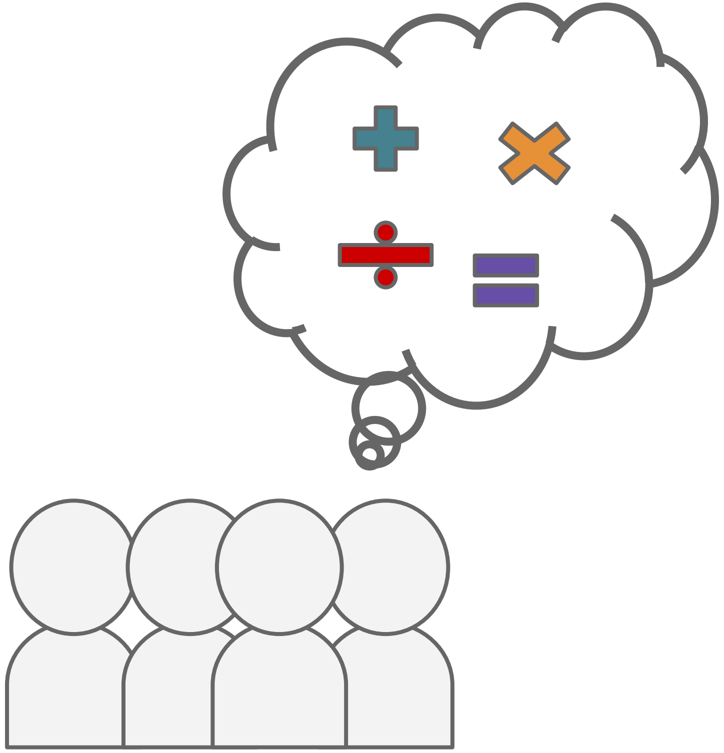
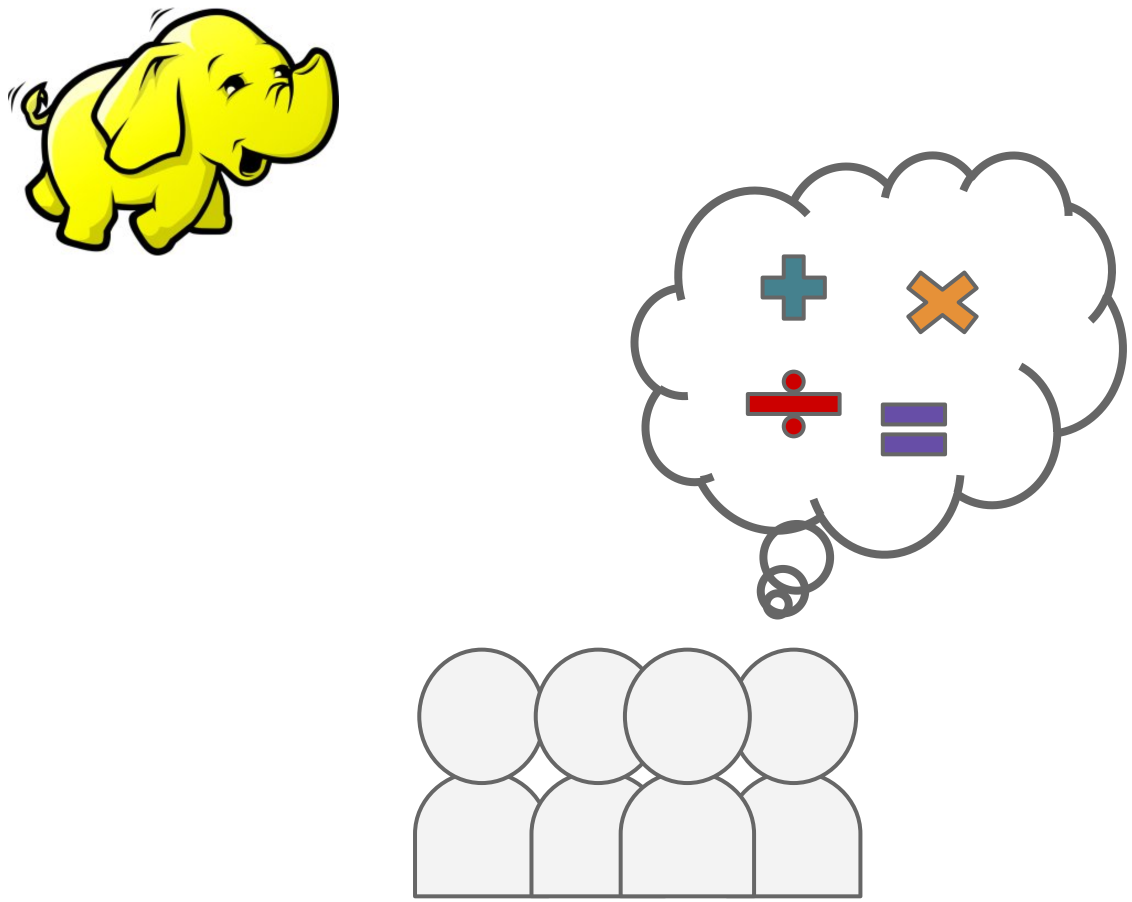
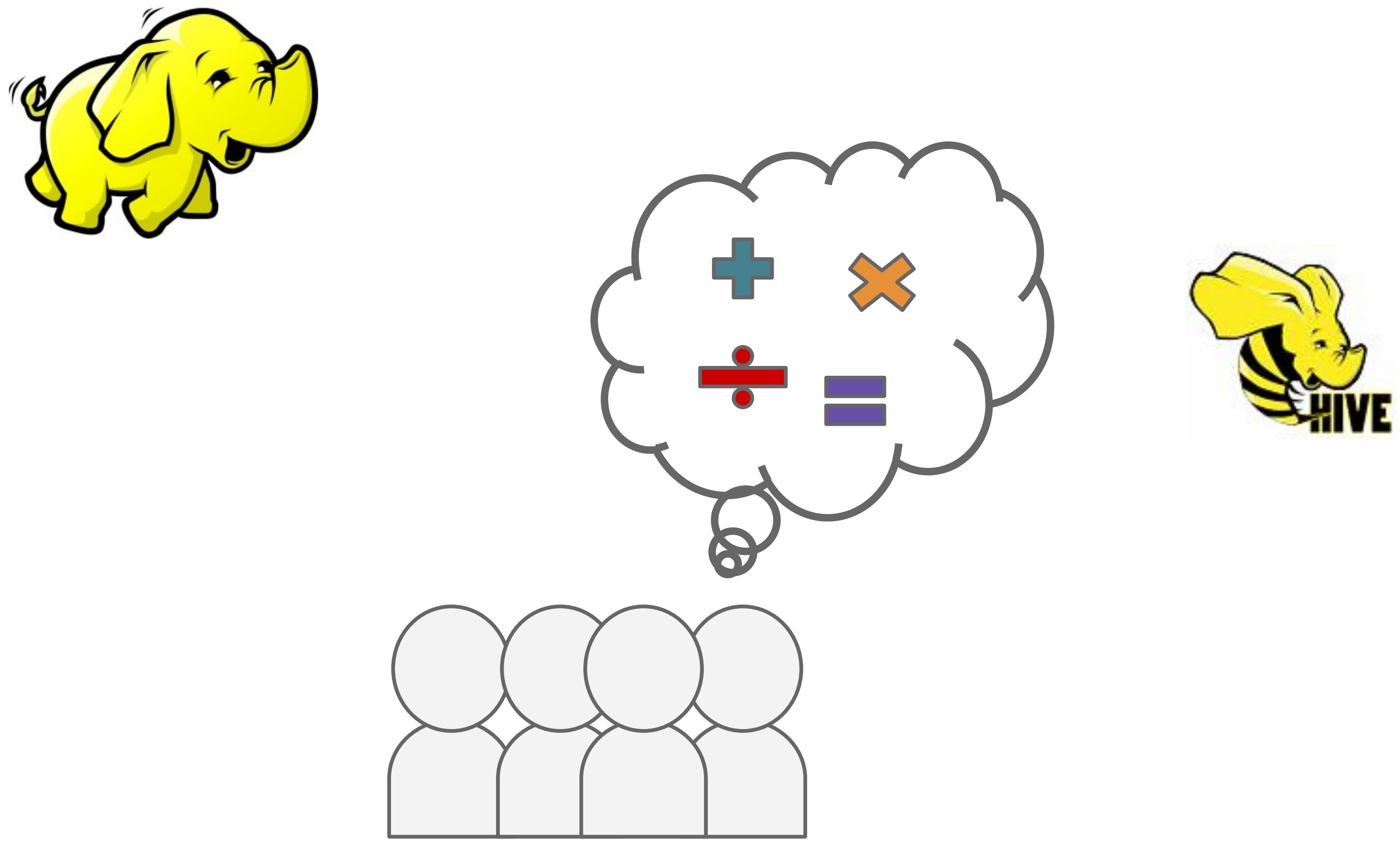
Simplifying Big Data with Apache Crunch

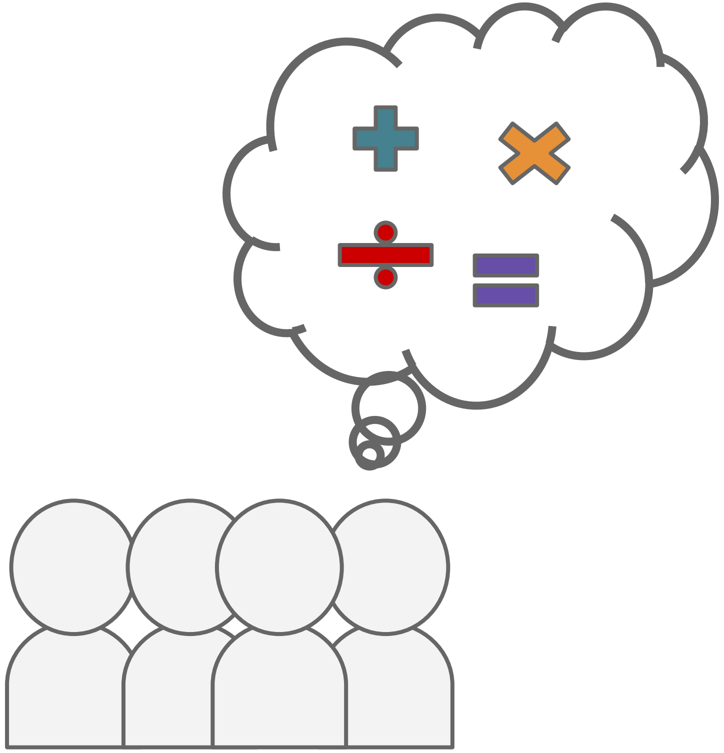
Micah Whitacre
@mkwhit

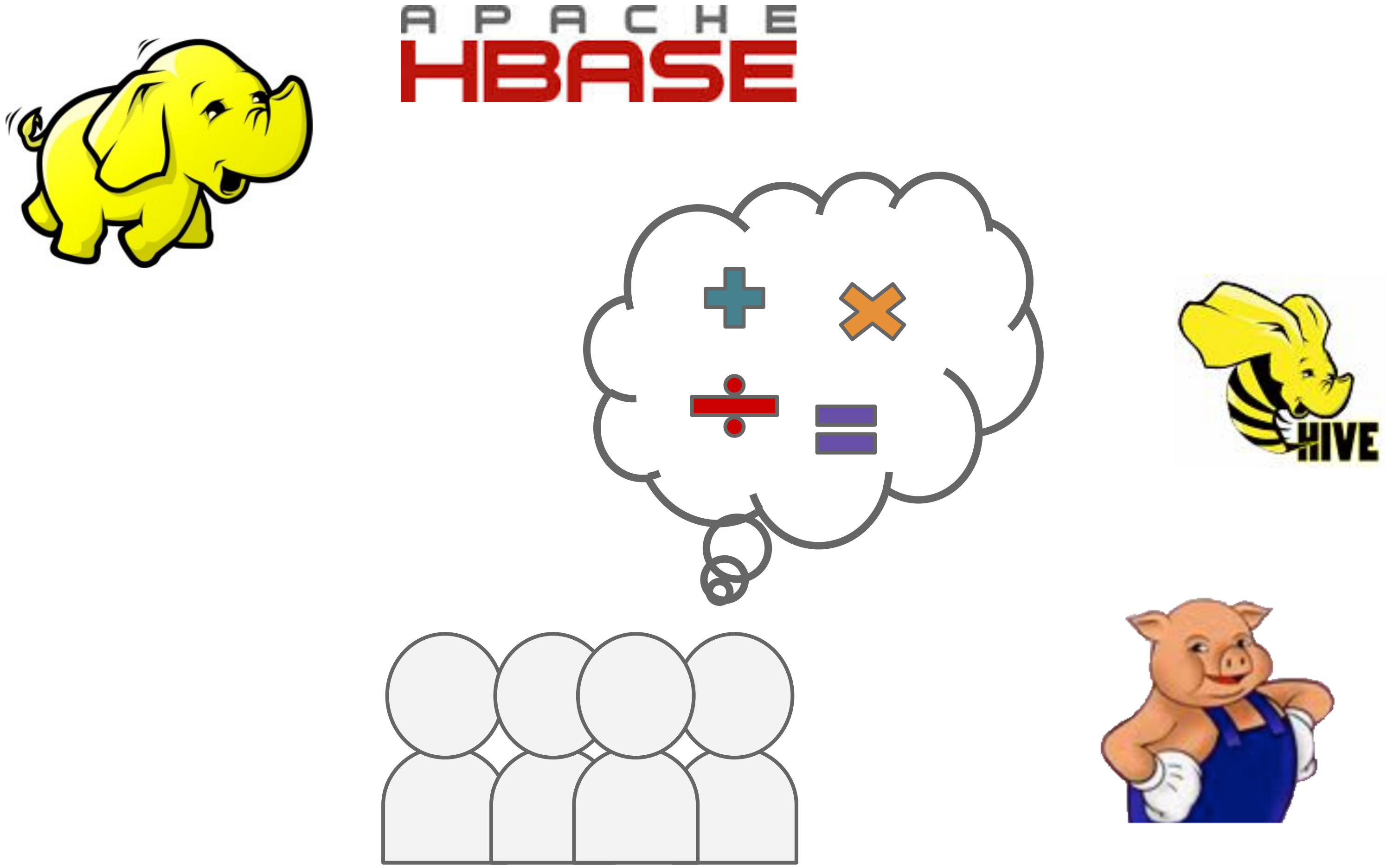






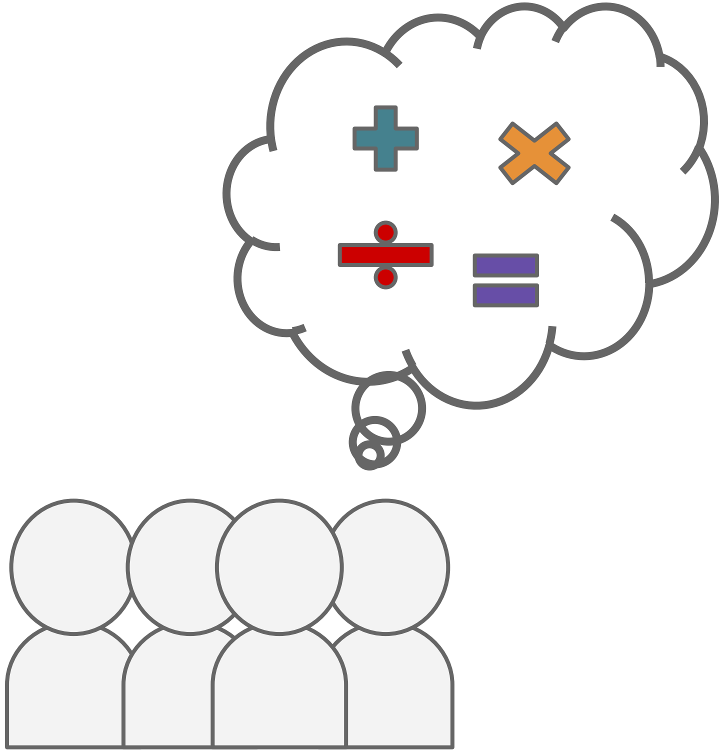

HIVE




APACHE
HBASE
HIVE

APACHE HBASE

OOZIE



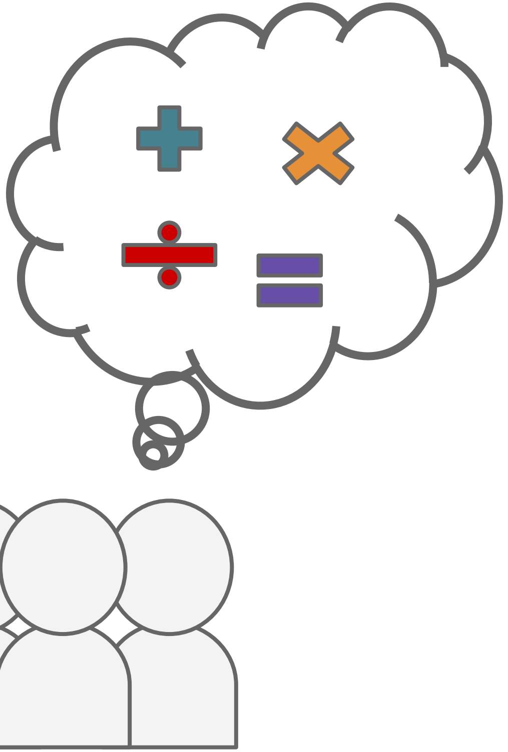
HIVE

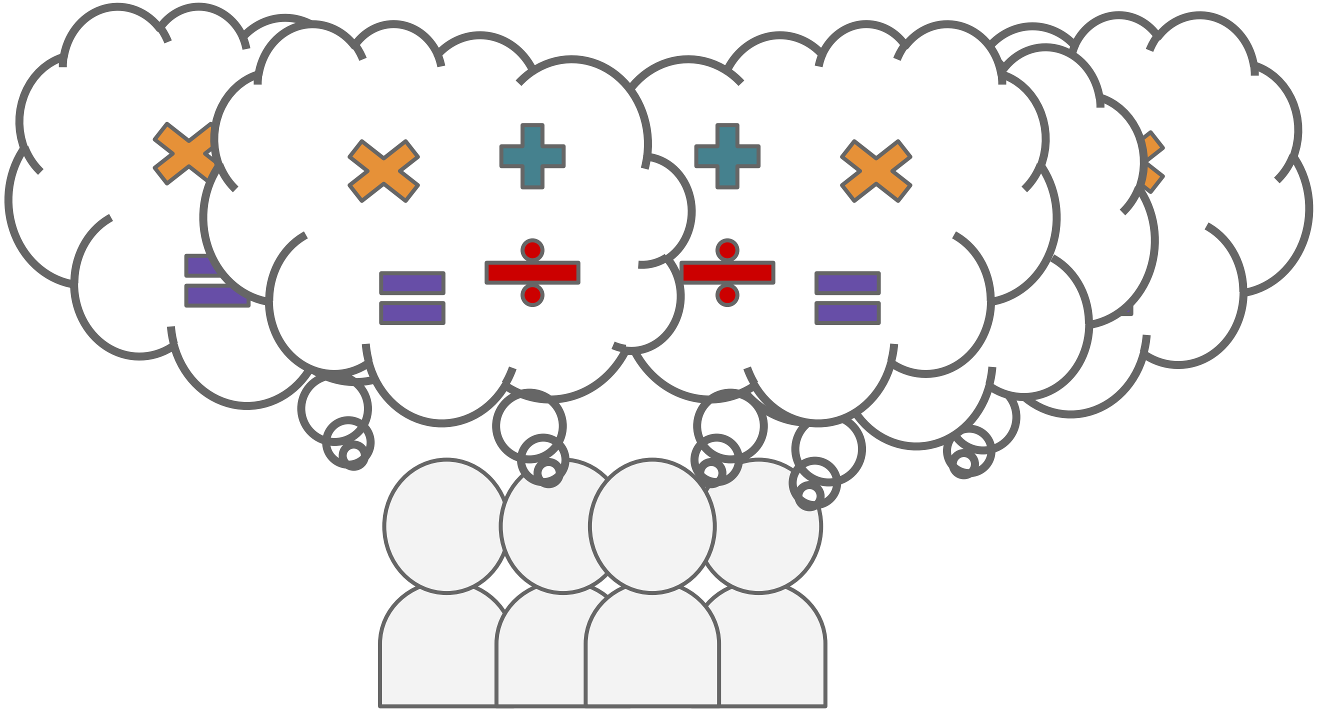
APACHE HBASE

OOZIE

sqoop

HIVE





Cerner™

Semantic Chart Search

Medical Alerting System

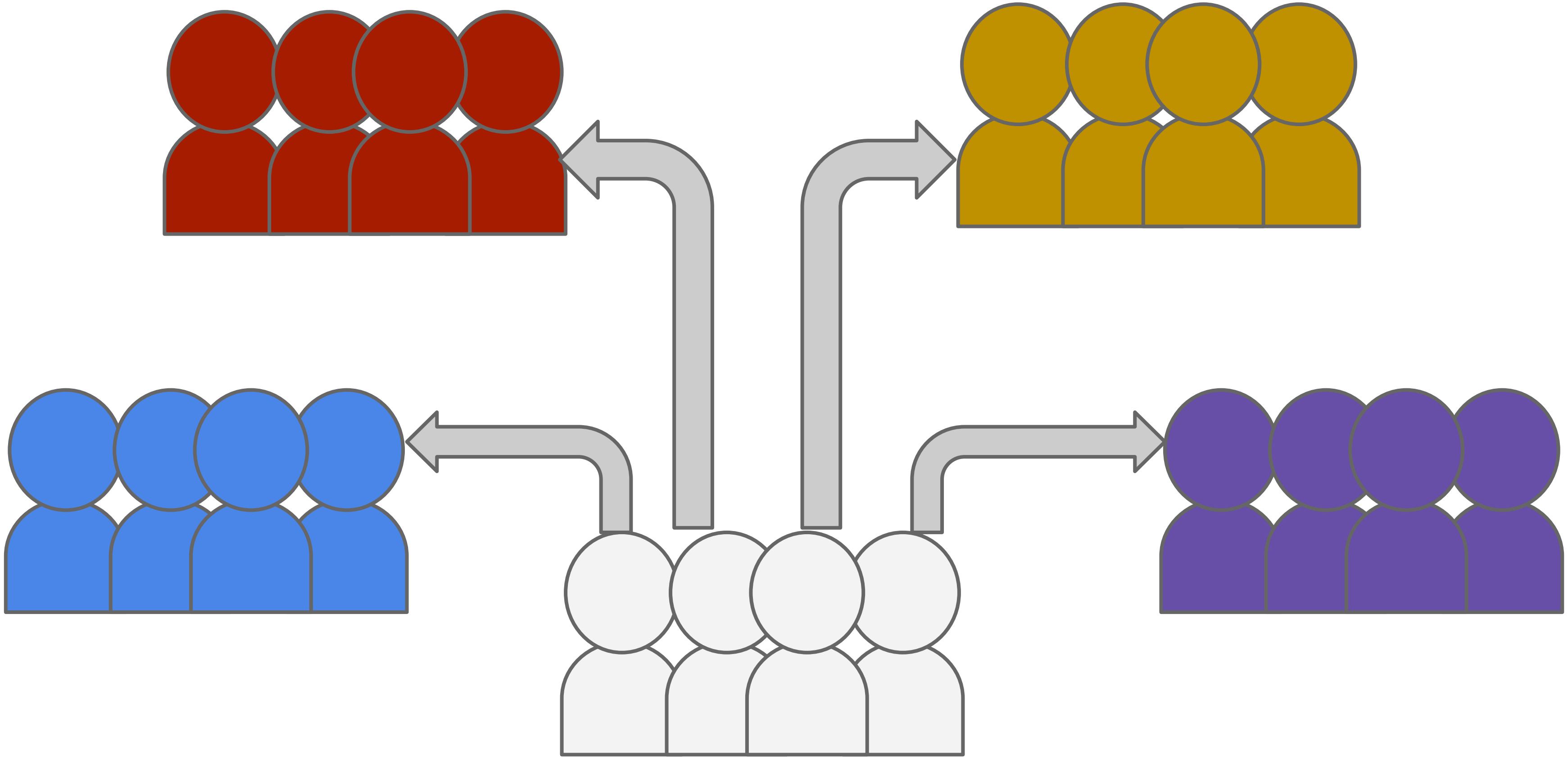
Cloud Based EMR

Population Health Management

Problem moves from scaling *architecture*...

Problem moves from not only scaling *architecture*...

To how to scale the *knowledge*



Battling the 3 V’s

Daily, weekly, monthly ***uploads***

Battling the 3 V’s

Daily, weekly, monthly *uploads*

60+ *different* data formats

Battling the 3 V's

Daily, weekly, monthly *uploads*

60+ *different* data formats

Battling the 3 V’s

Constant *streams* for near real time

Daily, weekly, monthly *uploads*

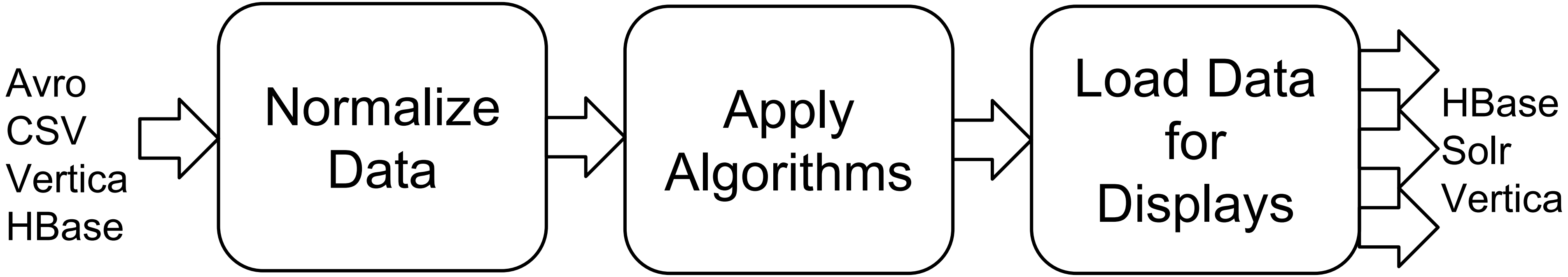
60+ *different* data formats

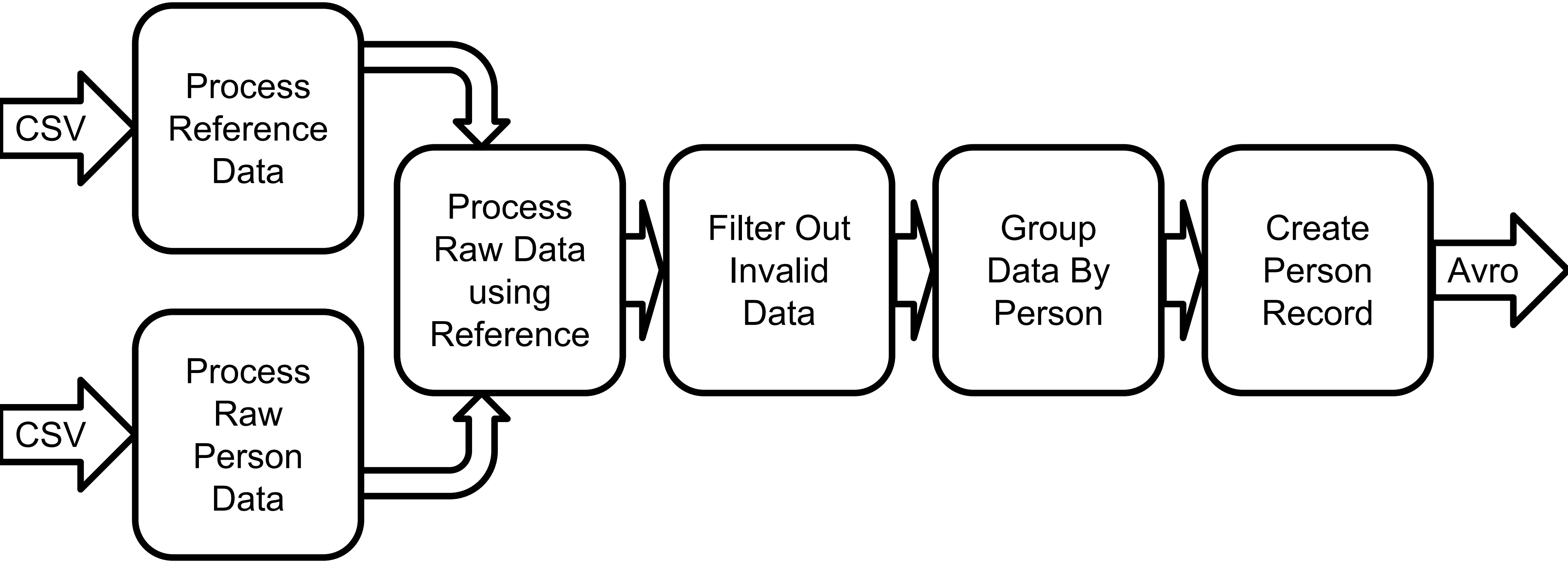
Battling the 3 V’s

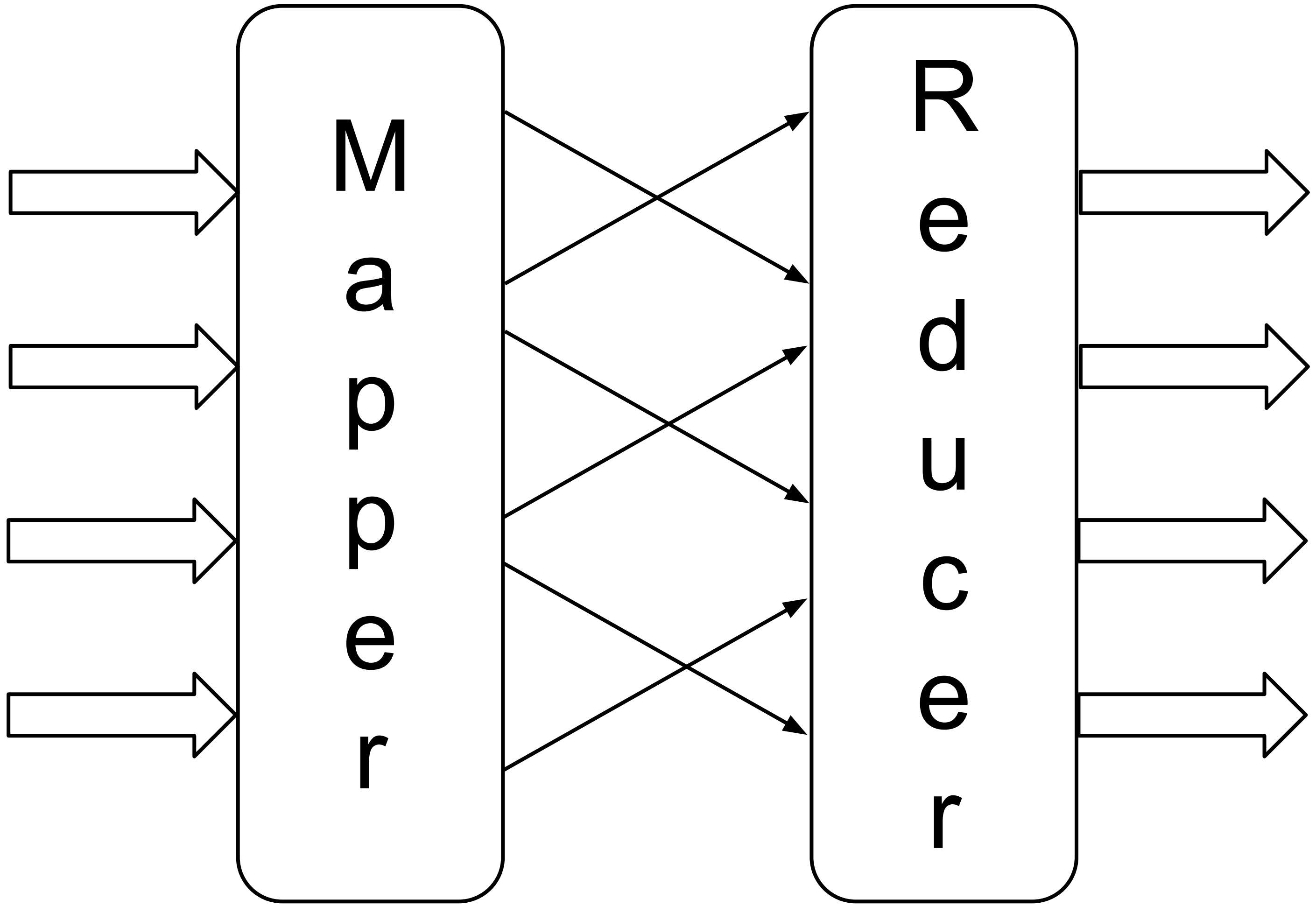
Constant *streams* for near real time

2+ *TB* of streaming data *daily*

Population Health




CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro


Mapper
Reducer

Struggle to fit into single MapReduce job

Struggle to fit into single MapReduce job

Integration done through persistence

Struggle to fit into single MapReduce job

Integration done through persistence

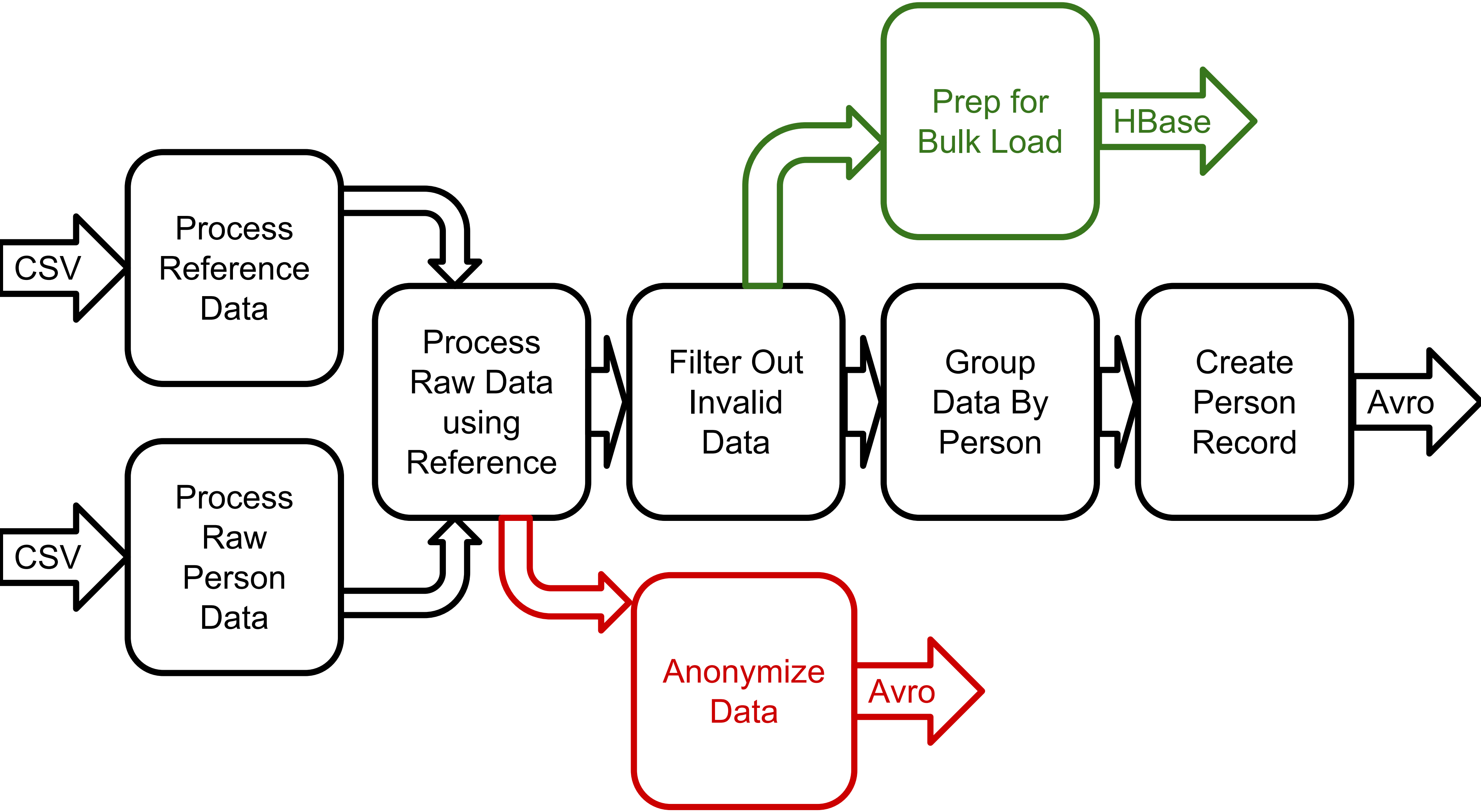
Custom impls of common patterns

Struggle to fit into single MapReduce job

Integration done through persistence

Custom impls of common patterns

Evolving Requirements


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Prep for Bulk Load
HBase
Group Data By Person
Create Person Record
Avro
Anonymize Data
Avro

Easy integration between teams

Focus on processing steps

Shallow learning curve

Ability to tune for performance

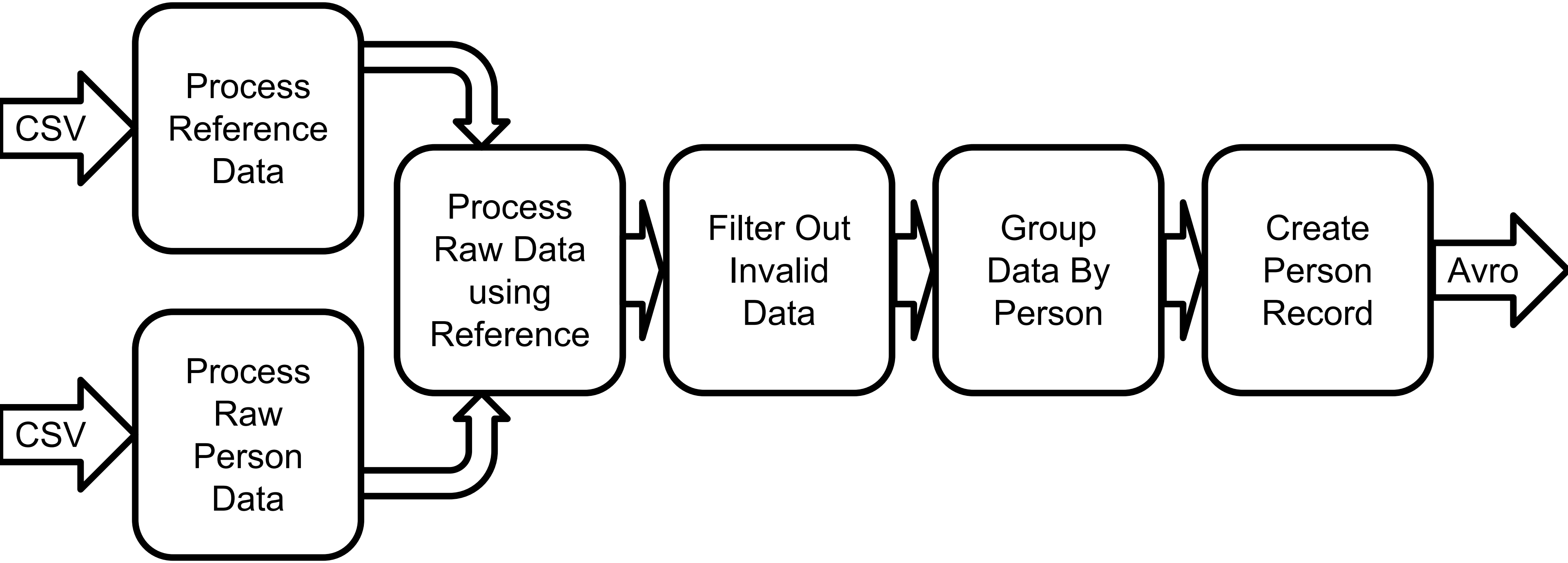
Apache Crunch

Compose processing into pipelines

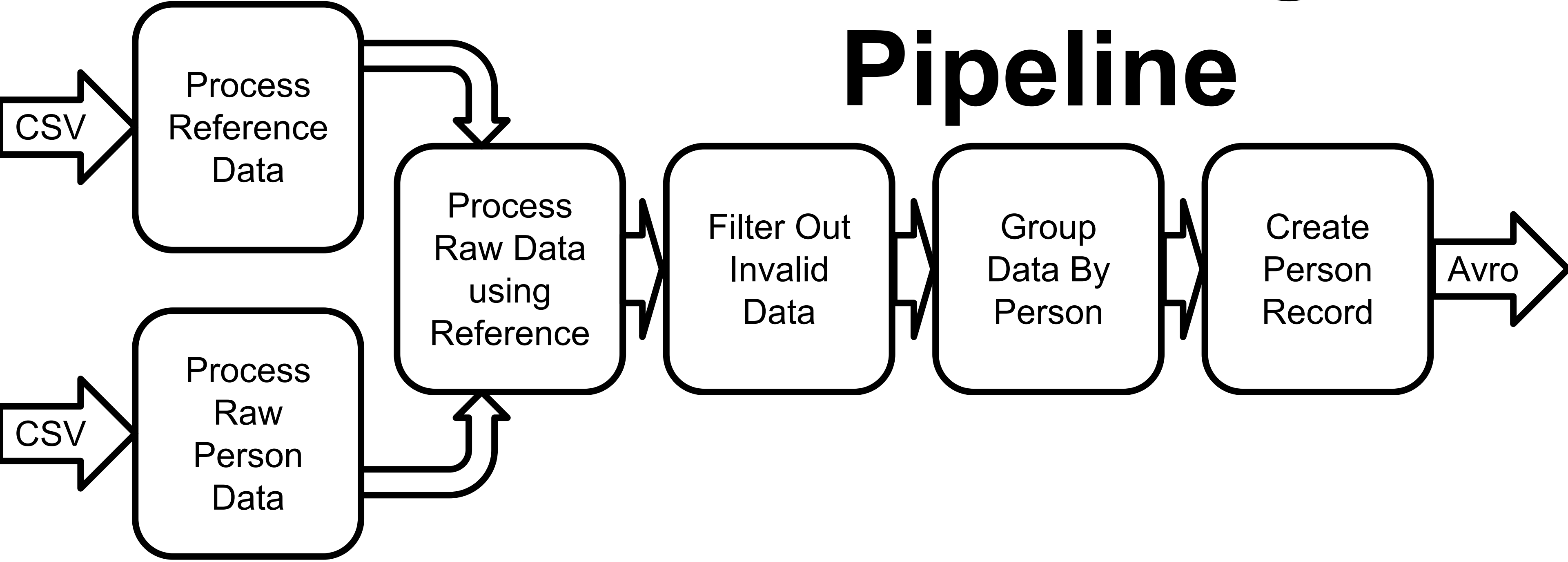
Open Source FlumeJava impl

Transformation through fns (not job)

Utilizes POJOs (hides serialization)


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro

Processing Pipeline



Pipeline

Programmatic description of DAG

Supports lazy execution

Implementations indicate runtime

MapReduce, Spark, Memory

```
Pipeline pipeline =
    new MRPipeline(Driver.class, conf);

Pipeline pipeline =
   MemPipeline.getIntance();

Pipeline pipeline =
    new SparkPipeline(sparkContext, “app”);
```

Source

Reads various inputs

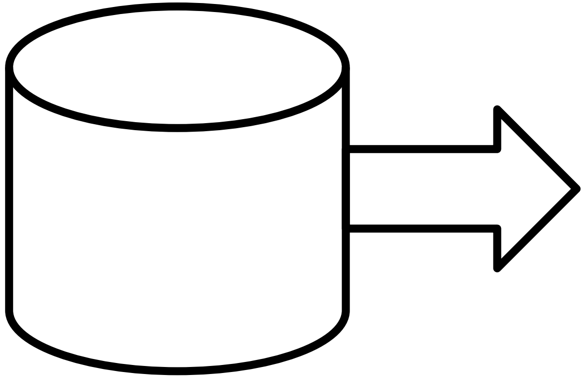
At least one required per pipeline

Creates initial collections for processing

Custom implementations

Source

Sequence Files
Avro
Parquet
HBase
JDBC
HFiles
Text
CSV



Strings
AvroRecords
Results
POJOs
Protobufs
Thrift
Writables

```
pipeline.read(
        From.textFile(path));
```

```
pipeline.read(
  new TextFileSource(path,ptype));
```

```
PType<String> ptype = …;
pipeline.read(
  new TextFileSource(path,ptype));
```

PType

Hides serialization

Exposes data in native Java forms

Supports composing complex types

Avro, Thrift, and Protocol Buffers

Multiple Serialization Types

Serialization Type = PTypeFamily

Avro & Writable available

Can’t mix families in single type

Can easily convert between families

```
PType<Integer> intTypes =
  Writables.ints();
PType<String> stringType =
  Avros.strings();
PType<Person> personType =
  Avros.records(Person.class);
```

```
PType<Pair<String, Person>> pairType =
  Avros.pairs(stringType, personType);
```

```
PTableType<String, Person> tableType =
 Avros.tableOf(stringType,personType);
```

```
PType<String> ptype = …;
PCollection<String> strings = pipeline.read(
   new TextFileSource(path, ptype));
```

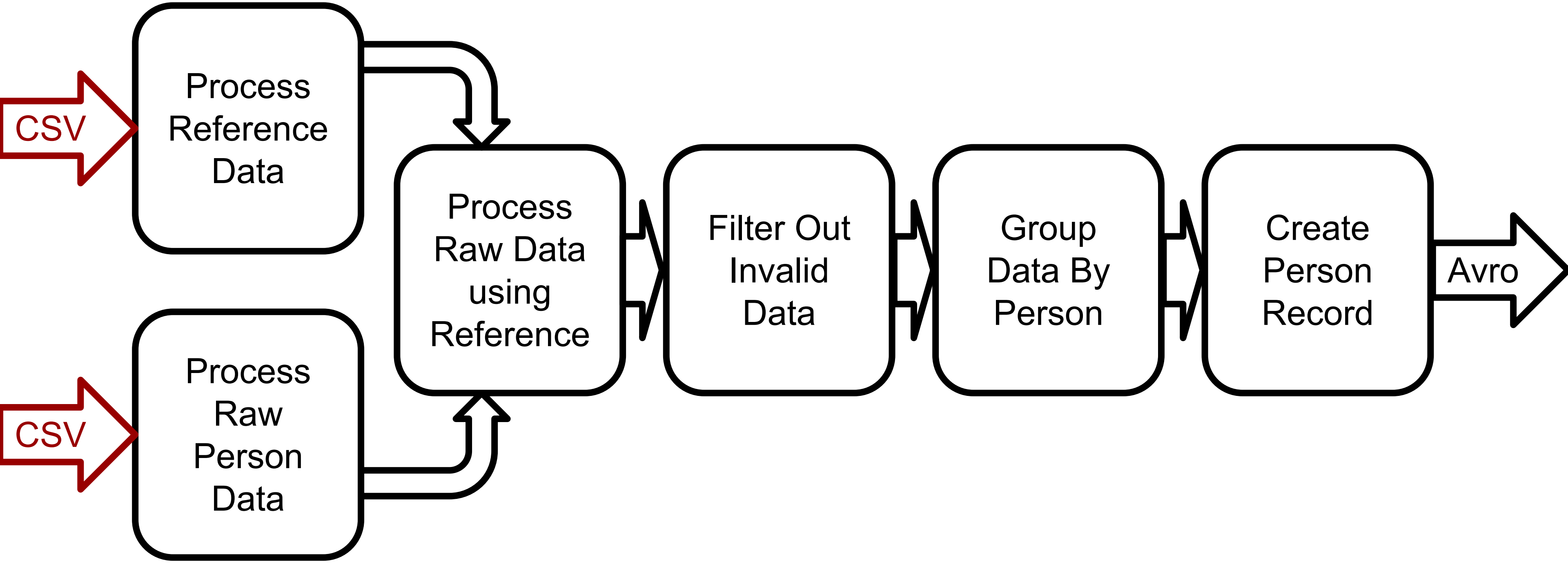
PCollection

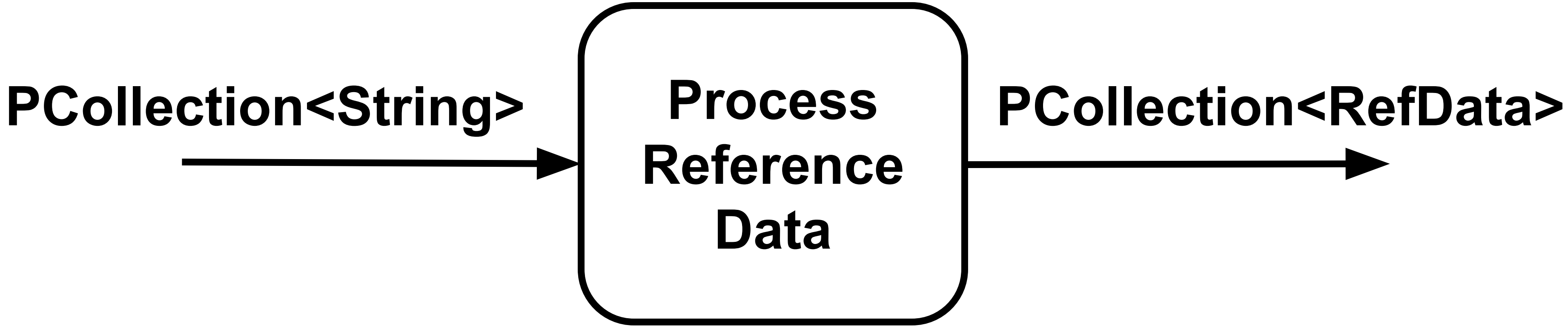
Immutable

Unsorted

Not created only read or transformed

Represents potential data


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro


PCollection<String>
Process Reference Data
PCollection<RefData>

DoFn

Simple API to implement

Transforms PCollection between forms

Location for custom logic

Processes one element at a time

For each item emits 0:M items

MapFn - emits 1:1

FilterFn - returns boolean

DoFn API

```
class ExampleDoFn extends
DoFn<String, RefData>{
    ...
}
```

Type of Data In

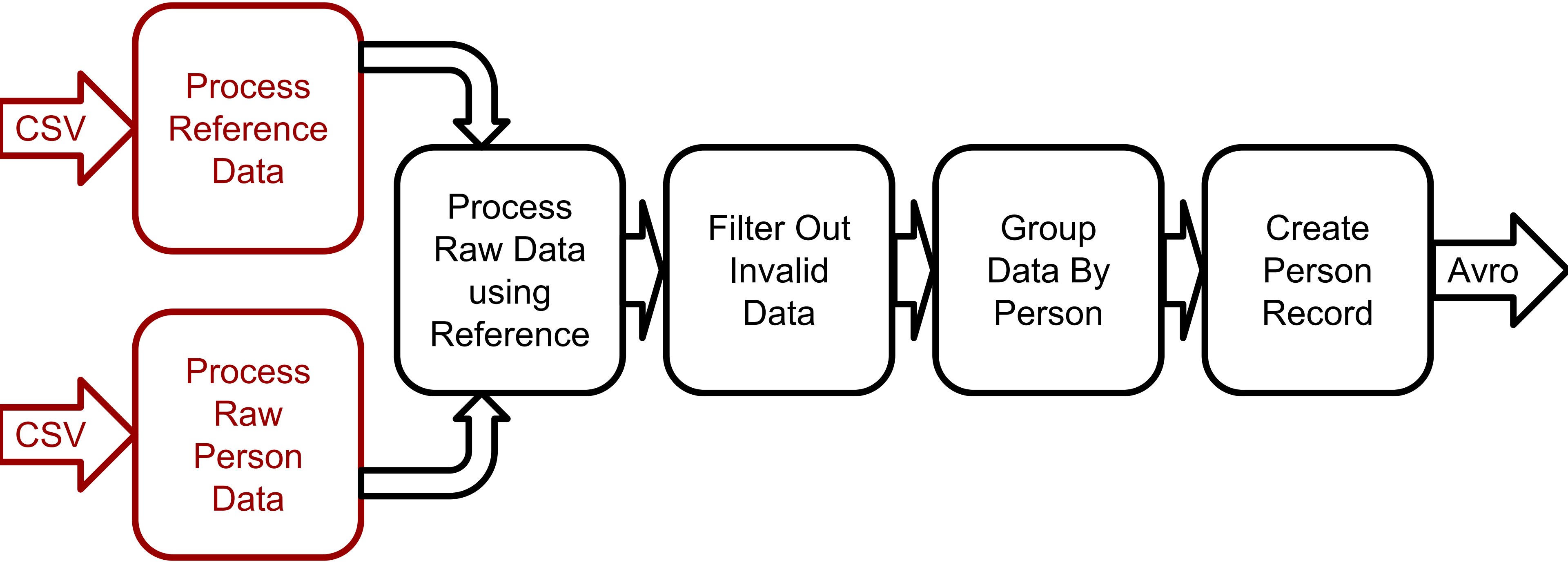
Type of Data Out

Type of Data In Type of Data Out

```
public void process
 (String s,
    Emitter<RefData> emitter) {
    RefData data = …;
    emitter.emit(data);
}
```

```
PCollection<String> refStrings
PCollection<RefData> refs =
  refStrings.parallelDo(fn,
  Avros.records(RefData.class));
```

```
PCollection<String> dataStrs...
PCollection<RefData> refs =
  dataStrs.parallelDo(diffFn,
  Avros.records(Data.class));
```


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro

Hmm now I need to join...

But they don’t have a common key?

We need a PTable

PTable<K, V>

Immutable & Unsorted

Multimap of Keys and Values

Variation PCollection<Pair<K, V>>

Joins, Cogroups, Group By Key

```
class ExampleDoFn extends
DoFn<String, RefData>{
   ...
}
```

```
class ExampleDoFn extends
DoFn<String,
  Pair<String, RefData>>{
   ...
}
```

```
PCollection<String> refStrings
PTable<String, RefData> refs =
 refStrings.parallelDo(fn,
 Avros.tableOf(Avros.strings(),
 Avros.records(RefData.class));
```

```
PTable<String, RefData> refs…;
PTable<String, Data> data…;
```

```
data.join(refs);
```

(inner join)

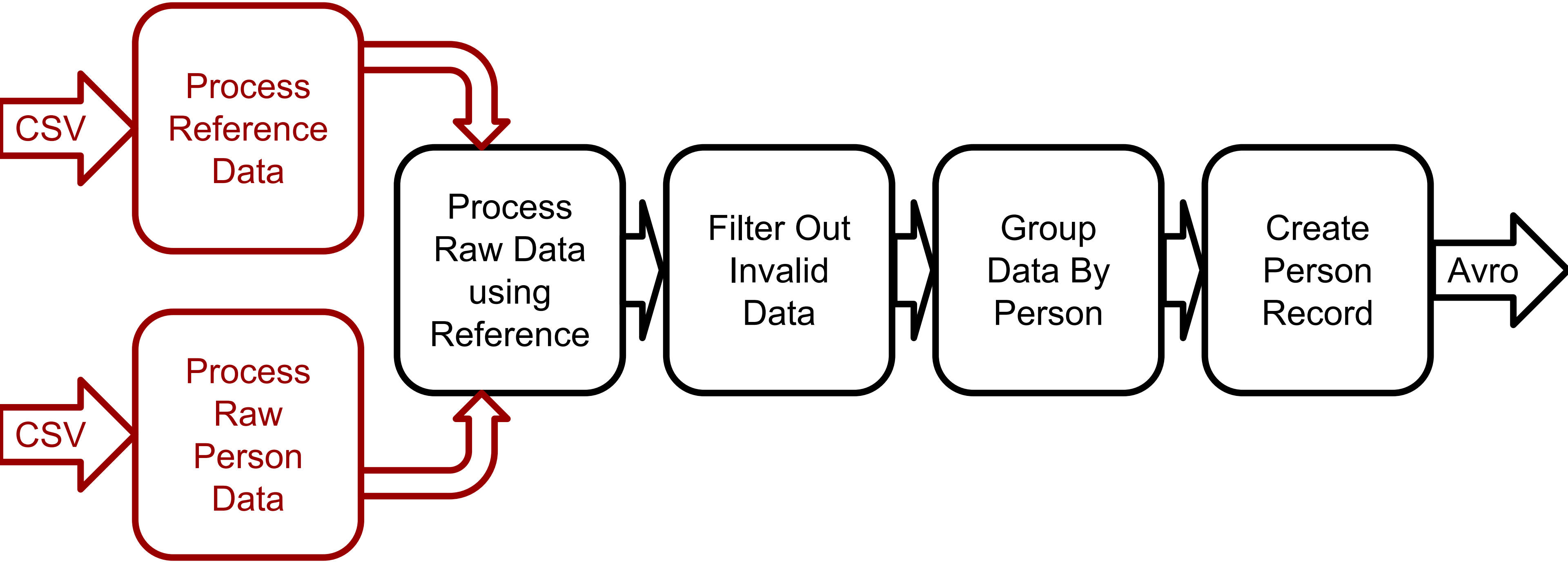
```
PTable<String,
  Pair<Data, RefData>>
  joinedData = data.join(refs);
```

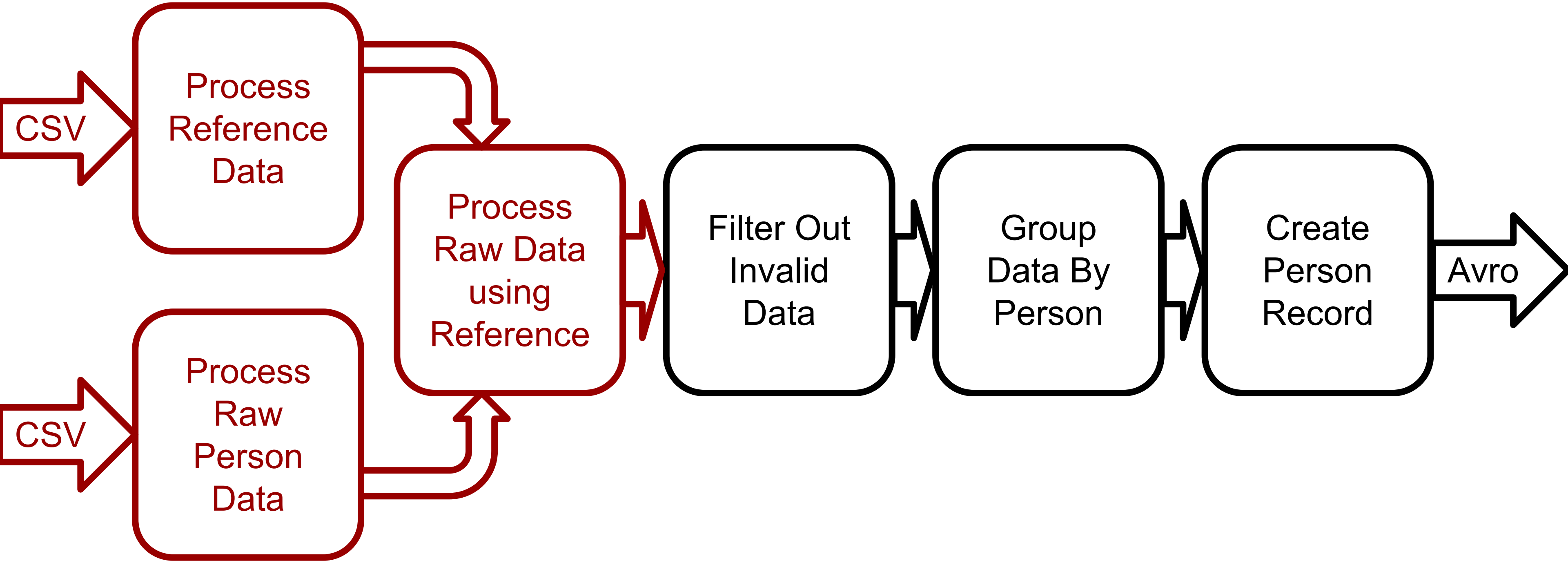
Joins

right, left, inner, outer

Eliminates custom impls

Mapside, BloomFilter, Sharded


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro

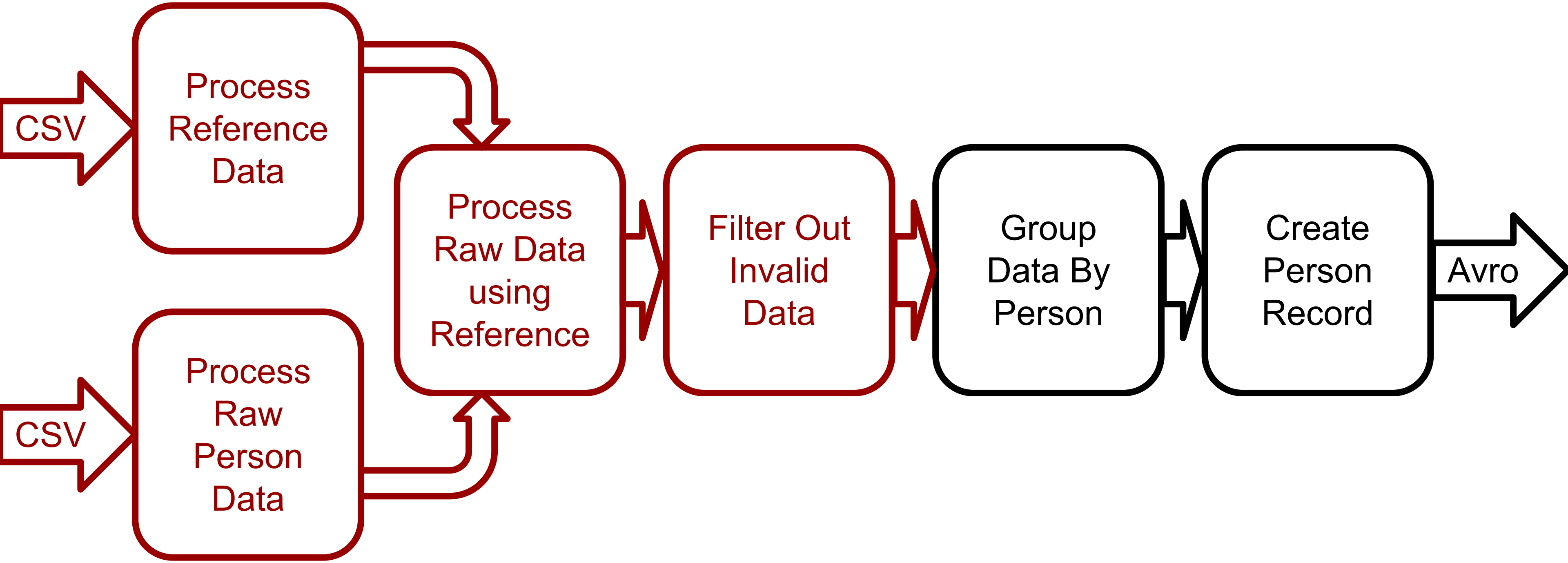
FilterFn API

```
class MyFilterFn extends
FilterFn<...>{
   ...
}
```

Type of Data In

```
public boolean accept
(... value){
   return value > 3;
}
```

```
PCollection<Model> values = …;
PCollection<Model> filtered =
 values.filter(new
   MyFilterFn());
```


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro

Keyed By PersonId

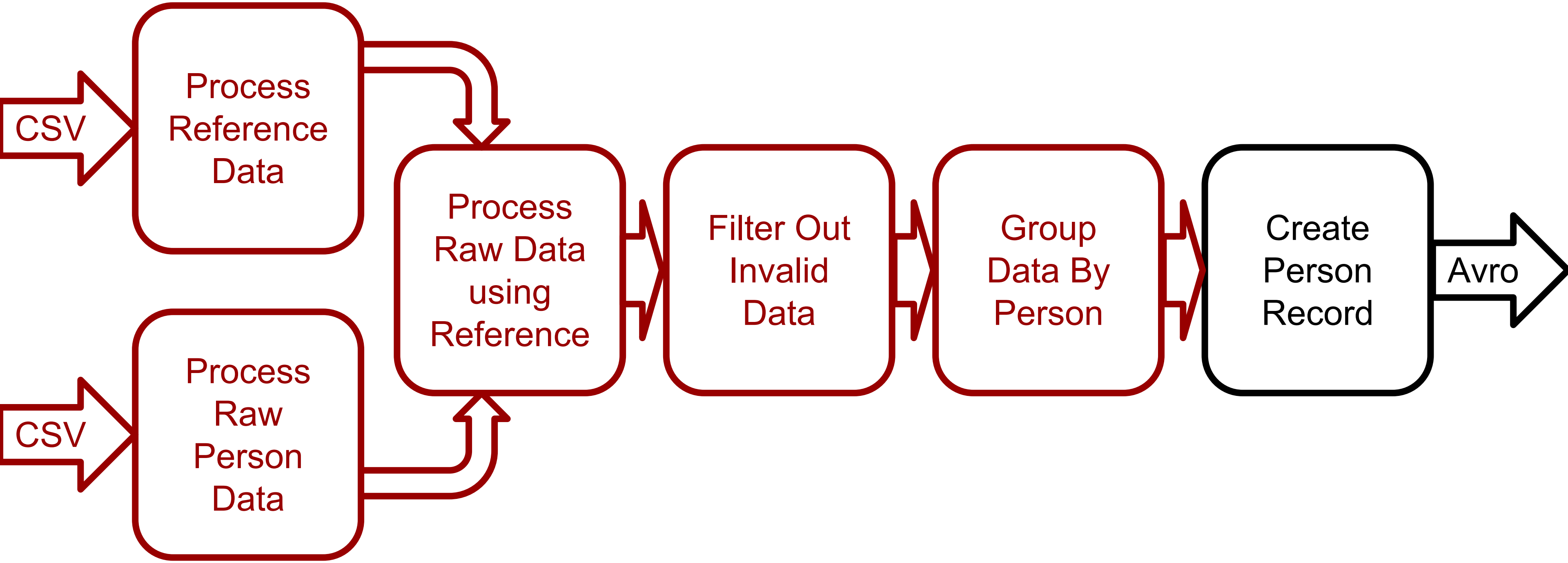
```
PTable<String,Model> models = …;
```

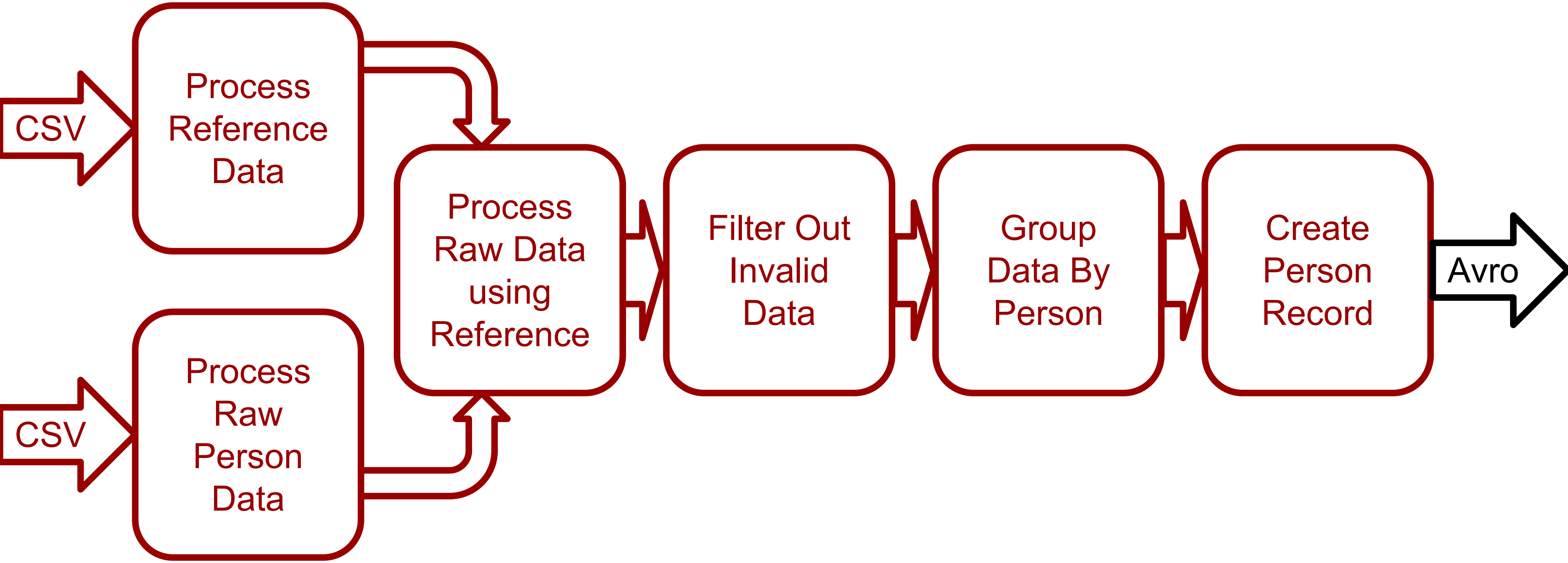
```
PTable<String,Model> models = …;
PGroupedTable<String, Model>
  groupedModels =
    models.groupByKey();
```

PGroupedTable<K, V>

Immutable & Sorted

PCollection<Pair<K, Iterable<V>>>


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro

```
PCollection<Person> persons = …;
```

```
PCollection<Person> persons = …;
pipeline.write(persons,
  To.avroFile(path));
```

```
PCollection<Person> persons = …;
pipeline.write(persons,
  new AvroFileTarget(path));
```

Target

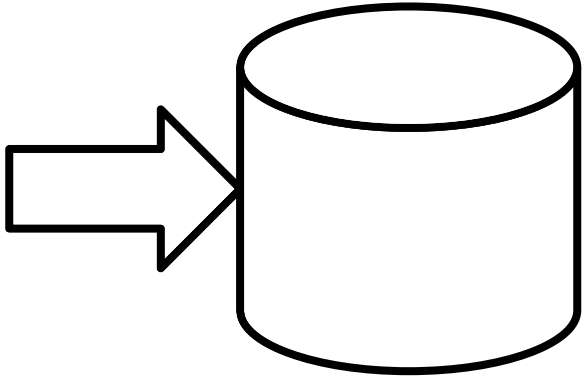
Persists PCollection

At least one required per pipeline

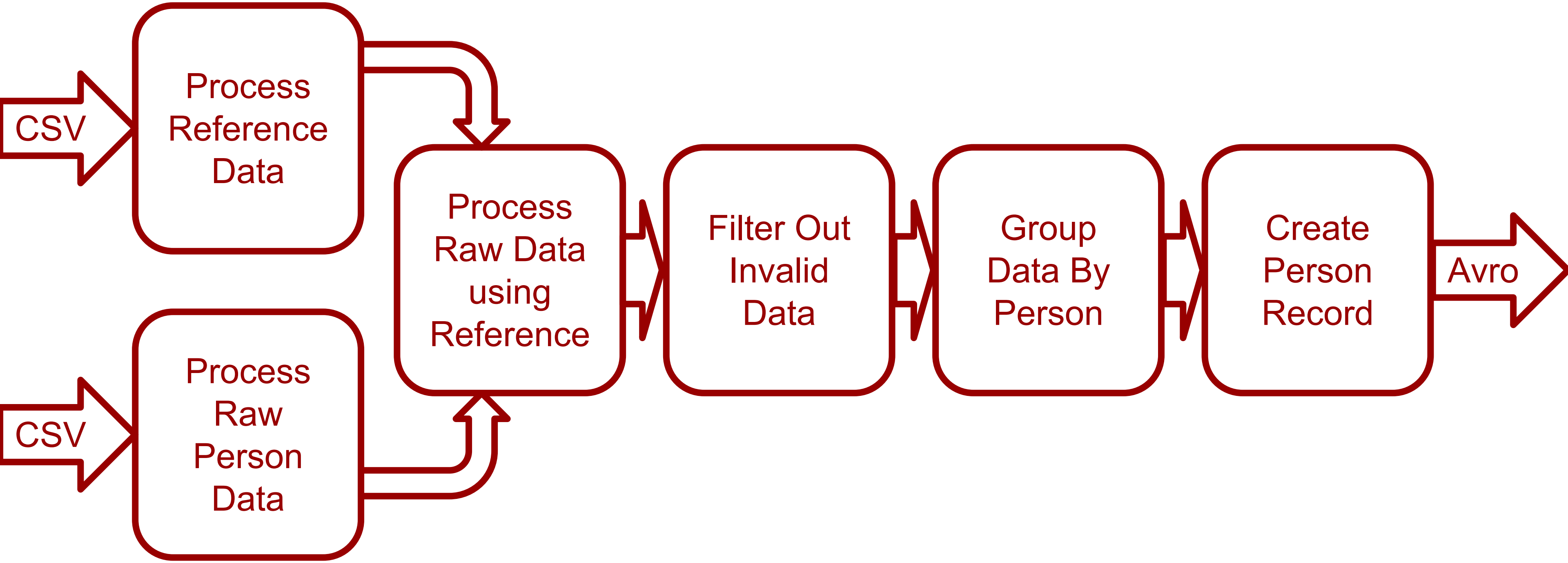
Custom implementations

Target

Strings
AvroRecords
Results
POJOs
Protobufs
Thrift
Writables

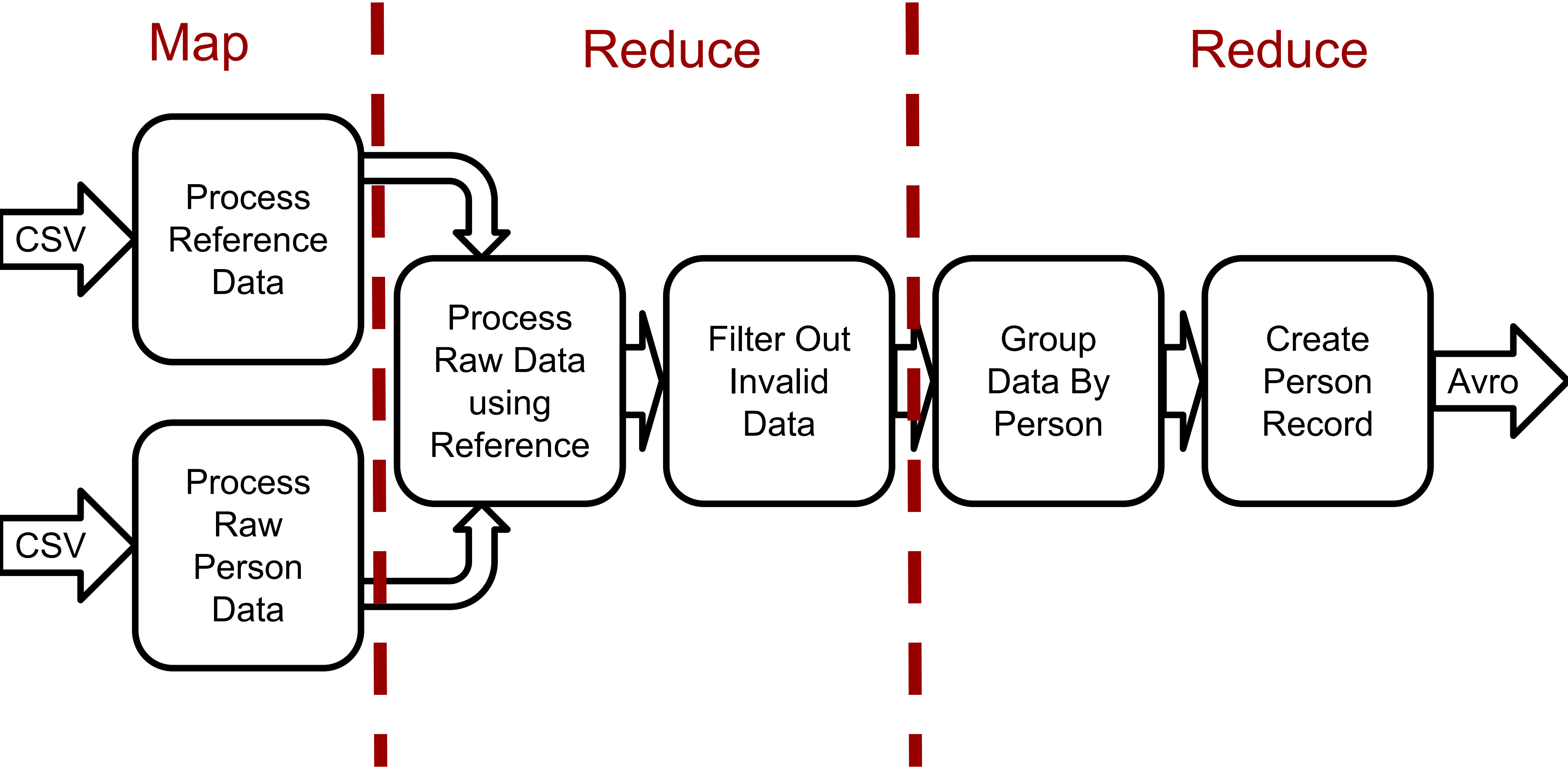


Sequence Files
Avro
Parquet
HBase
JDBC
HFiles
Text
CSV


CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro

Execution

```
Pipeline pipeline = …;
...
pipeline.write(...);
PipelineResult result =
  pipeline.done();
```


Map
Reduce
Reduce
CSV
Process Reference Data
CSV
Process Raw Person Data
Process Raw Data using Reference
Filter Out Invalid Data
Group Data By Person
Create Person Record
Avro

Tuning

Tweak pipeline for performance

GroupingOptions/ParallelDoOptions

Scale factors

Functionality first

Focus on the transformations

Smaller learning curve

Less fragility

Iterate with confidence

Integration through PCollections

Extend pipeline for new features

Links

http://crunch.apache.org/

http://dl.acm.org/citation.cfm?id=1806596.1806638

http://www.quora.com/Apache-Hadoop/What-are-the-differences-between-Crunch-and-Cascading