Example 18:

Filtering rows based on a condition

Filter

Filter

We've seen how to use Scan to

1. Retrieve all rows, columns
2. Retrieve specific columns for all rows
3. Retrieve specific columns for rows in a specified range

Filter

1. Retrieve all rows, columns
2. Retrieve specific columns for all rows
3. Retrieve specific columns for rows in a specified range

You can add a filter to the Scan and customize it further

Filter

Filters allow you to control what data is retrieved by Scan

Filter

HBase provides a large number of built in filters

Specific set of row ids

Specific set of columns/ column families

Specific value for a column

Timestamps

and more..

Filter

Let's create a filter for a specific row id

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();


    }
}
```

Filter

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();


    }
}
```

Scan objects have a **setFilter** method where we can specify a filter

Filter

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();


    }
}
```

All filters are subclasses of the Filter abstract class

Filter

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();


    }
}
```

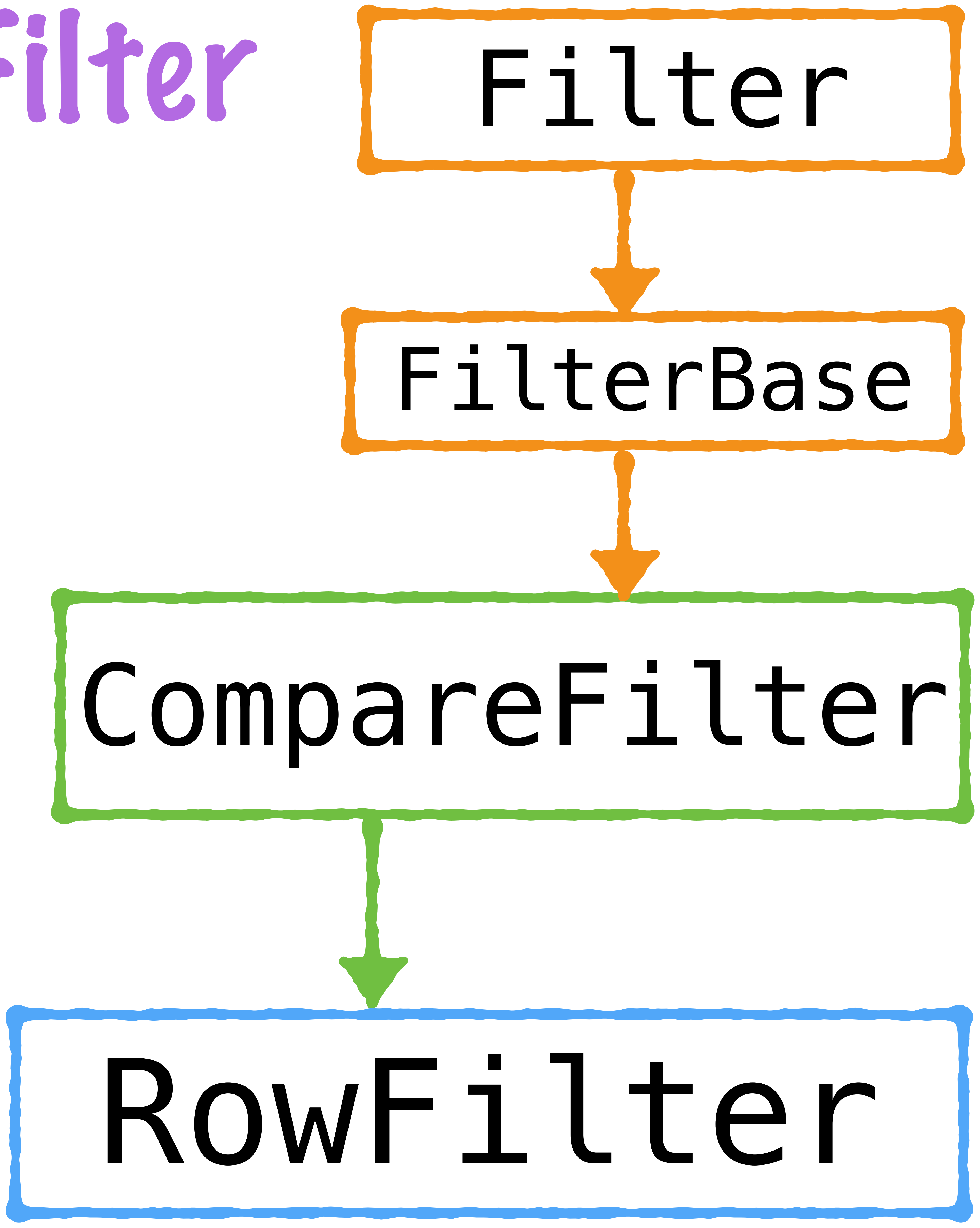
RowFilter is one of the built-in filters provided by HBase

Filter

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();


    }
}
```

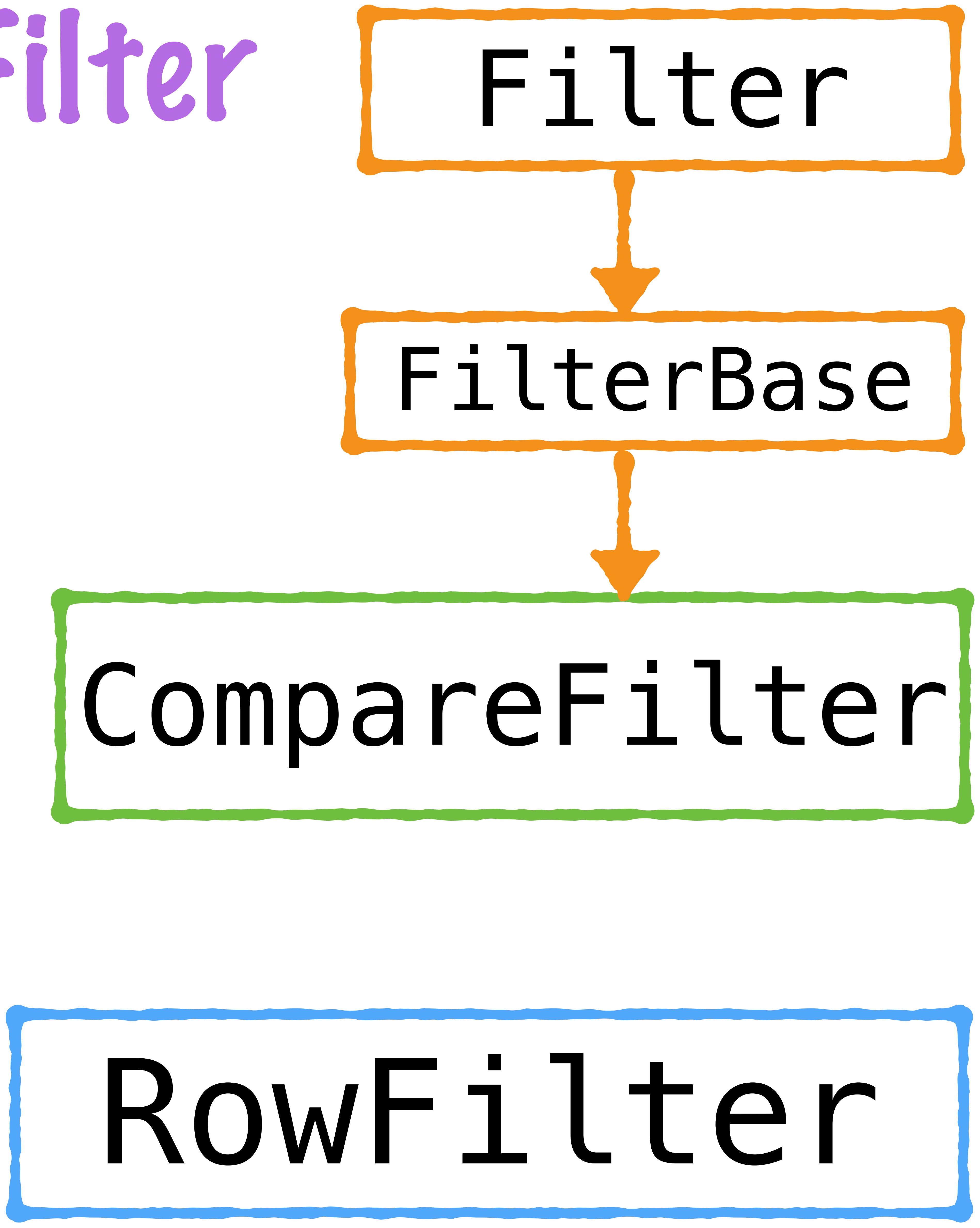
RowFilter descends from CompareFilter

Filter



CompareFilters are used to check if the data matches a specific condition

Filter



For instance RowFilters, will check if the row id matches the specified condition

Filter

`CompareFilter`

CompareFilters need

1. An operator
2. A Comparator object

Filter

CompareFilter

1. An operator

The **CompareFilter** class provides a few different operators

```
LESS
LESS_OR_EQUAL
EQUAL
NOT_EQUAL
GREATER_OR_EQUAL
GREATER
```

Filter

`CompareFilter`

1. An operator

2. A Comparator object

These are objects which will compare the table data against a specified value

Filter

`RowFilter`

1. An operator
2. A Comparator object

Row Filter will take the operator and Comparator object and use them to filter the row ids

Filter

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result [illegible] : userScanResult) {
            [illegible]
        }

        userScanResult.close();

    }
}
```

1. An operator

2. A Comparator object

Filter

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();


    }
}
```

2. A Comparator object

Filter

2. A Comparator object

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();

    }
}
```

BinaryComparator is used for comparing byte arrays

Filter

2. A Comparator object

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();


    }
}
```

This filter will check if the row id matches the string "1"

Filter

2. A Comparator object

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();

    }
}
```

If this operator had been **LESS**, the comparison would check **if the byte array representing the row id is less than the byte array for "1"**

Filter

2. A Comparator object

```
public class rowFilter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

Filter filter = new RowFilter(CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("1")));

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {
            printValues.printAllValues(res);
        }

        userScanResult.close();

    }
}
```

To compare Strings you can also use other comparators like RegexStringComparator and SubstringComparator

Example 19:
Filtering rows based on the value in a column

SingleColumnValueFilter

SingleColumnValueFilter

Let's see how to filter based on the value for a specified column

This is equivalent to using a where clause in SQL for a single column

```
… where for_user="Daniel"
```

SingleColumnValueFilter … where for_user="Daniel"

```java
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {

            printValues.printAllValues(res);
        }

        userScanResult.close();
    }
}
```

SingleColumnValueFilter … where for_user="Daniel"

```java
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                                Bytes.toBytes("attributes"),
                                Bytes.toBytes("for_user"),
                                CompareFilter.CompareOp.EQUAL,
                                new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanRes      {

            printValues.printAllValu  (res)
        }

        userScanResult.close();
    }
}
```

SingleColumnValueFilter requires 4 parameters

SingleColumnValueFilter

… where for_user="Daniel"

```java
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                            Bytes.toBytes("attributes"),
                            Bytes.toBytes("for_user"),
                            CompareFilter.CompareOp.EQUAL,
                            new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {

            printValues.printAllValues(res);
        }

        userScanResult.close();
    }
}
```

The column family:column
whose value should be compared

SingleColumnValueFilter

… where for_user="Daniel"

```java
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                              Bytes.toBytes("attributes"),
                              Bytes.toBytes("for_user"),
                              CompareFilter.CompareOp.EQUAL,
                              new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {

            printValues.printAllValues(res);
        }

        userScanResult.close();
    }
}
```

The operator

SingleColumnValueFilter … where for_user="Daniel"

```
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                                Bytes.toBytes("attributes"),
                                Bytes.toBytes("for_user"),
                                CompareFilter.CompareOp.EQUAL,
                                new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {

            printValues.printAllValues(res);
        }

        userScanResult.close();
    }
}
```

A Comparator object

SingleColumnValueFilter

… where for_user="Daniel"

```
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                          Bytes.toBytes("attributes"),
                          Bytes.toBytes("for_user"),
                          CompareFilter.CompareOp.EQUAL,
                          new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {

            printValues.printAllValues(res);
        }

        userScanResult.close();
    }
}
```

The value to be compared against

SingleColumnValueFilter … where for_user="Daniel"

```java
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {

            printValues.printAllValues(res);
        }

        userScanResult.close();
    }
}
```

This option will exclude any rows where the column is completely missing

SingleColumnValueFilter … where for_user="Daniel"

```
public class colValueFilter {

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter filter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        filter.setFilterIfMissing(true);

        Scan userScan = new Scan();
        userScan.setFilter(filter);
        ResultScanner userScanResult = table.getScanner(userScan);

        for (Result res : userScanResult) {

                    printValues.printAllValues(res);
            }

       userScanResult.close();
    }
}
```

Set this filter for a **Scan object** and perform the scan operation

Example 20:
Filtering rows based on multiple conditions

FilterList

FilterList

Let's see how to filter based on the values in multiple columns

This is equivalent to using a where clause in SQL for multiple columns

```
… where for_user="Daniel" and
type="Friend_Request"
```

FilterList

… where for_user="Daniel" and type="Friend_Request"

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);


        Date endDate = new Date();
        Date startDate = DateUtils.addDays(endDate, -3);
        Scan userTypeScan = new Scan();
        userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
        userTypeScan.setFilter(filters);
        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

Use 2 SingleColumnValueFilters and add them to a FilterList

FilterList

… where for_user="Daniel"
and type="Friend_Request"

```
 filterList {
tatic void main(String[] args) throws IOException {

iguration conf = HBaseConfiguration.create();

ection connection = ConnectionFactory.createConnection(conf);
e table = connection.getTable(TableName.valueOf("notifications"));
```

```
SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
         Bytes.toBytes("attributes"),
        Bytes.toBytes("for_user"),
        CompareFilter.CompareOp.EQUAL,
        new BinaryComparator(Bytes.toBytes("Daniel")));

userFilter.setFilterIfMissing(true);

SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
         Bytes.toBytes("attributes"),
        Bytes.toBytes("type"),
        CompareFilter.CompareOp.EQUAL,
        new BinaryComparator(Bytes.toBytes("Friend Request")));

typeFilter.setFilterIfMissing(true);
```

```
<Filter> listOfFilters = new ArrayList<>();
OfFilters.add(typeFilter);
OfFilters.add(userFilter);

erList filters = new FilterList(listOfFilters);


 endDate = new Date();
 startDate = DateUtils.addDays(endDate, -3);
 userTypeScan = new Scan();
TypeScan.setTimeRange(startDate.getTime(),endDate.g
TypeScan.setFilter(filters);
tScanner userTypeScanResult = table.getScanner(userTypeScan);

(Result res : userTypeScanResult) {

printValues.printAllValues(res);

TypeScanResult.close();
```

2 SingleColumnValueFilters

FilterList

… where for_user="Daniel"
and type="Friend_Request"

```
] args) throws IOException {

onfiguration.create();

nectionFactory.createConnection(conf);
tTable(TableName.valueOf("notifications"));
```

```
SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
        Bytes.toBytes("attributes"),
        Bytes.toBytes("for_user"),
        CompareFilter.CompareOp.EQUAL,
        new BinaryComparator(Bytes.toBytes("Daniel")));

userFilter.setFilterIfMissing(true);
```

```
alueFilter typeFilter = new SingleColumnValueFilter(
Bytes("attributes"),
.toBytes("type"),
reFilter.CompareOp.EQUAL,
inaryComparator(Bytes.toBytes("Friend Request")));

setFilterIfMissing(true);

listOfFilters = new ArrayList<>();
.add(typeFilter);
.add(userFilter);

lters = new FilterList(listOfFilters);


addDays(endDate, -3);
n();
tartDate.getTime(),endDate.getTime());
ers);
esult = table.getScanner(userTypeScan);

canResult) {

ues(res);
```

Set up 1
SingleColumnValueFilter for
the "for_user" column

FilterList

```
ss filterList {
 static void main(String[] args) throws IOException {

nfiguration [illegible] = [illegible]se[illegible]nfigu[illegible]tion.create();

nnection co[illegible]ction [illegible]Co[illegible]ec[illegible]acto[illegible].[illegible]eat[illegible]ect[illegible]n(con[illegible]
ble table = [illegible]cti[illegible].g[illegible]Table[illegible]able[illegible]am[illegible]val[illegible]f("n[illegible]ficat[illegible]s"

  SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
       Bytes.toBytes("attributes"),
          Bytes.toBytes("for_user"),
          CompareFilter.CompareOp.EQUAL,
          new BinaryComparator(Bytes.toBytes("Daniel")));


   userFilter.setFilterIfMissing(true);
```

… where for_user="Daniel"
and type="Friend_Request"

```
SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
        Bytes.toBytes("attributes"),
        Bytes.toBytes("type"),
        CompareFilter.CompareOp.EQUAL,
        new BinaryComparator(Bytes.toBytes("Friend Request")));

typeFilter.setFilterIfMissing(true);
```

```
st<Filter> listOfFilters = new ArrayList<>();
stOfFilters.add(typeFilter);
stOfFilters.add(userFilter);

lterList filters = new FilterList(listOfFilters);


te endDate = new Date();
te startDate = DateUtils.addDays(endDate, -3);
an userTypeScan = new Scan();
erTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
erTypeScan.setFilter(filters);
sultScanner userTypeScanResult = table.getScanner(userTypeScan);

r (Result res : userTypeScanResult) {

  printValues.printAllValues(res);

erTypeScanResult.close();
```

Set up another SingleColumnValueFilter for the "type" column

FilterList

… where for_user="Daniel" and type="Friend_Request"

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

FilterList filters = new FilterList(listOfFilters);

        Date endDate = new Date();
        Date startDate = DateUtils.addDays(endDate, -3);
        Scan userTypeScan = new Scan();
        userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
        userTypeScan.setFilter(filters);
        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

Add the filters to a List

FilterList

… where for_user="Daniel" and type="Friend_Request"

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
                listOfFilters.add(typeFilter);
                listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);

        Date endDate = new Date();
        Date startDate = DateUtils.addDays(endDate, -3);
        Scan userTypeScan = new Scan();
        userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
        userTypeScan.setFilter(filters);
        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

Use the List to create a FilterList

FilterList

… where for_user="Daniel"
and type="Friend_Request"

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);


        Date endDate = new Date();
        Date startDate = DateUtils.addDays(endDate, -3);

Scan userTypeScan = new Scan();
userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
userTypeScan.setFilter(filters);

        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

Set up a Scan object and set the filter option using the FilterList

Example 21:
Retrieving rows based on a time range

setTimeRange

setTimeRange

```
[illegible]tic [illegible]s fi[illegible]rLis[illegible] {
    public static void main(String[] args) throws IOException {

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);


        Date endDate = new Date();
        Date startDate = DateUtils.addDays(endDate, -3);

Scan userTypeScan = new Scan();
userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());


userTypeScan.setFilter(filters);

        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

Scan objects have a setTimeRange method to retrieve values created within a specified time range

setTimeRange

```
[illegible]tic [illegible]s fi[illegible]rLis[illegible] {
    public static void main(String[] args) throws IOException {

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);


        Date endDate = new Date();
        Date startDate = DateUtils.addDays(endDate, -3);

Scan userTypeScan = new Scan();
userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());


userTypeScan.setFilter(filters);

        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

Let's see how to retrieve **only the values created in the last 3 days**

setTimeRange

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);

  Date endDate = new Date();
  Date startDate = DateUtils.addDays(endDate, -3);
  Scan userTypeScan = new Scan();

userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
        userTypeScan.setFilter(filters);
        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

setTimeRange needs a start timestamp and an end timestamp

setTimeRange

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);

        Date endDate = new Date();
        Date startDate = DateUtils.addDays(endDate, -3);
        Scan userTypeScan = new Scan();

     userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
        userTypeScan.setFilter(filters);
        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

The end date = current date

setTimeRange

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing([illegible]);

        SingleColumnValueFilter type[illegible]ter = [illegible]w SingleColumnValueFilter(
                Bytes.toBytes("attri[illegible]es"),
                Bytes.toBytes("type"[illegible]
                CompareFilter.Compar[illegible].EQUA[illegible]
                new BinaryComparator[illegible]tes.t[illegible]ces([illegible]equest")));

        typeFilter.setFilterIfMissin[illegible]rue);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);
```

The start date = current date -3

```
Date endDate = new Date();
Date startDate = DateUtils.addDays(endDate, -3);
Scan userTypeScan = new Scan();

userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
```

```
        userTypeScan.setFilter(filters);
        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);

        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult.close();
    }
}
```

setTimeRange

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

        Configuration conf = HBaseConfiguration.create();

        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        SingleColumnValueFilter userFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("for_user"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Daniel")));

        userFilter.setFilterIfMissing(true);

        SingleColumnValueFilter typeFilter = new SingleColumnValueFilter(
                Bytes.toBytes("attributes"),
                Bytes.toBytes("type"),
                CompareFilter.CompareOp.EQUAL,
                new BinaryComparator(Bytes.toBytes("Friend Request")));

        typeFilter.setFilterIfMissing(true);

        List<Filter> listOfFilters = new ArrayList<>();
        listOfFilters.add(typeFilter);
        listOfFilters.add(userFilter);

        FilterList filters = new FilterList(listOfFilters);
```

```
Date endDate = new Date();
Date startDate = DateUtils.addDays(endDate, -3);
Scan userTypeScan = new Scan();

userTypeScan.setTimeRange(startDate.getTime(),endDate.getTime());
```

```
        userTypeScan.setFilter(filters);
        ResultScanner userTypeScanResult = table.getScanner(userTypeScan);
        for (Result res : userTypeScanResult) {

            printValues.printAllValues(res);
        }
        userTypeScanResult
    }
}
```

This Scan will now only retrieve values in the specified time range

Example 22:
Incrementing a value

Counter

Counter

In a notifications database, you would need to track a few metrics

#views, #clicks etc

You'll need to maintain counts for these metrics

Counter

#views, #clicks etc

The typical way to do such an operation

1. Read the current value of the metric
2. Increment it
3. Update the value in HBase

Counter

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));


        long opencount=1;
        if (val!=null){
             opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);


         table.incrementColumnValue(Bytes.toBytes("2"),Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);


          Increment increment =new Increment(Bytes.toBytes("2"));
          increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
          increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);


         table.increment(increment);

    }
}
```

1. Read
2. Increment
3. Update

Counter

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));
```

1. Read

```
Get get =new Get(Bytes.toBytes("2"));
get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
Result result = table.get(get);
byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
```

2. Increment

```
long opencount=1;
if (val!=null){
     opencount = Bytes.toLong(val)+1;
}

Put put =new Put(Bytes.toBytes("2"));
put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

table.put(put);
```

3. Update

```
        table.incrementColumnValue(Bytes.toBytes("2"),Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

         Increment increment =new Increment(Bytes.toBytes("2"));
         increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
         increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

        table.increment(increment);

    }
}
```

Counter

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));
```

1. Read

```
Get get =new Get(Bytes.toBytes("2"));
get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));

Result result = table.get(get);
byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
```

```
        long opencount=1;
        if (val!=null){
            opencount = Bytes.toLon
        }

        Put put =new Put(Bytes.toByte
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);

    table.incrementColumnValue(Bytes.toBytes("2"),Bytes.toBytes("metrics"),Bytes.toBytes("views"),

     Increment increment =new Increment(Bytes.toBytes("2"));
     increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
     increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

    table.increment(increment);

  }
}
```

Use a Get to read the value for the metric

Counter

```
[illegible]
[illegible]
        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));

        long opencount=1;
                if (val!=null){
                        opencount = Bytes.toLong(val)+1;
                }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put[illegible]);

        table.incrementColumnValue(Bytes.toByt[illegible]"2")[illegible]tes.toBytes([illegible]trics[illegible]iews"),[illegible]

        Increment increment =new Increment(By[illegible]toBy[illegible]("2"));
        increment.addColumn(Bytes.toBytes("met[illegible]cs"),[illegible]es.toBytes([illegible]icks"[illegible]1);
        increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

        table.increment(increment);

    }
}
```

2. Increment

If the column doesn't yet exist, initialize the value to 1

Counter

If the column doesn't yet exist,
initialize the value to 1

```
get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));

long opencount=1;
    if (val!=null){
        opencount = Bytes.toLong(val)+1;
    }

Put put =new Put(Bytes.toBytes("2"));
put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

table.put(put);
```

2. Increment

else, increment the value

Counter

Use a Put to update the value for the metric

```
Put put =new Put(Bytes.toBytes("2"));
put.addColumn(Bytes.toBytes("metrics"),
        Bytes.toBytes("open"),
        Bytes.toBytes(opencount));

table.put(put);
```

3. Update

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"),
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), ytes.toBytes( pen"));

        long opencount=1;
        if (val!=null){
            opencount = Bytes.toLong(val)+1;
        }

        table.incrementColumnValue(Bytes.toBytes("2"),Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

        Increment increment =new Increment(Bytes.toBytes("2"));
        increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
        increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

        table.increment(increment);

    }
}
```

Counter

#views, #clicks etc

1. Read
2. Increment
3. Update

This is not an atomic operation

Counter

#views, #clicks etc

1. Read
2. Increment
3. Update

Each of these steps can fail/succeed independently

Counter

#views, #clicks etc

1. Read
2. Increment
3. Update

HBase provides Counters to allow for an atomic increment operation

Counter

This operation will increment the value for the specified row, column

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));

        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));

        long opencount=1;
        if (val!=null){
             opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);
```

```
table.incrementColumnValue(
      Bytes.toBytes("2"),
      Bytes.toBytes("metrics"),
      Bytes.toBytes("views"),
      1);
```

row id

col family : column

increment by

```
        Increment increment =new Increment(Bytes.toBytes("2"));
        increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
        increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

        table.increment(increment);

    }
}
```

Counter

There's also a shell command to do the same operation

```
incr 'notifications',2,'metrics:views',1
```

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes(


        long opencount=1;
        if (val!=null){
             opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);
```

```
table.incrementColumnValue(
        Bytes.toBytes("2"),
        Bytes.toBytes("metrics"),
        Bytes.toBytes("views"),
        1);
```

```
            Increment increment =new Increment(Bytes.toBytes("2"));
            increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
            increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);


           table.increment(increment);

       }
    }
```

Counter

Columns which use counters should be initialized using an increment operation

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));


        long opencount=1;
        if (val!=null){
             opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);
```

```
table.incrementColumnValue(
     Bytes.toBytes(“2"),
     Bytes.toBytes(“metrics"),
     Bytes.toBytes(“views"),
     1);
```

```
         Increment increment =new Increment(Bytes.toBytes("2"));
         increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
         increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);


        table.increment(increment);

    }
}
```

Counter

You cannot use a put operation to insert a value and then try to increment it

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));


        long opencount=1;
        if (val!=null){
            opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);
```

```
table.incrementColumnValue(
     Bytes.toBytes("2"),
     Bytes.toBytes("metrics"),
     Bytes.toBytes("views"),
     1);
```

```
            Increment increment =new Increment(Bytes.toBytes("2"));
            increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
            increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);


           table.increment(increment);

    }
}
```

Counter

This method can only be used to increment a single row, column

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));


        long opencount=1;
        if (val!=null){
            opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);
```

```
table.incrementColumnValue(
     Bytes.toBytes(“2"),
     Bytes.toBytes(“metrics"),
     Bytes.toBytes(“views"),
     1);
```

```
        Increment increment =new Increment(Bytes.toBytes("2"));
        increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
        increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);


        table.increment(increment);

    }
}
```

Counter

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));


        long opencount=1;
        if (val!=null){
            opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);

Increment increment =new Increment(Bytes.toBytes("2"));

increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

table.increment(increment);
    }
}
```

Use an Increment object to increment multiple columns for a specific row id

Counter

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));


        long opencount=1;
        if (val!=null){
            opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);


         table.incrementColumnValue(Bytes.toBytes("2"),Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

Increment increment =new Increment(Bytes.toBytes("2"));

increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

table.increment(increment);
    }
}
```

Add the columns and increment values to the Increment object

Counter

```
public class counter {

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

        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf("notifications"));


        Get get =new Get(Bytes.toBytes("2"));
        get.addColumn(Bytes.toBytes("metrics"), Bytes.toBytes("open"));
        Result result = table.get(get);
        byte[] val= result.getValue(Bytes.toBytes("metrics"), Bytes.toBytes("open"));


        long opencount=1;
        if (val!=null){
            opencount = Bytes.toLong(val)+1;
        }

        Put put =new Put(Bytes.toBytes("2"));
        put.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("open"),Bytes.toBytes(opencount));

        table.put(put);


         table.incrementColumnValue(Bytes.toBytes("2"),Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

Increment increment =new Increment(Bytes.toBytes("2"));

increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("clicks"),1);
increment.addColumn(Bytes.toBytes("metrics"),Bytes.toBytes("views"),1);

table.increment(increment);
    }
}
```

Pass it to the increment method