

## Table of Contents

<b>1. Introduction.....</b>	<b>2</b>
<b>2. Use cases: row-level filters.....</b>	<b>2</b>
<b>Use case #1: restrict users to access subset of rows based on group the user belongs to .....</b>	<b>2</b>
Policy details .....	3
Query Results.....	3
<b>Use case #2: restrict users to access subset of rows based on data in another table .....</b>	<b>4</b>
Policy details .....	4
Query Results.....	5
<b>Use case #3: restrict users to access subset of rows based on data in reference tables .....</b>	<b>5</b>
Policy details .....	6
Query Results.....	6
<b>Use case #4: allow specific users/groups to access all rows .....</b>	<b>7</b>
Policy details .....	7
Query Results.....	8
<b>3. Use cases: data-masking .....</b>	<b>8</b>
<b>Use case #1: only last 4 digits of phone numbers should be shown.....</b>	<b>8</b>
Policy details .....	8
Query Results.....	9
<b>Use case #2: only year value of date_of_birth column should be shown.....</b>	<b>9</b>
Policy details .....	9
Query Results.....	10
<b>Use case #3: name_last column value should not be shown .....</b>	<b>10</b>
Policy details .....	10
Query Results.....	11
<b>Use case #4: apply a custom transformation to name_first column .....</b>	<b>11</b>
Policy details .....	11
Query Results.....	12
<b>4. Policy Model .....</b>	<b>12</b>
<b>5. References.....</b>	<b>13</b>
<b>6. FAQ .....</b>	<b>13</b>

## 1. Introduction

Apache Ranger provides centralized security for Enterprise Hadoop ecosystem, including fine-grained access control and centralized auditing. Apache Ranger policy model supports policies to allow or deny an access based on users, groups, access-types and other dynamic attributes like IP-address, time of access, etc. In addition, the model also supports authorization based on the classification of the resources - like PII, FINANCE, SESITIVE, etc. ([tag-based authorization](#)).

Couple of the often asked features are the ability to allow users to access only a subset of rows in a table or restrict users to access only masked/redacted value of sensitive data. In version 0.6 release, Apache Ranger policy model has been enhanced to support row-filtering and data-masking features. With this release, Apache Ranger plugin for Apache Hive implements these new features, allowing security administrators to set appropriate row-filters and data-masking for Hive tables and columns. This document covers various details of these enhancements, using a number of examples.

## 2. Use cases: row-level filters

Let us go through few use cases to understand the row-level filter feature. Hive table given below will be used in the following use cases:

Table: customer

id	name_first	name_last	addr_country	date_of_birth	phone_num
1	Mackenzzy	Smith	US	1993-12-18	123-456-7890
2	Sherlyn	Miller	US	1975-03-22	234-567-8901
3	Khiana	Wilson	US	1989-08-14	345-678-9012
4	Jack	Thompson	US	1962-10-28	456-789-0123
5	Audrey	Taylor	UK	1985-01-11	12-3456-7890
6	Ruford	Walker	UK	1976-05-19	23-4567-8901
7	Marta	Lloyd	UK	1981-07-23	34-5678-9012
8	Derick	Schneider	DE	1982-04-17	12-345-67890
9	Anna	Richter	DE	1995-09-07	23-456-78901
10	Raina	Graf	DE	1999-02-06	34-567-89012
11	Felix	Lee	CA	1982-04-17	321-654-0987
12	Adam	Brown	CA	1995-09-07	432-765-1098
13	Lucas	Jones	CA	1999-02-06	543-876-2109
14	Yvonne	Dupont	FR	1982-04-17	01-23-45-67-89
15	Pascal	Fournier	FR	1995-09-07	23-45-67-89-01
16	Ariel	Simon	FR	1999-02-06	34-56-78-90-12

### Use case #1: restrict users to access subset of rows based on group the user belongs to

Let us start with a simple use case. The requirement is to restrict users to access only records of customers located in the same country where the user works. For example, US users can only access US customer records; and UK users can only access UK customer records. Users belong to one of the country-specific groups maintained in LDAP/AD, as shown in the example below:

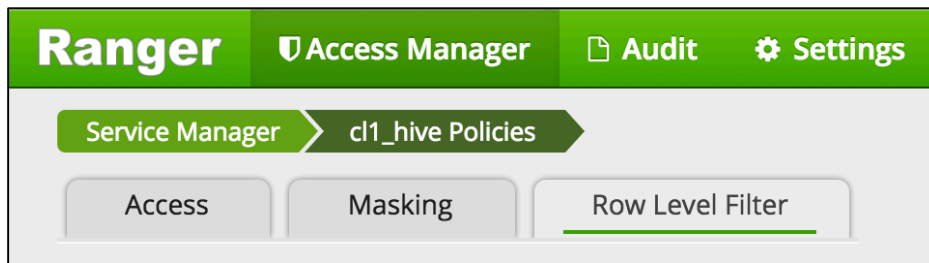
Group name	Users
us-employees	john,scott
uk-employees	mary,adam
de-employees	drew,alice

# Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

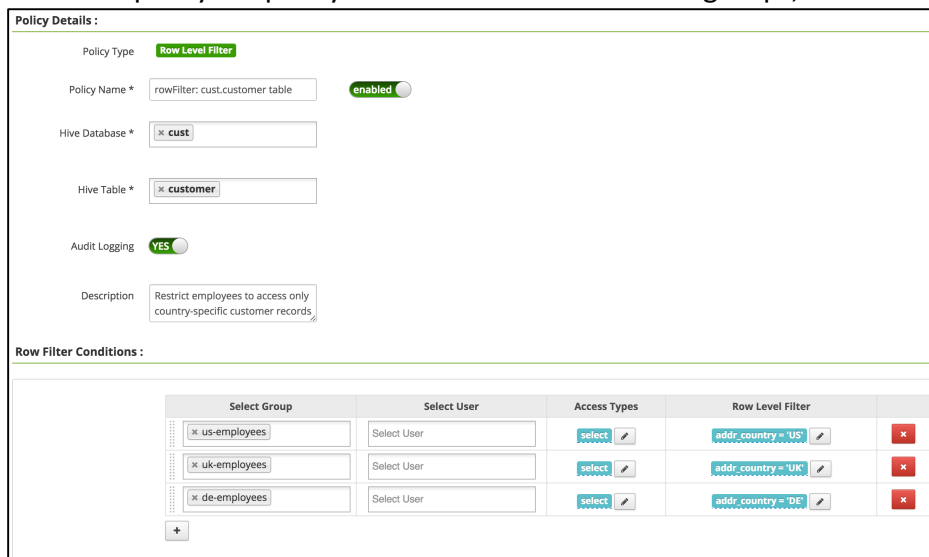
## Policy details

Follow the steps given below to create a policy to enforce row-level access control:

1. Select 'Row Level Filter' tab



2. Add a policy to specify row-filters for various user groups, as shown below:

The screenshot shows the 'Policy Details' form in Apache Ranger. The 'Policy Type' is 'Row Level Filter'. The 'Policy Name' is 'rowFilter: cust.customer table' and is marked as 'enabled'. The 'Hive Database' is 'cust' and the 'Hive Table' is 'customer'. 'Audit Logging' is set to 'YES'. The description is 'Restrict employees to access only country-specific customer records'. Below this, the 'Row Filter Conditions' section contains a table with three rows, each representing a user group and its corresponding filter condition.

Select Group	Select User	Access Types	Row Level Filter	
us-employees	Select User	select	addr_country = 'US'	✖
uk-employees	Select User	select	addr_country = 'UK'	✖
de-employees	Select User	select	addr_country = 'DE'	✖

## Query Results

Let us now access the customer table with different users and see how the results have only subset of rows in the table, as set by the above Apache Ranger policy:

*User john, a member of us-employees group:*

The result includes only customers in US.

```
[john@localhost ~]$ beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 1 | Mackenzy | Smith | US | 1993-12-18 | 123-456-7890 |
| 2 | Sherlyn | Miller | US | 1975-03-22 | 234-567-8901 |
| 3 | Khiana | Wilson | US | 1989-08-14 | 345-678-9012 |
| 4 | Jack | Thompson | US | 1962-10-28 | 456-789-0123 |
+-----+-----+-----+-----+-----+-----+
```

*User mary, a member of uk-employees group:*

The result includes only customers in UK.

## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

```
[mary@localhost ~]$ beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 5 | Audrey | Taylor | UK | 1985-01-11 | 12-3456-7890 |
| 6 | Ruford | Walker | UK | 1976-05-19 | 23-4567-8901 |
| 7 | Marta | Lloyd | UK | 1981-07-23 | 34-5678-9012 |
+-----+-----+-----+-----+-----+-----+
```

*User drew, a member of de-employees group:*

The result includes only customers in DE.

```
[drew@localhost ~]$ beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 8 | Derick | Schneider | DE | 1982-04-17 | 12-345-67890 |
| 9 | Anna | Richter | DE | 1995-09-07 | 23-456-78901 |
| 10 | Raina | Graf | DE | 1999-02-06 | 34-567-89012 |
+-----+-----+-----+-----+-----+-----+
```

[Use case #2: restrict users to access subset of rows based on data in another table](#)

Let us get a little more sophisticated now. The requirement is to use an attribute in employee table to find the country where a user works, instead of using user-groups in LDAP/AD.

Table: employee

```
+-----+-----+-----+
| id | userid | country |
+-----+-----+-----+
| 1 | john | US |
| 2 | scott | US |
| 3 | mary | UK |
| 4 | adam | UK |
| 5 | drew | DE |
| 6 | alice | DE |
+-----+-----+-----+
```

### Policy details

Update the policy created earlier to use the following filter, for `group=public`, as shown below:

```
addr_country in (select e.country from emp.employee e
                 where e.userid = current_user())
```

Please note that multiple policy-items in the previous use case, one for each user-group, are replaced with a single policy-item in the updated policy below.

## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

**Policy Details :**

Policy Type **Row Level Filter**

Policy ID **417**

Policy Name \*  **enabled**

Hive Database \*

Hive Table \*

Audit Logging **YES**

Description

**Row Filter Conditions :** hide ->

Select Group	Select User	Access Types	Row Level Filter
<input type="text" value="public"/>	<input type="text" value="Select User"/>	<input type="text" value="select"/>	<input type="text" value="addr_country in (select e.country from emp.employee e where e.userid = current_user())"/>

### Query Results

The query results should be same as the previous use case.

### Use case #3: restrict users to access subset of rows based on data in reference tables

Let us get more sophisticated now. Instead of restricting users to customer records of a single country, the requirement is to allow users to access records of customers who are located in the same region as the user. For example, a user in US should be able to access records of customers in North America region; and a user in DE or FR should be able to access records of customers in European Union region.

A reference table, `country_region`, is used to find the region for a given country. This table and couple of helper views, shown below, will be used to setup appropriate filter in Apache Ranger policy:

Table: `country_region`

```
+-----+-----+
| country | region |
+-----+-----+
| US      | NA     |
| CA      | NA     |
| UK      | UK     |
| DE      | EU     |
| FR      | EU     |
+-----+-----+
```

```
create view employee_region(userid, region) as
  select e.userid, cr.region from emp.employee e, emp.country_region cr
  where e.country = cr.country;
```

```
create view employee_country(userid, country) as
  select er.userid, cr.country from emp.employee_region er, emp.country_region cr
  where cr.region = er.region;
```

## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

### Policy details

Update the policy created earlier to use the following filter, for `group=public`, as shown below:

```
addr_country in (select ec.country from emp.employee_country ec
                 where ec.userid = current_user())
```

**Policy Details :**

Policy Type: **Row Level Filter**

Policy ID: **417**

Policy Name \*: rowFilter: cust.customer table **enabled**

Hive Database \*: **cust**

Hive Table \*: **customer**

Audit Logging: **YES**

Description: Restrict employees to access only country-specific customer records

**Row Filter Conditions :**

Select Group	Select User	Access Types	Row Level Filter
<b>public</b>	Select User	<b>select</b>	addr_country in (select ec.country from emp.employee_country ec where ec.userid = current_user())

### Query Results

The query results now will include customers in all countries in the region where the user works – not just the customers in the country where the user works.

#### User john, who works in 'US':

The result includes customers in North America region.

```
[john@localhost ~]$ beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
-----+-----+-----+-----+-----+-----+
| 1 | Mackenzy | Smith | US | 1993-12-18 | 123-456-7890 |
| 2 | Sherlyn | Miller | US | 1975-03-22 | 234-567-8901 |
| 3 | Khiana | Wilson | US | 1989-08-14 | 345-678-9012 |
| 4 | Jack | Thompson | US | 1962-10-28 | 456-789-0123 |
| 11 | Felix | Lee | CA | 1982-04-17 | 321-654-0987 |
| 12 | Adam | Brown | CA | 1995-09-07 | 432-765-1098 |
| 13 | Lucas | Jones | CA | 1999-02-06 | 543-876-2109 |
-----+-----+-----+-----+-----+-----+

```

#### User mary, who works in 'UK':

The result includes customers in UK region.

```
[mary@localhost ~]$ beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
-----+-----+-----+-----+-----+-----+
| 5 | Audrey | Taylor | UK | 1985-01-11 | 12-3456-7890 |
| 6 | Ruford | Walker | UK | 1976-05-19 | 23-4567-8901 |
| 7 | Marta | Lloyd | UK | 1981-07-23 | 34-5678-9012 |
-----+-----+-----+-----+-----+-----+

```

## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

User drew, who works in 'DE':

The result includes customers in European Union region.

```
[drew@localhost ~]$ beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 8 | Derick | Schneider | DE | 1982-04-17 | 12-345-67890 |
| 9 | Anna | Richter | DE | 1995-09-07 | 23-456-78901 |
| 10 | Raina | Graf | DE | 1999-02-06 | 34-567-89012 |
| 14 | Yvonne | Dupont | FR | 1982-04-17 | 01-23-45-67-89 |
| 15 | Pascal | Fournier | FR | 1995-09-07 | 23-45-67-89-01 |
| 16 | Ariel | Simon | FR | 1999-02-06 | 34-56-78-90-12 |
+-----+-----+-----+-----+-----+-----+
```

Use case #4: allow specific users/groups to access all rows

It might be necessary to allow specific users/group to access all rows in a table, while restricting rest of the users to only subset of rows. For example, let us say the requirement is for user 'falcon' to be able to access all rows.

Policy details

Update the policy created earlier by inserting a policy-item, with user=falcon and empty value for filter, as shown below. Ensure that the new policy-item appears before the policy-item that has groups=public.

While determining the filter to apply for a table, Apache Ranger policy engine evaluates the policy-items in the order listed in the policy. The filter specified in the first policy-item that matches the access-request (i.e. user/groups) will be used in the query.

In this case, since no filter is specified for user falcon, the user will be allowed access all rows.

The screenshot displays the 'Policy Details' configuration page for a 'Row Level Filter' policy. The policy is named 'rowFilter: cust.customer table', is enabled, and applies to the 'cust' database and 'customer' table. The description is 'Restrict employees to access only country-specific customer records'. The 'Row Filter Conditions' section shows two items:

Select Group	Select User	Access Types	Row Level Filter
Select Group	falcon	select	Add Row Filter +
public	Select User	select	addr_country in (select ec.country from emp.employee_country ec where ec.userid = current_user())

## Query Results

The query results will include all customers, without any filters.

```
[falcon@localhost ~]# beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
```

id	name_first	name_last	addr_country	date_of_birth	phone_num
1	Mackenzzy	Smith	US	1993-12-18	123-456-7890
2	Sherlyn	Miller	US	1975-03-22	234-567-8901
3	Khiana	Wilson	US	1989-08-14	345-678-9012
4	Jack	Thompson	US	1962-10-28	456-789-0123
5	Audrey	Taylor	UK	1985-01-11	12-3456-7890
6	Ruford	Walker	UK	1976-05-19	23-4567-8901
7	Marta	Lloyd	UK	1981-07-23	34-5678-9012
8	Derick	Schneider	DE	1982-04-17	12-345-67890
9	Anna	Richter	DE	1995-09-07	23-456-78901
10	Raina	Graf	DE	1999-02-06	34-567-89012
11	Felix	Lee	CA	1982-04-17	321-654-0987
12	Adam	Brown	CA	1995-09-07	432-765-1098
13	Lucas	Jones	CA	1999-02-06	543-876-2109
14	Yvonne	Dupont	FR	1982-04-17	01-23-45-67-89
15	Pascal	Fournier	FR	1995-09-07	23-45-67-89-01
16	Ariel	Simon	FR	1999-02-06	34-56-78-90-12

## 3. Use cases: data-masking

Let us use `customer` table used in row-level filter use cases for data-masking feature use cases as well.

### Use case #1: only last 4 digits of phone numbers should be shown

It is often necessary to show only part of the sensitive data to most users. In this use case, we will see the details of Apache Ranger policy to show only last 4 digits unmasked for values in `phone_num` column.

### Policy details

Follow the steps given below to create a policy to enforce masking on the column:

1. Select 'Masking' tab



2. Add a policy with the following details:



## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

**Policy Details :**

Policy Type: **Masking**

Policy ID: **419**

Policy Name \*:  **enabled**

Hive Database \*:

Hive Table \*:

Hive Column \*:

Audit Logging: **YES**

Description:

**Mask Conditions :**

Select Group:  Select User:  Access Types:

Select Masking Option

Redact

Partial mask: show last 4

Partial mask: show first 4

Hash

Nullify

Unmasked (retain original value)

Date: show only year

Custom

### Query Results

The result of query on customer table will only show last 4 digits of phone\_num column unmasked, as shown below. In addition, please note that the row-level filter policy created earlier is also applied in the query result.

```
[john@localhost ~]# beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 1 | Mackenzy | Smith | US | 1993-12-18 | xxx-xxx-7890 |
| 2 | Sherlyn | Miller | US | 1975-03-22 | xxx-xxx-8901 |
| 3 | Khiana | Wilson | US | 1989-08-14 | xxx-xxx-9012 |
| 4 | Jack | Thompson | US | 1962-10-28 | xxx-xxx-0123 |
| 11 | Felix | Lee | CA | 1982-04-17 | xxx-xxx-0987 |
| 12 | Adam | Brown | CA | 1995-09-07 | xxx-xxx-1098 |
| 13 | Lucas | Jones | CA | 1999-02-06 | xxx-xxx-2109 |
+-----+-----+-----+-----+-----+-----+

```

### Use case #2: only year value of date\_of\_birth column should be shown

In this use case, we will see the details of Apache Ranger policy to show only year value of date\_of\_birth column. To ensure that the value returned is of type date, a constant value of '01' will be shown for day and month fields of the column value.

### Policy details

Add a policy with the following details:

## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

**Policy Details :**

Policy Type: **Masking**

Policy Name \*:  **enabled**

Hive Database \*:

Hive Table \*:

Hive Column \*:

Audit Logging: **YES**

Description:

**Mask Conditions :**

Select Group:  Select User:  Access Types:

Select Masking Option:

- Redact
- Partial mask: show last 4
- Partial mask: show first 4
- Hash
- Nullify
- Unmasked (retain original value)
- Date: show only year
- Custom

### Query Results

In addition to earlier masked result on phone\_num column, the query result will only show year value for date\_of\_birth column, as shown below.

```
[john@localhost ~]# beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 1 | Mackenzy | Smith | US | 1993-01-01 | xxx-xxx-7890 |
| 2 | Sherlyn | Miller | US | 1975-01-01 | xxx-xxx-8901 |
| 3 | Khiana | Wilson | US | 1989-01-01 | xxx-xxx-9012 |
| 4 | Jack | Thompson | US | 1962-01-01 | xxx-xxx-0123 |
| 11 | Felix | Lee | CA | 1982-01-01 | xxx-xxx-0987 |
| 12 | Adam | Brown | CA | 1995-01-01 | xxx-xxx-1098 |
| 13 | Lucas | Jones | CA | 1999-01-01 | xxx-xxx-2109 |
+-----+-----+-----+-----+-----+-----+

```

### Use case #3: name\_last column value should not be shown

In this use case, we will see the details of Apache Ranger policy to effectively not show the value of name\_last column – by returning NULL as value for all rows.

### Policy details

Add a policy with the following details:

## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

**Policy Details :**

Policy Type: **Masking**

Policy Name \*:  **enabled**

Hive Database \*:

Hive Table \*:

Hive Column \*:

Audit Logging: **YES**

Description:

**Mask Conditions :**

Select Group:  Select User:  Access Types:

Select Masking Option

Redact

Partial mask: show last 4

Partial mask: show first 4

Hash

Nullify

Unmasked (retain original value)

Date: show only year

Custom

### Query Results

In addition to earlier masked result on phone\_num, date\_of\_birth columns, the query result will only show null value for name\_last column, as shown below.

```
[john@localhost ~]# beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 1 | Mackenzy | NULL | US | 1993-01-01 | xxx-xxx-7890 |
| 2 | Sherlyn | NULL | US | 1975-01-01 | xxx-xxx-8901 |
| 3 | Khiana | NULL | US | 1989-01-01 | xxx-xxx-9012 |
| 4 | Jack | NULL | US | 1962-01-01 | xxx-xxx-0123 |
| 11 | Felix | NULL | CA | 1982-01-01 | xxx-xxx-0987 |
| 12 | Adam | NULL | CA | 1995-01-01 | xxx-xxx-1098 |
| 13 | Lucas | NULL | CA | 1999-01-01 | xxx-xxx-2109 |
+-----+-----+-----+-----+-----+-----+

```

### Use case #4: apply a custom transformation to name\_first column

In this use case, we will see the details of Apache Ranger policy to set a custom expression to transform the value of name\_first column.

#### Policy details

Add a policy with the following details to specify the expression, `initcap(reverse({col}))`, to transform name\_first column values. Ensure that the datatype of the expression is same as the datatype of the column. Token `{col}` in the expression will be replaced by Apache Ranger policy engine by the name of the column on which masking is being applied.

## Row-level filtering and column-masking using Apache Ranger policies in Apache Hive

**Policy Details :**

Policy Type **Masking**

Policy Name \*  **enabled**

Hive Database \*

Hive Table \*

Hive Column \*

Audit Logging **YES**

Description

---

**Mask Conditions :**

Select Group	Select User	Access Types	Select Masking Option
<input type="text" value="public"/>	<input type="text" value="Select User"/>	<input type="button" value="select"/> <input type="button" value="edit"/>	<input type="button" value="Custom"/> <input type="button" value="edit"/> <input type="text" value="initcap(reverse({col}))"/> <input type="button" value="x"/>

### Query Results

In addition to earlier masked result on phone\_num, date\_of\_birth, name\_last columns, the query result will show transformed value name\_first column, as shown below.

```
[john@localhost ~]# beeline -u jdbc:hive2://localhost.localdomain:10000/cust
0: jdbc:hive2://localhost.localdomain:10000> select * from cust.customer;
+-----+-----+-----+-----+-----+-----+
| id | name_first | name_last | addr_country | date_of_birth | phone_num |
+-----+-----+-----+-----+-----+-----+
| 1 | Yznekcama | NULL | US | 1993-01-01 | xxx-xxx-7890 |
| 2 | Nylrehs | NULL | US | 1975-01-01 | xxx-xxx-8901 |
| 3 | Anaihk | NULL | US | 1989-01-01 | xxx-xxx-9012 |
| 4 | Kcaj | NULL | US | 1962-01-01 | xxx-xxx-0123 |
| 11 | Xilef | NULL | CA | 1982-01-01 | xxx-xxx-0987 |
| 12 | Mada | NULL | CA | 1995-01-01 | xxx-xxx-1098 |
| 13 | Sacul | NULL | CA | 1999-01-01 | xxx-xxx-2109 |
+-----+-----+-----+-----+-----+-----+

```

## 4. Policy Model

This section summarizes the updates to Apache Ranger policy model to support row-filter and data-mask features.

1. Support for two new policy-types has been added: row-filter and data-mask
2. RangerServiceDef, the class that represents a service/component (like HDFS/Hive/HBase, ..), has been updated with addition of two attributes – rowFilterDef and dataMaskDef. Services that need row-filter or data-masking feature should populate these attributes with appropriate values
3. RangerPolicy, the class that represents a policy in Apache Ranger, has been updated with addition of two attributes – rowFilterPolicyItems and dataMaskPolicyItems. References

4. RangerPolicyEngine interface has been updated with two new methods, evalRowFilterPolicies() and evalDataMaskPolicies(). Corresponding implementations have been added to RangerPolicyEngineImpl and other related classes.

## 5. References

- [HIVE-13125](#): Support masking and filtering of rows/columns
- [HIVE-13568](#): Add UDFs to support column-masking
- [RANGER-873](#): Ranger policy model to support data-masking
- [RANGER-908](#): Ranger policy model to support row-filtering
- [RANGER-895](#): Ranger Hive plugin to support column-masking
- [RANGER-909](#): Ranger Hive plugin to support row-filtering

## 6. FAQ

1. Which version of Apache Hive version supports row-filtering and data-masking?  
Apache Hive version 2.1 and above support row-filtering and data-masking features
2. Can wildcards or multiple-values be used to specify database/table/column in row-filtering and data-masking policies?  
Use of wildcards or multiple-values to specify database/table/column values is not supported in row-filtering and data-masking policies. Row-filter expressions often refer to columns in the same table; such expressions may not be applicable for other tables, making wildcards/multiple-values less useful or more error-prone here.
3. How are operations like insert, update and delete are handled when the user has row-filter/column-masking on the table/columns?  
Operations insert/update/delete/export are denied for users if row-filter or column-masking policies are applicable on the table for the user.
4. How do I exclude specific users/groups from row-filter and column-masking?  
Policy-items in row-filter and column-masking policies are evaluated in the order listed in the policy. The filter or mask specified in the first matched policy-item will be applied in the query.  
To exclude specific users/groups from row-filter, create a policy-item for specific users/groups with empty value as row-filter and ensure that the policy-item is the first one to appear in the list for the users/groups.  
To exclude specific users/groups from column-masking, create a policy-item for specific users/groups with 'Unmasked' as the masking option and ensure that the policy-item is the first one to appear in the list for the users/groups.