Table of Contents

Transformed columns

Relations (and custom reports too) can use in the query a NetYCE specific extension to the SQL syntax named “TRANSFORM”. This command will convert the values of a named column into its own column.

When creating a relation (or 'named context') the SQL definition of the relation can be modified to automatically create additional columns based on the values found in the SQL results.

This extension of columns is the result of transforming a field value from a selected column to a column name. The value(s) in that column then points to another selected value from the record. In that way, a result table in which every line describes a parameter, can be changed to a table in which this parameter-name becomes a field-name. That field name then receives one of the field values that belong to that parameter.

The syntax is **`TRANSFORM <source-column> WITH <value-column>`** and must be the first line of the SQL statement and can only be used in SELECT queries.

Examples

The example below shows it is very useful to create a column named after the the subnets' name ('Net_name') which then contains the vlan-id ('Vlan_id'). The readability of a template can be much improved this way. Compare the sections below:

```
-- using the relation without TRANSFORM
 port hybrid vlan <Vlan_id@Vlans:Net_name='IPT'> tagged
 port hybrid vlan <Vlan_id@Vlans:Net_name='Users'> untagged
 port hybrid pvid vlan <Vlan_id@Vlans:net_name='Users'>

-- using transformed net-name -> vlan-ids:
 port hybrid vlan <IPT@Vlans> tagged
 port hybrid vlan <Users@Vlans> untagged
 port hybrid pvid vlan <Users@Vlans>
```

The relation query to for 'Vlans' is now:

```
TRANSFORM Net_name WITH Vlan_id
SELECT DISTINCT Ip_subnet.*, Ip_dhcp.*, Node_vrf.Vrf_name
FROM Port_map
  INNER JOIN Ip_map ON (Port_map.Interface_id = Ip_map.Interface_id)
  INNER JOIN Ip_subnet ON (Ip_map.Subnet_id = Ip_subnet.Subnet_id)
  LEFT JOIN Node_vrf ON (Ip_subnet.Vrf_id = Node_vrf.Vrf_id)
  LEFT JOIN Ip_dhcp ON (Ip_dhcp.Dhcp_id = Ip_subnet.Subnet_id)
WHERE Port_map.Hostname = '<Hostname>'
  AND Ip_subnet.Vlan_id > 0
ORDER BY Ip_subnet.Vlan_id
```

The TRANSFORM is mostly used to extend an object like a Client with its configured **custom attributes**. In the example below, the custom column value ('Var_name') is transformed into a

column of that name which then has the value of that custom attribute ('Var_value').

Using the TRANSFORM is the only way to access the Custom attributes of an object.

```
TRANSFORM Var_name WITH Var_value
SELECT Client.*, Par_groups.Var_name, Par_vals.Var_value
FROM Client
LEFT JOIN Par_groups ON (Client.Par_group = Par_groups.Par_group)
LEFT JOIN Par_vals ON (CONCAT_WS('|',Client.Client_type,Client.ClientCode) =
Par_vals.Par_key
    AND Par_groups.Par_group = Par_vals.Par_group
    AND Par_groups.Var_name = Par_vals.Var_name)
WHERE Client.ClientCode = '<clientcode>'
```

The Par_groups table is included to find the custom attributes of the object type (here 'Client') and the Par_vals table has the actual attributes and values for that object. The Par_key to select the actual Client object is often a concatenation of the objects key-fields and will differ per object type. For the 'Client' object are that the Client_type and ClientCode.

Look at the Relation definitions for 'Client', 'Site', 'Service', 'SiteRouter', 'Vlan', 'Node_vrf' and many of the “Ip[4|6]_*' to find the correct use of the TRANSFORM to access an objects extended attributes.

> Note: The current implementation of Custom-attributes use the Par_vals table to store the custom attribute values for all objects. Although very flexible in setup, this method will not allow us to create custom attributes to ports and subnets due to scalability issues. A future NetYCE version will address this issue by implementing a different design.