Query Postprocessor Examples

Contents

Query Postprocessor Examples

This topic provides examples of two postprocessor scripts. One example does a simple data lookup to replace values in one column of the incoming data. The other example aggregates and formats data (although the aggregation itself could be done on the EDW, the example uses it for illustrative purposes). The examples cover all postprocessor API functions documented above in *Perls Functions that a Postprocessor Calls*.

This topic provides a script that loads the required data and runs the postprocessors. The major steps are outlined below.

EXAMPLE DATA SET

Both postprocessor examples execute against the following data set, which is a simple weblog of presidents shopping for cars:

```
+---------------------------+-----------
+----------------------------------------+
|           ts              | user_name |                    url
     |
|       (timestamp)         | (varchar) |                 (varchar)
     |
+---------------------------+-----------
+----------------------------------------*
|2002-02-28T14:23:57.000000Z|gwashington|www.shop.com/home.htm
     |
|2002-02-28T14:37:15.000000Z|gwashington|www.shop.com/catalog.htm
     |
|2002-02-28T14:39:23.000000Z|gwashington|www.shop.com/prodinfo/infiniti.htm
     |
|2002-02-28T14:49:27.000000Z|gwashington|www.shop.com/other_page.htm
     |
|2002-02-28T14:29:30.000000Z|jadams     |www.shop.com/home.htm
     |
|2002-02-28T14:41:51.000000Z|jadams     |www.shop.com/specials.htm
     |
|2002-02-28T14:46:25.000000Z|jadams     |www.shop.com/specials/
audi_europe.htm   |
|2002-02-28T14:46:29.000000Z|jadams     |www.shop.com/purchase.htm
     |
|2002-02-28T14:47:35.000000Z|jadams     |www.shop.com/
transaction_complete.htm   |
|2002-02-28T14:28:23.000000Z|ztaylor    |www.shop.com/home.htm
     |
|2002-02-28T14:44:13.000000Z|ztaylor    |www.shop.com/catalog.htm
     |
|2002-02-28T14:46:22.000000Z|ztaylor    |www.shop.com/prodinfo/cadillac.htm
     |
|2002-02-28T14:48:31.000000Z|ztaylor    |www.shop.com/purchase.htm
     |
|2002-02-28T14:50:01.000000Z|ztaylor    |www.shop.com/catalog.htm
     |
---------------------------+-----------
+----------------------------------------+
```

SIMPLE LOOKUP EXAMPLE

The URLs are not as descriptive as they could be, so for presentation purposes, the example substitutes the URL with a more descriptive string. The postprocessor script:

- contains only the postprocRow function, with no initialization or finalization function
- uses an outgoing schema that is identical to the incoming schema

The postprocessor script shown below uses an internal Perl hash lookup to map URL values to more user friendly page descriptions, and allows for pages not found in the hash.

```
# ---------- FILE: 01_lookup.pproc_in
my %easyUrlMap = (
  "home",                          "Home Page",
  "catalog",                       "Product Catalog",
  "prodinfo/infiniti",             "Car Info Page -- INFINITI",
  "prodinfo/cadillac",             "Car Info Page -- CADILLAC",
  "prodinfo/audi",                 "Car Info Page -- AUDI",
  "prodinfo/saturn",               "Car Info Page -- SATURN",
  "specials",                      "Special Offers Central",
  "specials/audi_europe",          "Special Offer--Purchase AUDI in Europe",
  "purchase",                      "CASH REGISTER",
  "transaction_complete",          "-SALE-"
);
sub postprocRow
  my ($response, $inputRow) = @_;
  # Create a copy of the input row, for output
  my $outputRow = {};
  $response->copyValuesForOutputSchema($inputRow, $outputRow);
  # Get the URL, prepare default friendly URL (which isn't very friendly)
  my $rawUrl = $inputRow->getColumnValue("url");
  my $resultUrl = "... $rawUrl ...";
  # Try to find a more friendly URL
  my $url2;
  if (($url2) = $rawUrl =~ /^www.shop.com\/(.*)\.htm$/) {
     my $url3 = $easyUrlMap{$url2};
     if (defined($url3)) {
       $resultUrl = $url3;
     }
  }
  # Replace the URL in the output row
  $outputRow->{"url"} = $resultUrl;
  # Add the modified row to the response
  $response->addRowData(rowdata => $outputRow);
# ---------- END OF FILE: 01_lookup.pproc_in
```

The following command runs the example:

```
echo "SELECT ts, user_name, url FROM test ORDER BY 2, 1 DURING ALL;" | \
 atquery lmshost:8072 --namespace=myNamespace.pproc_example -
 --postproc=01_lookup.pproc_in
```

The query results are illustrated below:

```
+---------------------------+-----------+---------------------------------+
|     ts                    | user_name |             url                 |
| (timestamp)               | (varchar) |           (varchar)             |
+---------------------------+-----------+---------------------------------+
|2002-02-28T14:23:57.000000Z|gwashington|Home Page                        |
|2002-02-28T14:37:15.000000Z|gwashington|Product Catalog                  |
|2002-02-28T14:39:23.000000Z|gwashington|Car Info Page -- INFINITI        |
|2002-02-28T14:49:27.000000Z|gwashington|... www.shop.com/other_page.htm .|
```

```
|2002-02-28T14:29:30.000000Z|jadams     |Home Page                          |
|2002-02-28T14:41:51.000000Z|jadams     |Special Offers Central             |
|2002-02-28T14:46:25.000000Z|jadams   |Special Offer--Purchase AUDI in
 Europe|
|2002-02-28T14:46:29.000000Z|jadams     |CASH REGISTER                      |
|2002-02-28T14:47:35.000000Z|jadams     |-SALE-                             |
|2002-02-28T14:28:23.000000Z|ztaylor    |Home Page                          |
|2002-02-28T14:44:13.000000Z|ztaylor    |Product Catalog                    |
|2002-02-28T14:46:22.000000Z|ztaylor    |Car Info Page -- CADILLAC          |
|2002-02-28T14:48:31.000000Z|ztaylor    |CASH REGISTER                      |
|2002-02-28T14:50:01.000000Z|ztaylor    |Product Catalog                    |
+---------------------------+-----------+-----------------------------------+
```

MORE COMPLEX AGGREGATION EXAMPLE

The second example uses a postprocessor to aggregate some per-page metrics directly from the query results. The metrics are “hits” on any given URL, and “unique users” that saw the given URL. This example also formats the results with more white space than would typically display in query results.

Additionally, this example puts within the query results both the query's incoming schema from the EDW and the output schema the postprocessor defines for the results.

The postprocessor script for this more complex aggregation example:

```
# ---------- FILE: 01_lookup.pproc_in
my %easyUrlMap = (
  "home",                       "Home Page",
  "catalog",                    "Product Catalog",
  "prodinfo/infiniti",          "Car Info Page -- INFINITI",
  "prodinfo/cadillac",          "Car Info Page -- CADILLAC",
  "prodinfo/audi",              "Car Info Page -- AUDI",
  "prodinfo/saturn",            "Car Info Page -- SATURN",
  "specials",                   "Special Offers Central",
  "specials/audi_europe",       "Special Offer--Purchase AUDI in Europe",
  "purchase",                   "CASH REGISTER",
  "transaction_complete",       "-SALE-"
);

sub mapUrl
   my ($rawUrl) = @_;
   # Prepare default friendly URL (which isn't very friendly)
   my $resultUrl = "... $rawUrl ...";
   # Try to find a more friendly URL
   my $url2;
   if (($url2) = $rawUrl =~ /^www.shop.com\/(.*)\.htm$/) {
       my $url3 = $easyUrlMap{$url2};
     if (defined($url3)) {
         $resultUrl = $url3;
       }
    }
   # Return the mapped URL
   return $resultUrl;

sub pushSchemaInfoIntoTable
   my ($response, $message, $schema) = @_;
   # For each element of the schema, push out the column name/value
   my $schemaElement;
   foreach $schemaElement (@{$schema}) {
       # Find this column's name and type
    my $columnName;
    my $columnType;
    ($columnName, $columnType) = $schemaElement =~ /^(.*):(.*)$/;
```

```
    # Prepare a row
    my $rowData = {
     "direction" => $message,
     "column_name" => $columnName,
     "column_type" => $columnType,
     };
     # Push the row
     $response->addRowData(rowdata => $rowData);
    }
    my %pageMetrics;


sub postprocInit
 my ($response) = @_;
 # Initialize aggregation metrics -- we MUST do this each time
 # postprocInit is called, because we can have multiple queries
 # running within one 'atquery' session, and we want fresh results
 # each time we run a query.
 %pageMetrics = ();
 # Set the output schema, which will be different from the input schema
 my $outputSchema = [
    "section:varchar",
    "direction:varchar",
    "column_name:varchar",
    "column_type:varchar",
    "page:varchar",
    "page_metric:varchar",
    "metric_value:int32",
 ];
 $response->setMetadata(schema => $outputSchema);
 # To help illustrate input and output schemas, let's print out the
 # input/output schemas in the result set itself
 my $emptyRow = {};
 $response->addRowData(rowdata => $emptyRow);
 my $sectionRow = { section => "input/output schema" };
 $response->addRowData(rowdata => $sectionRow);
 $response->addRowData(rowdata => $emptyRow);
 my @inputSchema = $response->getIncomingSchema();
 my @outputSchema2 = $response->getSchema();
 pushSchemaInfoIntoTable($response, "input", \@inputSchema);
 $response->addRowData(rowdata => $emptyRow);
 pushSchemaInfoIntoTable($response, "output", \@outputSchema2);
 $response->addRowData(rowdata => $emptyRow);

sub postprocRow
 my ($response, $inputRow) = @_;
 # Get relevant fields out of the row
 my $userName = $inputRow->getColumnValue("user_name");
 my $rawUrl = $inputRow->getColumnValue("url");
 # Translate URL to a more friendly value -- we assume
 # the return value is unique
 my $page = &mapUrl($rawUrl);
 # We're doing per-page aggregation, make sure we have space
 # for this page (url)
 if (!defined($pageMetrics{$page})) {
    $pageMetrics{$page} = {
       numHits => 0,
       uniqueUsers => {},
    };
 }
 # Now update the per-page aggregates
 $pageMetrics{$page}->{numHits}++;
```

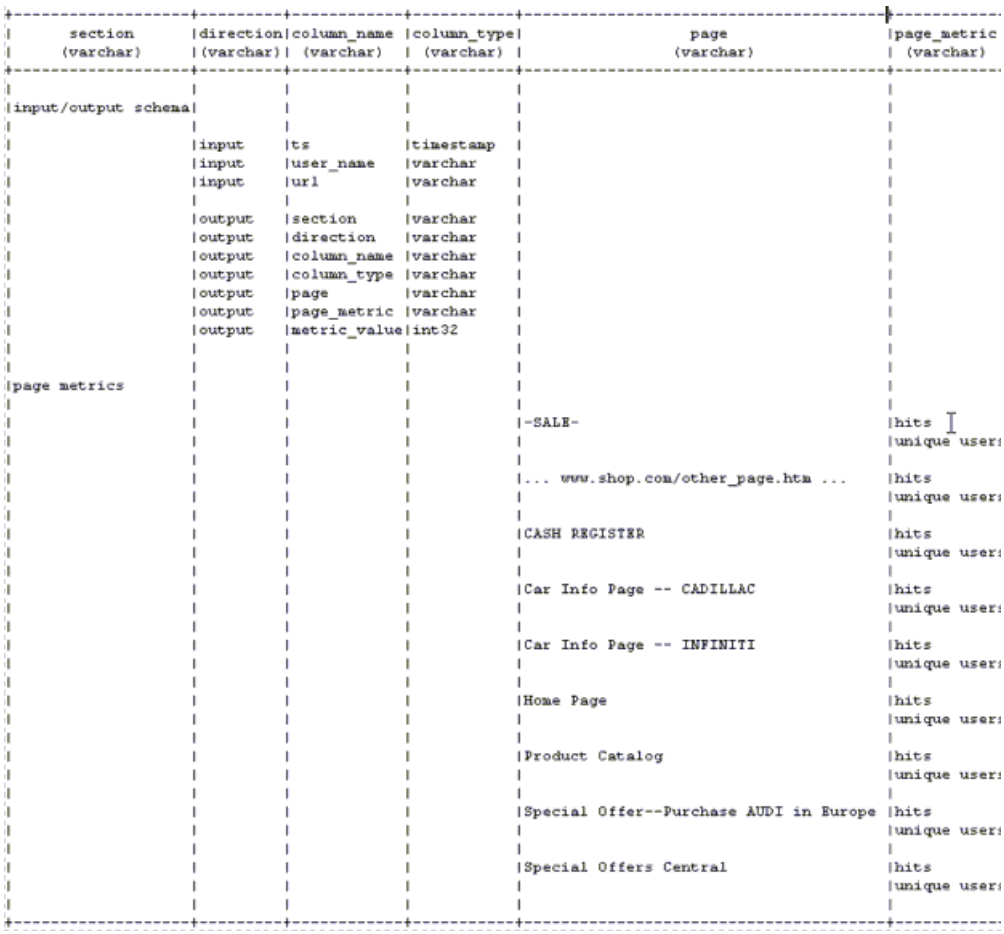
```
 $pageMetrics{$page}->{uniqueUsers}->{$userName}++;


sub postprocFinal
 my ($response) = @_;
 # Start a new section
 my $emptyRow = {};
 $response->addRowData(rowdata => $emptyRow);
 my $sectionRow = { section => "page metrics" };
 $response->addRowData(rowdata => $sectionRow);
 $response->addRowData(rowdata => $emptyRow);
 # Present each page and its metrics
 my $page;
 foreach $page (sort(keys(%pageMetrics))) {
    # Add a row for page hits
    my $row = {
       page => $page,
       page_metric => "hits",
       metric_value => $pageMetrics{$page}->{numHits}
    };
    $response->addRowData(rowdata => $row);
    # Add a row for number of unique users
    my @uniqueUsers = keys(%{$pageMetrics{$page}->{uniqueUsers}});
    my $row = {
       page_metric => "unique users",
       metric_value => ($#uniqueUsers + 1)
    };
    $response->addRowData(rowdata => $row);
    # Add empty row
    $response->addRowData(rowdata => $emptyRow);
 }
# ---------- END OF FILE: 02_aggregate.pproc_in
```

The following command runs the example:

```
echo "SELECT ts, user_name, url FROM test DURING ALL;" | \
      atquery lmshost:8072 --namespace=myNamespace.pproc_example -
      --postproc=02_aggregate.pproc_in
```

The query results are illustrated below:



section (varchar)	direction (varchar)	column_name (varchar)	column_type (varchar)	page (varchar)	page_metric (varchar)
input/output schema					
	input	ts	timestamp		
	input	user_name	varchar		
	input	url	varchar		
	output	section	varchar		
	output	direction	varchar		
	output	column_name	varchar		
	output	column_type	varchar		
	output	page	varchar		
	output	page_metric	varchar		
	output	metric_value	int32		
page metrics					
				-SALE-	hits
					unique user
				... www.shop.com/other_page.htm ...	hits
					unique user
				CASH REGISTER	hits
					unique user
				Car Info Page -- CADILLAC	hits
					unique user
				Car Info Page -- INFINITI	hits
					unique user
				Home Page	hits
					unique user
				Product Catalog	hits
					unique user
				Special Offer--Purchase AUDI in Europe	hits
					unique user
				Special Offers Central	hits
					unique user