# LanguageManual Transform

## Transform/Map-Reduce Syntax

Users can also plug in their own custom mappers and reducers in the data stream by using features natively supported in the Hive language. e.g. in order to run a custom mapper script - map_script - and a custom reducer script - reduce_script - the user can issue the following command which uses the TRANSFORM clause to embed the mapper and the reducer scripts.

By default, columns will be transformed to *STRING* and delimited by TAB before feeding to the user script; similarly, all NULL values will be converted to the literal string **\N** in order to differentiate NULL values from empty strings. The standard output of the user script will be treated as TAB-separated *STRING* columns, any cell containing only **\N** will be re-interpreted as a NULL, and then the resulting STRING column will be cast to the data type specified in the table declaration in the usual way. User scripts can output debug information to standard error which will be shown on the task detail page on hadoop. These defaults can be overridden with *ROW FORMAT ...*.

In windows, use "cmd /c your_script" instead of just "your_script"

Warning

It is your responsibility to sanitize any STRING columns prior to transformation. If your STRING column contains tabs, an identity transformer will not give you back what you started with! To help with this, see REGEXP_REPLACE and replace the tabs with some other character on their way into the TRANSFORM() call.

Warning

Formally, *MAP ...* and *REDUCE ...* are syntactic transformations of *SELECT TRANSFORM ( ... )*. In other words, they serve as comments or notes to the reader of the query. BEWARE: Use of these keywords may be **dangerous** as (e.g.) typing "REDUCE" does not force a reduce phase to occur and typing "MAP" does not force a new map phase!

Please also see Sort By / Cluster By / Distribute By and Larry Ogrodnek's blog post.

```
clusterBy: CLUSTER BY colName (',' colName)*
distributeBy: DISTRIBUTE BY colName (',' colName)*
sortBy: SORT BY colName (ASC | DESC)? (',' colName (ASC | DESC)?)*

rowFormat
  : ROW FORMAT
    (DELIMITED [FIELDS TERMINATED BY char]
               [COLLECTION ITEMS TERMINATED BY char]
               [MAP KEYS TERMINATED BY char]
               [ESCAPED BY char]
               [LINES SEPARATED BY char]
     |
     SERDE serde_name [WITH SERDEPROPERTIES
                            property_name=property_value,
                            property_name=property_value, ...])

outRowFormat : rowFormat
inRowFormat : rowFormat
outRecordReader : RECORDREADER className

query:
  FROM (
    FROM src
    MAP expression (',' expression)*
    (inRowFormat)?
    USING 'my_map_script'
    ( AS colName (',' colName)* )?
    (outRowFormat)? (outRecordReader)?
    ( clusterBy? | distributeBy? sortBy? ) src_alias
  )
  REDUCE expression (',' expression)*
    (inRowFormat)?
    USING 'my_reduce_script'
    ( AS colName (',' colName)* )?
    (outRowFormat)? (outRecordReader)?

  FROM (
    FROM src
    SELECT TRANSFORM '(' expression (',' expression)* ')'
    (inRowFormat)?
    USING 'my_map_script'
    ( AS colName (',' colName)* )?
    (outRowFormat)? (outRecordReader)?
    ( clusterBy? | distributeBy? sortBy? ) src_alias
  )
  SELECT TRANSFORM '(' expression (',' expression)* ')'
    (inRowFormat)?
    USING 'my_reduce_script'
    ( AS colName (',' colName)* )?
    (outRowFormat)? (outRecordReader)?
```

### SQL Standard Based Authorization Disallows TRANSFORM

The TRANSFORM clause is disallowed when SQL standard based authorization is configured in Hive 0.13.0 and later releases (HIVE-6415).

### TRANSFORM Examples

Example #1:

```
  FROM (
    FROM pv_users
    MAP pv_users.userid, pv_users.date
    USING 'map_script'
    AS dt, uid
    CLUSTER BY dt) map_output
  INSERT OVERWRITE TABLE pv_users_reduced
    REDUCE map_output.dt, map_output.uid
    USING 'reduce_script'
    AS date, count;
  FROM (
    FROM pv_users
    SELECT TRANSFORM(pv_users.userid, pv_users.date)
    USING 'map_script'
    AS dt, uid
    CLUSTER BY dt) map_output
  INSERT OVERWRITE TABLE pv_users_reduced
    SELECT TRANSFORM(map_output.dt, map_output.uid)
    USING 'reduce_script'
    AS date, count;
```

Example #2

```
  FROM (
    FROM src
    SELECT TRANSFORM(src.key, src.value) ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.
TypedBytesSerDe'
    USING '/bin/cat'
    AS (tkey, tvalue) ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.TypedBytesSerDe'
    RECORDREADER 'org.apache.hadoop.hive.contrib.util.typedbytes.TypedBytesRecordReader'
  ) tmap
  INSERT OVERWRITE TABLE dest1 SELECT tkey, tvalue
```

## Schema-less Map-reduce Scripts

If there is no *AS* clause after *USING my_script*, Hive assumes that the output of the script contains 2 parts: key which is before the first tab, and value which is the rest after the first tab. Note that this is different from specifying *AS key, value* because in that case, value will only contain the portion between the first tab and the second tab if there are multiple tabs.

Note that we can directly do *CLUSTER BY key* without specifying the output schema of the scripts.

```
  FROM (
    FROM pv_users
    MAP pv_users.userid, pv_users.date
    USING 'map_script'
    CLUSTER BY key) map_output
  INSERT OVERWRITE TABLE pv_users_reduced
    REDUCE map_output.key, map_output.value
    USING 'reduce_script'
    AS date, count;
```

## Typing the output of TRANSFORM

The output fields from a script are typed as strings by default; for example in

```
  SELECT TRANSFORM(stuff)
  USING 'script'
  AS thing1, thing2
```

They can be immediately casted with the syntax:

```
SELECT TRANSFORM(stuff)
USING 'script'
AS (thing1 INT, thing2 INT)
```