Comparison with detect duplicates

Another option is to feed the source data into a Table Comparison transform and check its option "input contains duplicates".

Table Comparison receives a first row and tries to find if a row with this EmployeeID exists in the target already. If it doesn't - because the target table is empty at that time - the transform marks the row as Insert and sends it downstream. The table loader will generate an insert statement for this and load it into the table a few milliseconds later. What "input contains duplicate" does is updating the Table Comparison internal cache too with that row. So the next row with the same primary key will be compared again, but this time with the cached row and get marked as Update to loaded via an update statement into the target.

This approach certainly has its pros and cons. First, the data does not need to come in sorted as the transform will remember if such a row arrived already before. On the other hand, what source row will we see after the dataflow? Well, the latest row Table Comparison got for each group as this will be the one with the final update. Therefore you have to sort the data anyway, to ensure which source row to be the last.

One true advantage is, the source no longer requires an unique key.

The downsides are that Table Comparison will cache the data, so requires memory but at least will not stage the data like in the self join example. Each row will flow immediately from source to target. And it will issue multiple update statements against the same target row.

(In case you wonder why there are just two rows with EmployeeID=3, two of them are identical hence outputted just once by TC.)

Therefore, if the dataflow requires a Table Comparison transform anyway as there are rows in the target table from the last load that will get updated now, using this flag is very convenient, especially if the transform runs in "cache comparison table" mode. Still, multiple updates could happen.