Topofminds

Top of Minds Report series
Removing duplicates using Informatica PowerCenter

Introduction

There is more than one way to skin a cat, and there are four ways to choose one distinct row of several possibilities in an Informatica PowerCenter mapping; Rank, Sort, Aggregate and the sorted Expression-filter technique. Each method has its own pros and cons, and a good developer should know when to use which technique. This white paper detail four different methods and discuss when each one of them should be used. Ideally, you want to have more than one possible solution for each scenario you encounter, and determine which ultimate technique to use by comparing run time, memory usage and other parameters of importance to your configuration.

Target Audience

This white paper is directed at Informatica developers who want to write more efficient code that will execute faster, use fewer resources and cost less to maintain.

Summary

Knowing when to use which transformation and how transformations impact memory, collection points, cache size and strain on underlying databases is key to being a successful developer and creating long-lasting value for your employer or customer. The benefits of using the correct transformation in each mapping include, but are not limited to:

- Increased operational stability
- More efficient use of hardware resources
- Higher data quality by selecting the correct row to move forward at all times
- Better performance and faster code

About the author

Therese Ahlstam is a senior specialist at Top of Minds AB. Therese is specialized in data warehousing with specific focus on large data volumes, efficient processing and best practice code. She has extensive experience in Bank & Finance, Retail and Gaming.

Comments on this white paper can be sent to: therese.ahlstam@topofminds.se.

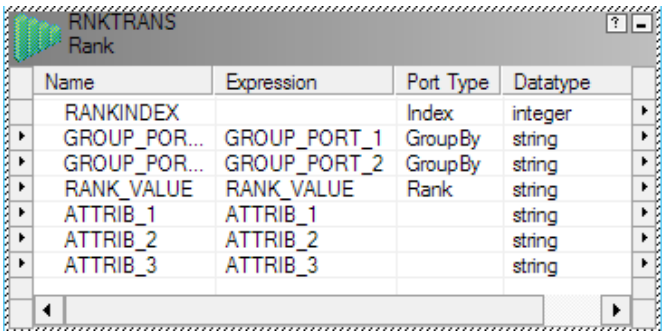
About Top of Minds

Top of Minds is a specialized company that offers services in the data warehousing and business intelligence area. We are premier partner in the Nordic countries on the data warehouse development using an agile approach where we use Data Vault to increase the benefits of the projects we participate in. We are a company with great focus on competence, our expertise and our clients' expertise and we are constantly working to spread our knowledge.

Visit www.topofminds.se for further details.

Method - Rank transformation

The rank transformation is used for returning the top- or bottom-ranked rows in a data set. It differs from an aggregate transformation in that it returns entire rows, not just the evaluated max/min-values in a single port. For instance, you can use Rank to pass on only the top 10 selling products, or using group-ports to pass on the top 10 by region. Use rank to filter duplicates when you want to select one row of several possibilities, where certain values have precedence over others.



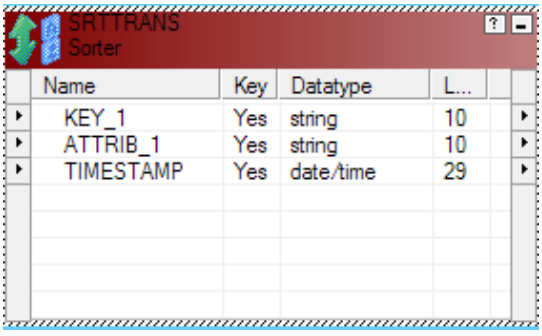
Name	Expression	Port Type	Datatype
RANKINDEX		Index	integer
GROUP_POR...	GROUP_PORT_1	GroupBy	string
GROUP_POR...	GROUP_PORT_2	GroupBy	string
RANK_VALUE	RANK_VALUE	Rank	string
ATTRIB_1	ATTRIB_1		string
ATTRIB_2	ATTRIB_2		string
ATTRIB_3	ATTRIB_3		string

You can only use one port for evaluating the rank and choose to take the top or bottom row/s, and it's important to remember that if more than one row meets the same criteria, they will have the same rank. For instance, if you rank rows grouped on natural keys by their create-timestamp and return the top row you would get the latest row for each key – but if two rows have the same create-timestamp you will still have duplicates after the rank.

The Rank transformation does not have an option for using Sorted Input, unlike many other active transformations that require the use of cache-files. It will hold the number of rows to be passed on in memory and evaluate each row as it arrives, replacing rows in cache when conditions apply. When all rows have arrived the Rank is finished and the correct row/s matching your criteria is passed on. This makes Rank a collection point, forcing all rows to be read and ranked before write to target can start. When working with large data volumes one always strives to stream rows from source to target as much as possible and avoid collection points and large caches. While rank does not create a large cache (only the row/s that will be passed on are cached) it does create a collection point. If the mapping handles large data volumes and there are no other collection points, rank might not be the optimal choice since it would hinder a continuous stream of data from source to target. When evaluating its performance against other options it should be compared to an unsorted aggregate or a sort-transformation, since it does not require data to be sorted in advance.

Method - Sort transformation

One of the simplest but certainly not the most efficient way to create distinct rows is to use a Sort transformation and check the 'Distinct' option in Properties. If you have to sort data for some other reason and the number of ports are reasonably low (the more ports, the larger the cache file) this might be a viable option. However, it will only work for removing duplicates if all values across the entire row are identical. Also, since sorting distinct really means, "Sort all ports ascending and remove duplicates" the memory and disk cache required will be larger than for non-distinct sorts.



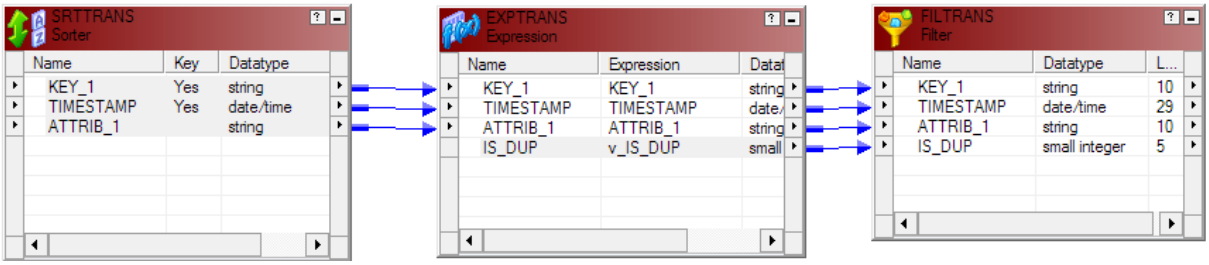
Name	Key	Datatype	L...
KEY_1	Yes	string	10
ATTRIB_1	Yes	string	10
TIMESTAMP	Yes	date/time	29

There is a much more efficient way of removing duplicates using sorted data as input than to sort all ports; the expression-filter technique which will be covered next. I have yet to find a situation where using a distinct sort is preferable, so this section serves more as a "what not to do" than anything else. If pressed, I would say this may be used when the number of rows is small with few columns, and I know that any duplicates are identical across all columns. The loss in memory/cache-performance

Copyright © 2015 Top of Minds. All rights reserved

is then minimal, and you will save yourself about 30 minutes of coding compared to the expression-filter technique.

Method - Expression filter transformations



If your data is sorted properly, you do not have to spend any resources at all to remove your duplicates. No memory usage or caching of data is required and no collection points are required, which would interrupt a continuous stream of data from source to target. It is arguably the most efficient and least time-consuming of all options in this white paper, and there are virtually no limitations as to how complex the logic that evaluates a duplicate can be. The one caveat is – it needs a sorted data input.

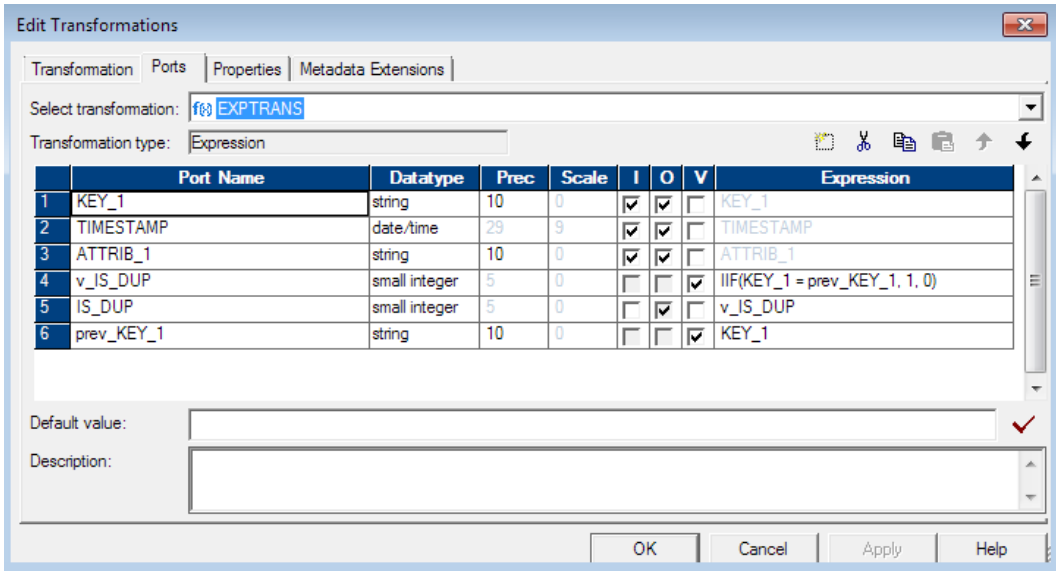
This solution takes advantage of using variables within an expression to keep track of previous rows. The current row is evaluated against values from previous rows, and if it is a duplicate it is marked as such in a separate port. The next transformation in the mapping is a filter, which only passes on the rows that are not marked as duplicates.

You can sort your data in a Sort transformation or add an order by-clause in your Source Qualifier – which one you choose depends on your configuration. If you're not sure, try both options and evaluate which one fits best. Regardless of which approach you choose, sort your data first by the group columns, followed by the column/s used for evaluating which row to choose. Your object is to have the best row first in a group, pass this on and then mark all rows coming after as duplicates.

In the example above, rows are sorted by key (ascending) and by timestamp (descending), thus ensuring that the latest row is listed first by each key. In the following expression transformation, variables keep track of the previous row values:

In Informatica, ports within a transformation are evaluated according to a set order of operations; first, IN-ports, then Variable ports and lastly OUT-ports.

Copyright © 2015 Top of Minds. All rights reserved

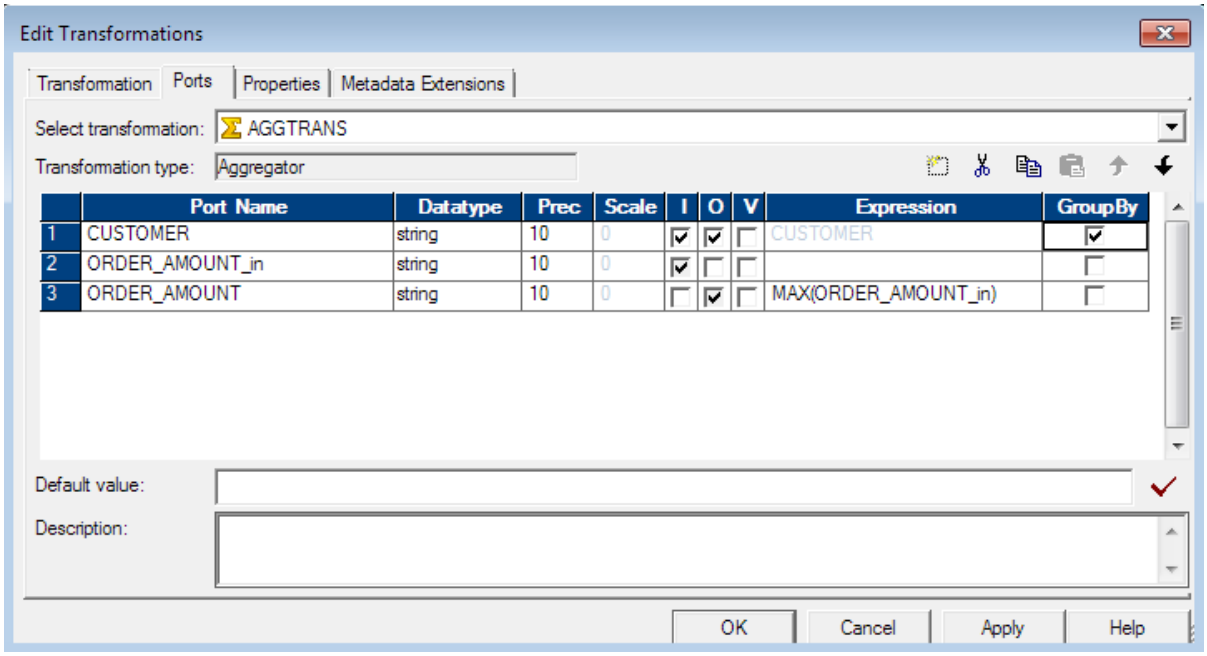


Here we take advantage of this by having the v_IS_DUP variable port as nr 4, the first port to be evaluated except for the IN-ports. It compares the current rows KEY_1 port with that from the previous row. If they are identical, then this is a duplicate row and should be marked as such. If not, it is a new key and not a duplicate. By sorting our data first by key and then by timestamp we ensure that the first row we encounter for each new key is the latest one, which is the one we want in this case. The next port is simply an OUT-port for our v_IS_DUP variable, passing it on to the next transformation. The last port is the variable for storing the previous rows key value. Since variable ports are processed from the top down, this is the last variable to get updated and more importantly it will still hold its previous value by the time we evaluate the v_IS_DUP-variable. Only after v_IS_DUP is calculated will we process this port, setting it to the current rows key value, preserving it for the next rows evaluation.
The only real limitation to this technique of removing duplicates and choosing one possible row of several is your sorting. The following is an example of solving more complex logic using this technique:
You want to get the latest active subscription per customer. An active subscription may have status OPEN or REOPENED, inactive subscriptions have status CLOSED. An activation_date shows when the subscription was activated. The business rules state that if there are no active subscriptions for a customer, then the latest inactive one should be displayed.
First of all, we want one subscription per customer so our first sort key is customer. Next, we want to choose OPEN or REOPENED rows over CLOSED ones. In this case, by sorting status descending both OPEN and REOPENED will appear before CLOSED. Alternatively, I could have assigned numerical values in a previous expression and sorted by those to ensure they end up in the right order. Lastly, I want the latest subscription so our final port to sort is the activation_date, descending to get the latest port first.
Remember – a Sort transformation is a collection point in your mapping and will prohibit continuous streaming from source to target. If your mapping will process a large number of rows and you want to limit the use of memory/disk you might want to consider overriding the Source Qualifier query, thus pushing the sort to your database and avoiding collection points in your mapping.

Method - Aggregate transformation

The Aggregate transformation differs from the others in this white paper in that it operates on single ports, not on entire rows. Using the aggregate transformation you can concatenate several rows into one, choosing the correct value for each port using min(), max(), sum(), first() and last() functions, amongst others.

Copyright © 2015 Top of Minds. All rights reserved



If I want the row with the latest update date for each customer and pass forward that row in its entirety, the Aggregate transformation is not the way to go. By choosing customer as my group port, and max(update_date) I would get the latest update date passed forward, but not necessarily the other attributes from that row. For each port I would have to choose some sort of aggregate function to determine which value from the rows available I should pass forward, independent of other ports. If I don't choose anything and leave the other ports as standard IN/OUT I have no control over which values will be chosen; it could be the first, the last or anyone in between. There is no guarantee you will get the values from the row with max(update_date).

However, if I want the highest order amount by customer and do not need any other information from the row that contains the highest order amount the aggregate may well be the way forward. I would group by customer and define a new OUT-port to pass the max(order_amount) in each group on to the next transformation.
The aggregate does not perform well if it is unsorted, and you should always try to check the Sorted Input checkbox in Properties. If the input is sorted on the group-ports the cache required will be smaller, but the aggregate will always be a collection point. Thus, adding another collection point via a Sort transformation should only improve performance, but as with all things performance-related you should test having a Sort transformation, adding the sort to the source qualifier if possible and running the aggregate with unsorted input, then choose the one that fits your environment best.

Copyright © 2015 Top of Minds. All rights reserved

Conclusion

Below follows a summary of the different types of duplicate removal in table form for easy reference.

Type	Use for/when	Collection Point / Interrupts continuous stream	Guaranteed unique key	Requires sorted input
Rank	Choose row based on a ranked port	Yes	No	No
Sort	Only small volumes and only where all ports are equal, not just keys	Yes	No	No
Expression - Filter	Best Practice! Should always be considered	No	Yes	Yes. For large volumes, push down sort in source qualifier to ensure continuous stream
Aggregate	Aggregated port values by key ports (eg min/max/first)	Yes	Yes	Yes and No. Performance will drastically increase and resource usage decreases if input is sorted.

Copyright © 2015 Top of Minds. All rights reserved