Topofminds

Top of Minds Report series

Agile Data Warehousing - Write generic code to limit codebase and gain performance in Informatica PowerCenter

Recommended reading

For deeper insights in Data Vault modeling techniques using Informatica PowerCenter see other publications at http://topofminds.se/wp/aktuellt/publicerat/. One example that can be of good use to get a more thorough picture of Data Vault modelling in Informatica PowerCenter is ToM Report Series on Data vault – Using Informatica PowerCenter to automate your Data vault. Also, read the whitepaper about hyper agile design pattern that takes generic coding to an extreme; ToM Report Series on Information Integration - Hyper Agile Design Pattern.

Background

Ever changing business needs and requirements are forcing BI-projects to be more agile than ever before. The value of "seeing the data" can't be emphasized enough in a BI-project which is why too rigid implementation processes and long delivery sprints should be avoided. Data Vault is one of the most flexible modelling techniques that is used today, much thanks to its abilities to cope and adapt with fast changing realities and increasing needs to extend existing data models.

When working with Data Vault or other ensemble modelling techniques the different building blocks; Hubs, Links and to some extent Satellites are well defined and all share a common structure that can be used to build generic code. With the use of generic code, the code base can be held to a minimum and hence the time to value can be shortened. One should be aware that one downside with generic code can be a greater complexity and reduced data load speed if you aren't careful in the design and implementation.

This report will give the reader hands on design examples in Informatica PowerCenter, to be used for writing generic, agile code without losing performance in the data load. Because of its suitability Data Vault is chosen to exemplify the performance gain.

Target audience

This paper turns to people working with ETL design/implementation and data warehouse modelling.

About the author

Daniela Björkbacke is a senior specialist at Top of Minds with a long experience from projects in different business areas, ranging from Retail to Logistics and Banking. She has deep knowledge in data modeling and architecture on a higher level as well as design and implementation of ETL-mappings and workflows on a more detailed level.

Comments on this white paper can be sent to
daniela.bjorkbacke@topofminds.se
or
Twitter @bjorkbacke

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

From Source to Target

The coming sections will briefly describe how the source and target should/could be set up in a generic environment. The source section covers how to parameterize the extraction and how the load can be steered dynamically depending on the number of keys in the table. To compare different design approaches and the performance, the target section in this whitepaper, is split up in three scenarios. Each of the scenarios describes an implementation of how to write to generic targets with varying load performance.

The examples are taken from an environment with Informatica PowerCenter and Oracle 11.

Generic sources

When the design of the data warehouse is to be done, think an extra time of the formats. If the hubs, links and satellites can be designed with a common interface, a lot of the programs can be generalized. For instance think about the data types of the keys, the name standard etc.

A source in Informatica PowerCenter can be overridden to a large extent. The number of keys in the source definition does not have to match the physical source, BUT the outgoing ports in the SQL-qualifier and the forthcoming objects are fixed both when it comes to number of ports and data type.

One way of overcome these limitations is to create a generic source or SQL-override with a source that has sufficient number of keys and attributes to cover all types of satellites, links or other components. In the example below, Figure 1, an SQL-override shows how to extract data from a link with unknown number of business keys, numbered 'BSN_KEY_1' to 'BSN_KEY_5'.

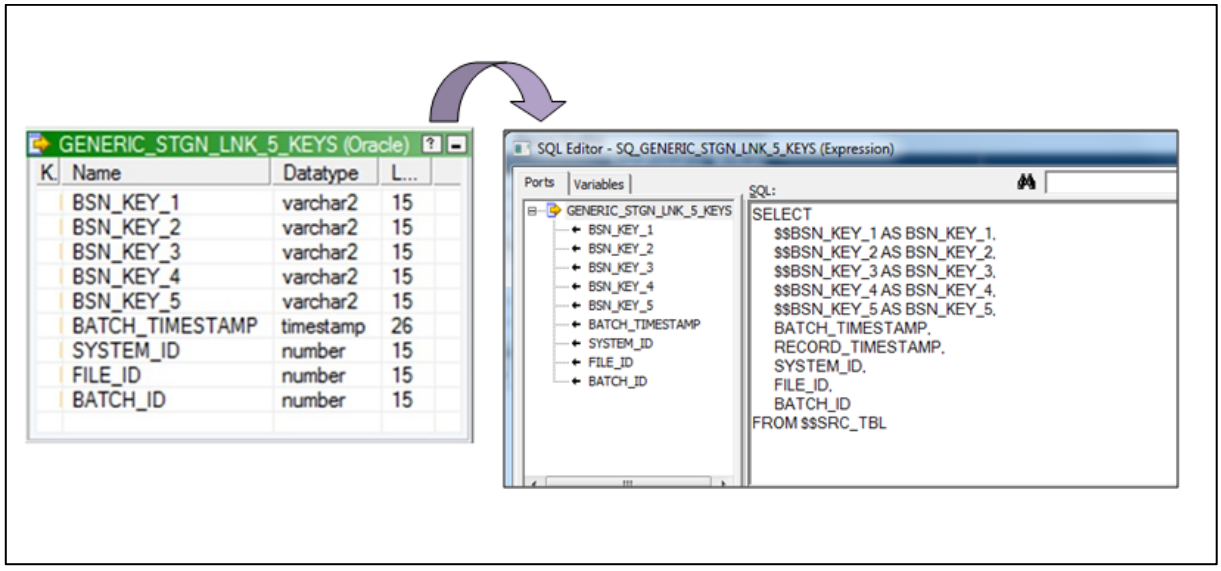

Figure 1: Generic source, override example. Source object to the left and SQL-override to the right.

As the figure shows, the SQL-override is completely generic and the mapping comes alive first when the parameters are set in the parameter file. Here it is important to remember to set the unused keys to a default string alternatively set an initial value in the parameter wizard, to ensure that the dynamic sql-statement that is generated copes with an arbitrary number of keys. See Figure 2 below for the two alternatives.

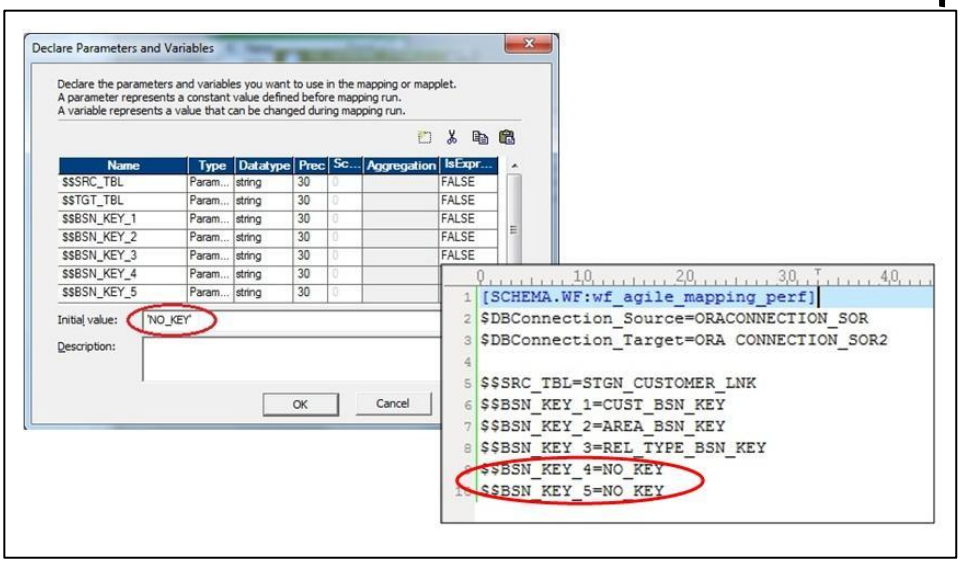

Figure 2: Initial values of the arbitrary number of keys in a generic source.

In the following steps of the mapping, the default string 'NO_KEY' (as seen in the example, Figure 2) can be used to determine how many keys are used at runtime and hence decide which generic target to use. One example of how the keys can be counted is by using DECODE statement to add all keys that match the default key string to get a final sum of keys, see example in Figure 3 below.

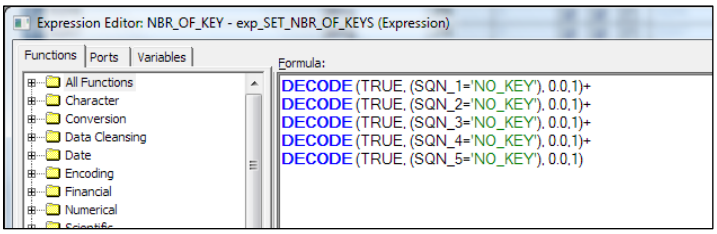

Figure 3: Determine the number of keys in an expression

The number of keys can be used in a router or in multiple filter transformations to route the data stream to the corresponding target. See Figure 4 below.

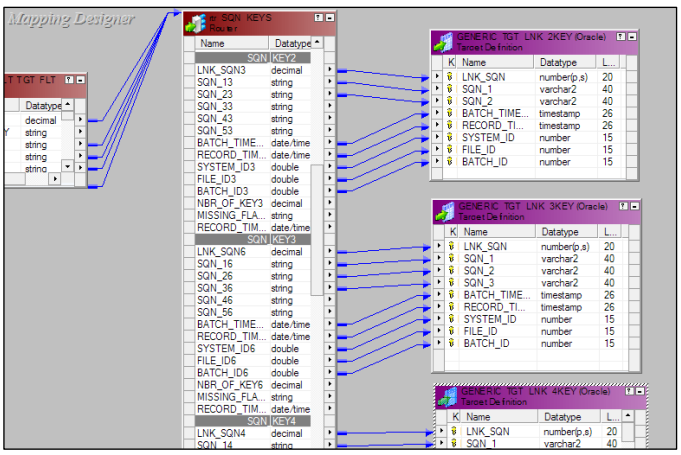

Figure 4: Router is used to determine which generic target to use and which ports to connect.

Generic targets

If the names of the keys in the generic target representation in Informatica PowerCenter don't match the key names in the actual target table, which usually is a challenge with generic mappings, the insert statement needs to be overwritten.

In the following sections, three scenarios will be presented where the override issue is solved, with different performance as result. The scenarios range from poor performance to high, where the least performing solution (scenario 1) is a possible design but presented in this whitepaper merely as a baseline for the other scenarios and should not be considered a recommendation.

The scenarios in this whitepaper cover a solution with inserts only. That is, the design is set up in a way that makes sure that no updates of old records have to be carried out. The reason for this is the performance; inserts are faster than updates. So if updates can be omitted they should!

The three solutions cover:

a. Update override with Data driven load and "Update transformation"
b. Update override without "Update transformation"
c. Write to target view

Scenario1) Update override with Data driven load and "Update transformation"

Informatica PowerCenter does not support "insert overrides", but there is a way to overcome the shortage by using "update overrides" instead. With update overrides table and column names can be set dynamically.

To trigger the insert/update override statement of the target, the mapping in this scenario has an update transformation with the update strategy 'DD_UPDATE' set, see Figure 5 below:

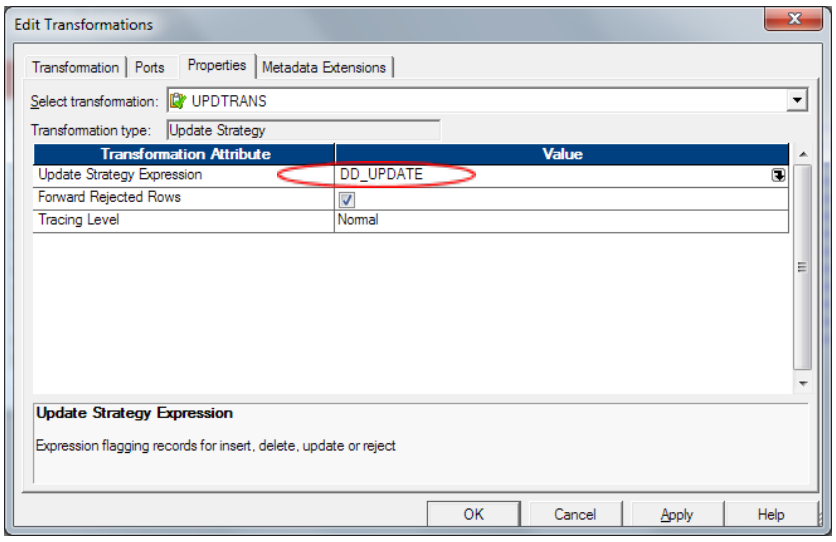

Figure 5: Update transformation is set to 'DD_UPDATE'

As stated above, there is no "Insert override" option, but by setting the update strategy to 'DD_UPDATE', the "Update override" section will be activated and the insert/update statement will be generated dynamically.

See the figures below for activating the “Update override” option; both steps are necessary for Scenario 1!

a. Set the load to Data driven (workflow designer, session level) (Figure 6)
b. Override the target table (Target definition, mapping designer) (Figure 7) with a parameterized Insert statement (Figure 8)

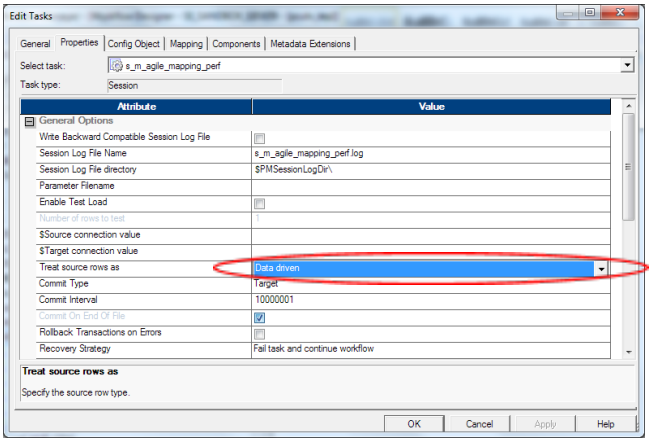

Figure 6: Set session to be Data driven

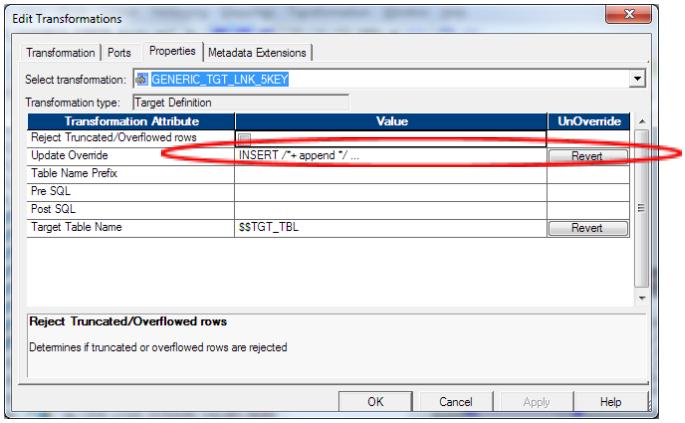

Figure 7: Write the update override at the target

See example of an update override, which in-fact is an insert!

```
INSERT /*+ append */
INTO $$TGT_TBL
(
    $$LNK_SQN, $$SQN_1, $$SQN_2, $$SQN_3, $$SQN_4, $$SQN_5,
    BATCH_TIMESTAMP, RECORD_TIMESTAMP,
    SYSTEM_ID, FILE_ID, BATCH_ID
)
VALUES (
    :TU.LNK_SQN, :TU.SQN_1, :TU.SQN_2, :TU.SQN_3, :TU.SQN_4, :TU.SQN_5,
    :TU.BATCH_TIMESTAMP, :TU.RECORD_TIMESTAMP,
    :TU.SYSTEM_ID, :TU.FILE_ID, :TU.BATCH_ID
)
```

Figure 8: Generic target update override

Scenario 2) Update override without "Update transformation"

The mapping doesn't have any Update transformation as in the previous scenario, but is instead set to "Update" at session level to trigger the update override section of the target, see Figure 9 below. By setting the load type to "Update" instead of "Data Driven" as in Scenario 1, there is not necessary to evaluate each row in the load, instead one generic approach is used for the whole batch, which is more efficient.

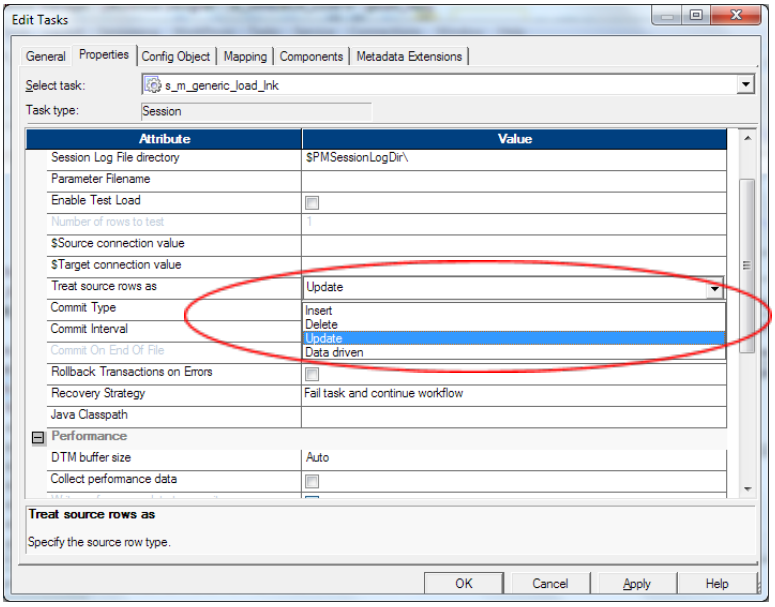

Figure 9: Set session to treat each row as Update

a. Set "Treat source rows as" to "Update" in the Workflow designer at session level.
b. Set up the target update strategy by ticking the "Insert" and "Update else Insert" and make sure nothing else is enabled in the target settings, see Figure 10 below.
c. Override the target table as in Scenario1, Figure 8 above.

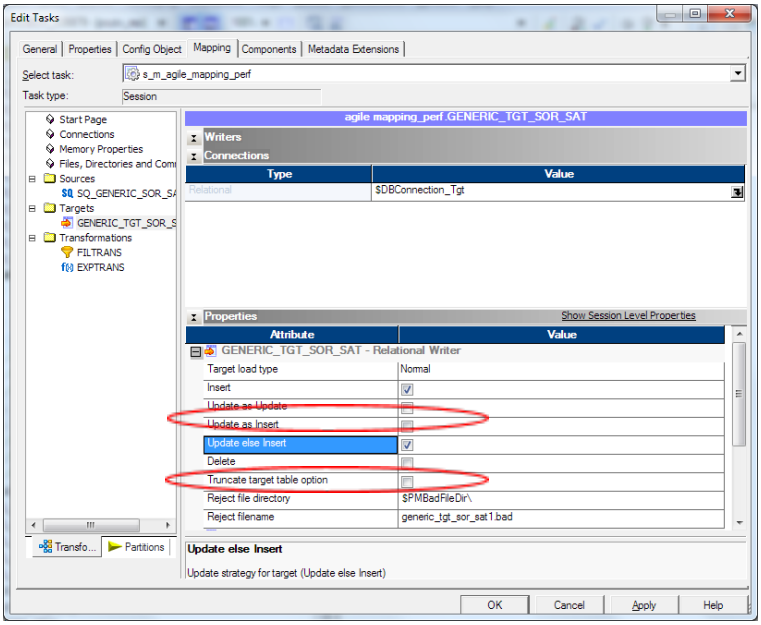

Figure 10: Settings for the load

Scenario 3) Write to target view

Create a view of all your physical target tables in the database with a common look of the keys that matches your generic target definition in Informatica PowerCenter. See the example below, Figure 11 where the target table SOR_CUSTOMER_LNK is used to make the view V_SOR_CUSTOMER_LNK with the same number and name of the keys as the target definition in Informatica:

```
CREATE OR REPLACE FORCE VIEW SCHEMA.V_SOR_CUSTOMER_LNK
(
   LNK_SQN,
   SQN_1,
   SQN_2,
   SQN_3,
   BATCH_TIMESTAMP,
   RECORD_TIMESTAMP,
   SYSTEM_ID,
   FILE_ID,
   BATCH_ID
)
AS
   SELECT "CUSTOMER_LNK_SQN",
          "CUST_SQN",
          "AREA_SQN",
          "REL_TYPE_SQN",
          "BATCH_TIMESTAMP",
          "RECORD_TIMESTAMP",
          "SYSTEM_ID",
          "FILE_ID",
          "BATCH_ID"
     FROM SCHEMA.SOR_CUSTOMER_LNK;
```

GENERIC_TGT_LNK_3KEY (Oracle)

K.	Name	Datatype
	LNK_SQN	number(p,s)
	SQN_1	varchar2
	SQN_2	varchar2
	SQN_3	varchar2
	BATCH_TIMESTAMP	timestamp
	RECORD_TIMESTAMP	timestamp
	SYSTEM_ID	number
	FILE_ID	number
	BATCH_ID	number

Figure 11: View that matches the Informatica target definition

In the parameter file, set up the connection parameters and override the target view name. Below follows an example how the parameter file may look like in the SOR_CUSTOMER_LNK example:

```
1 [SCHEMA.WF:wf_agile_mapping_perf]
2
3 $DBConnection_Src=ORACONNECTION_SOR
4 $DBConnection_Tgt=ORACONNECTION_SOR2
5
6 $$TGT_TBL =V_SOR_CUSTOMER_LNK
```

Figure 12: Parameter file with overridden target and key names

Remove the Update Strategy Transformations from the mapping if there are any (as shown in Scenario 2) and set the Target load type at session level to “Insert”, see Figure 13:

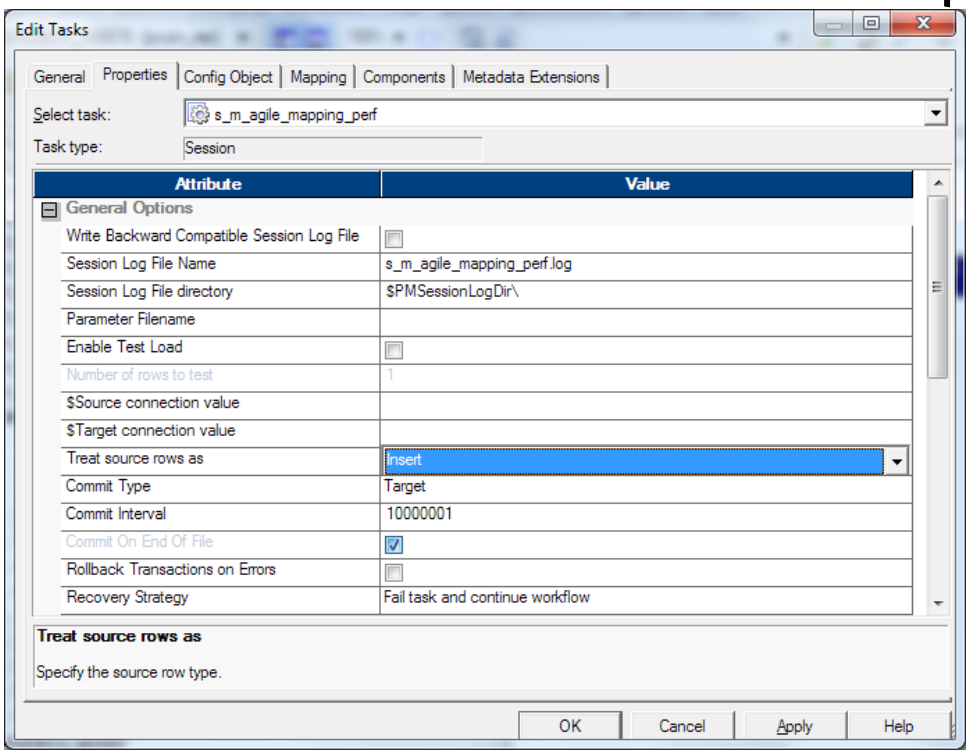

Figure 13: Set session to treat each row as Insert

Set the target update strategy in the session to “Insert “and “Update as Insert” as shown in Figure 14.

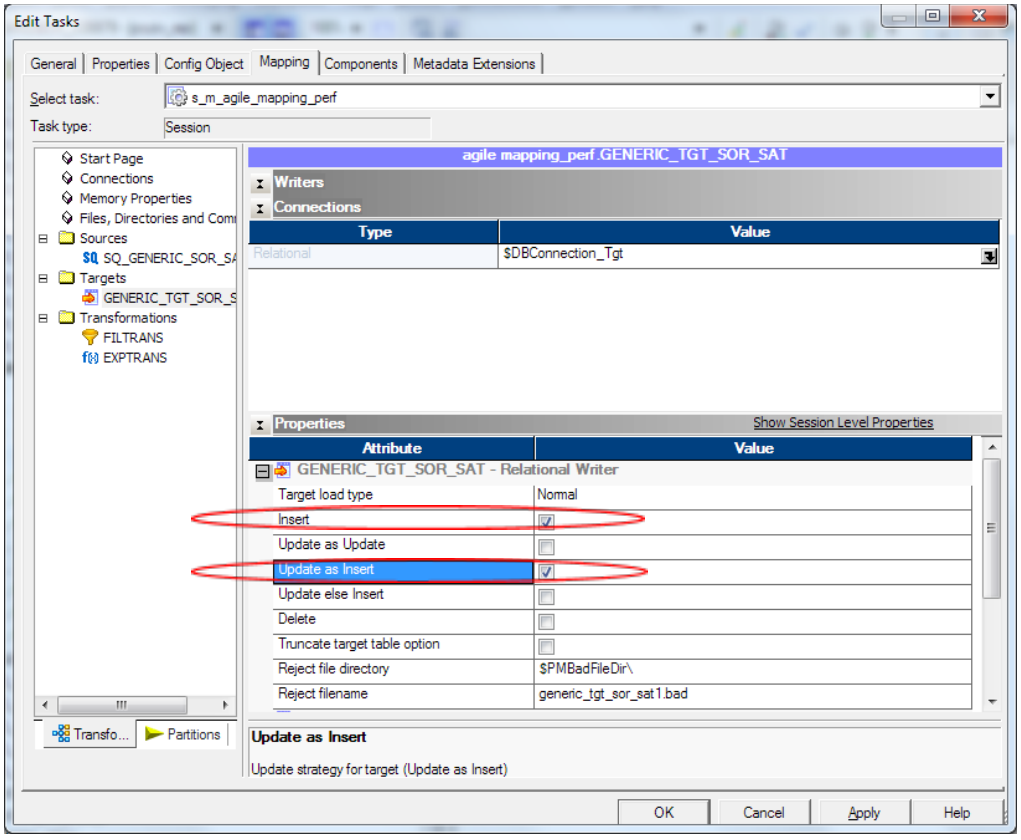

Figure 14: Set load parameters

Run performance

When the three scenarios presented above were tested, very different performance results were found. The test was carried out with the same data set for all three scenarios, 3 483 562 records.

As a benchmark for the data load, the writing was turned off, just to measure the reading throughput. A speed of 87 000 rows/sec was measured.

Scenario 1 - Update override with Data driven load and "Update transformation"

Because of a very low throughput, less than 1000 rows/sec, the data load was aborted after 20 minutes.

Result: ~ 1 000 rows/sec

Scenario 2 - Update override without "Update transformation"

The data load was stopped after 30 minutes, but a general throughput could be read out of the test, read and write with approximately 1300 rows/sec.
Result: ~ 1 300 rows/sec

Source/Target Statistics

Transformation Name	Node	Applied Rows	Affected Rows	Rejected Rows	Throughput (Rows/Sec)	Throughput (Bytes/Sec)	Bytes
GENERIC_TGT_...	node01_de...	2042292	2042292	0	1258	1134716	1842147384
SQ_GENERIC_S...	node01_de...	2049900	2049900	0	1263	6163440	100035120...

Figure 15: Throughput for Scenario2

Scenario 3 - Write to target view

The entire load was finished in less than two minutes, with a throughput of nearly 37 000 rows/sec.
Result: ~ 37 000 rows/sec

Source/Target Statistics

Transformation Name	Node	Applied Rows	Affected Rows	Rejected Rows	Throughput (Rows/Sec)	Throughput (Bytes/Sec)	Bytes
GENERIC_TGT_...	node01_de...	3483562	3483562	0	36670	33076340	3142172924
SQ_GENERIC_S...	node01_de...	3483562	3483562	0	36670	178949600	169997825...

Figure 16: Throughput for Scenario3

Performance evaluation

The first scenario used 'Data Driven' approach, in which Informatica PowerCenter uses single row insert, which implies that every row is evaluated and treated separately, and thus causing a longer load time. Single row processing should be avoided if possible!

In the second scenario, the load is set to 'Insert' which means that each row does not have to be evaluated, and hence enables a multiple row processing. That is why the performance is better. The third scenario, where a view is used, an even more effective load is achieved because array insert is applied.

Conclusion

Generic code can limit the code base in a project. Future changes can be carried out fast and safely because changes only have to be done in one place to affect all instances at once. But generic should be used with caution! The code will most probably be more complex and might cause slower loads if the design is carelessly done. Though, with a thought through DW and ETL-design with conformed structures and well-defined name standard, much time and money can be saved in development costs as well as load times.

The difference between various generic loading strategies can be a matter of hours in loading time! With the strategy of writing to generic views the load can be up to almost 30 times faster than writing to generic targets with an Insert override. Writing to views implies that more objects need to be created and maintained in the database, but if the documentation and name standard is properly done this is a minor dilemma and should not discourage from using this approach.

If you found this report interesting and want to hear more, please contact Top of Minds.