

INFORMATICA

Best practices for performances

Version - status:

|  |  |
| --- | --- |
| **Title :** | Best practices for performances |
| **Responsible of the content :** | *Stéphane Dulcieu* |
| **Document reference :** |  |
| **File name :** |  |

|  |  |  |
| --- | --- | --- |
| Document elaboration | | |
| **Entity/ Role** | **Date** | **Name** |
| RESG/GTS/MDB | 1/12/2016 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Enhancements and updates done in the document | | | |
| **Version** | **Date** | **Name** | **Modifications done** |
| V0.6 |  |  |  |
| V0.8 / V0.9 |  |  |  |
| V0.10 |  |  |  |

|  |  |  |
| --- | --- | --- |
| Validation of the document | | |
| **Organization / Entity** | **Date** | **Name** |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Distribution list of the document | | |
| **Organization / Entity** | **Date** | **Name** |
|  |  |  |

|  |
| --- |
| Comments |
|  |

Content

[1 - Introduction 5](#_Toc473632253)

[2 - Design 6](#_Toc473632254)

[2.1 Filter expressions 6](#_Toc473632255)

[2.1.1 Evaluation upstream of filter conditions 6](#_Toc473632256)

[2.2 Expression transformation 6](#_Toc473632257)

[2.2.1 Remove "DEFAULT" values 6](#_Toc473632258)

[2.2.2 Variable Ports are "slower" than Output Expressions. 7](#_Toc473632259)

[2.2.3 Datatype conversion 7](#_Toc473632260)

[2.2.4 String Functions 8](#_Toc473632261)

[2.2.5 IIF Conditionals are costly 8](#_Toc473632262)

[2.2.6 Test Expressions slow down calculation 9](#_Toc473632263)

[2.2.7 Unused Ports 9](#_Toc473632264)

[2.2.8 IIF and Datatypes 9](#_Toc473632265)

[2.2.9 Unexpected results when a NULL value is in a calculation 10](#_Toc473632266)

[2.3 Aggregator transformation 10](#_Toc473632267)

[2.3.1 Too Many Aggregators 10](#_Toc473632268)

[2.3.2 Mapplets containing Aggregators 11](#_Toc473632269)

[2.3.3 Sorted input option 11](#_Toc473632270)

[2.3.4 Lookup and Aggregators conflicts 12](#_Toc473632271)

[2.4 Sequence transformation 12](#_Toc473632272)

[2.4.1 Alternatives 12](#_Toc473632273)

[2.5 Joiner transformation 12](#_Toc473632274)

[2.5.1 Sorted input option 12](#_Toc473632275)

[2.5.1 Define the master table 13](#_Toc473632276)

[2.6 Update transformation 13](#_Toc473632277)

[2.6.1 Session set to “Update Else Insert“ 13](#_Toc473632278)

[2.7 Lookup transformation 14](#_Toc473632279)

[2.7.1 Eliminate "too many lookups" 14](#_Toc473632280)

[2.7.2 Lookup SQL performance considerations 14](#_Toc473632281)

[2.7.3 LKP conditions 14](#_Toc473632282)

[2.7.4 Unconnected Lookup port default 14](#_Toc473632283)

[2.7.5 Cached Lookup vs Uncached lookup 14](#_Toc473632284)

[2.7.1 Persistent LKP caches 15](#_Toc473632285)

[2.8 Which would be better: Join in source qualifier or Lookup? 15](#_Toc473632286)

[2.9 Source and target transformation 15](#_Toc473632287)

[2.9.1 Multiple Targets are too slow 15](#_Toc473632288)

[2.10 Mapping design 15](#_Toc473632289)

[2.10.1 Reduce the number of OBJETS in a mapping 15](#_Toc473632290)

[2.10.2 Variables are better than hard coding 16](#_Toc473632291)

[2.10.3 Filter data at the earliest 16](#_Toc473632292)

[2.10.4 Link only needed fields between the source and the target of the mapping 16](#_Toc473632293)

[2.10.5 Calculate values only once 16](#_Toc473632294)

[3 - Settings 17](#_Toc473632295)

[3.1 Session settings 17](#_Toc473632296)

[4 - Applicative architecture and infrastructure 18](#_Toc473632297)

[4.1 Network 18](#_Toc473632298)

[4.1.1 Slower performances across networks 18](#_Toc473632299)

[4.2 System 18](#_Toc473632300)

[4.2.1 Sizing of File systems 18](#_Toc473632301)

[4.3 Teradata 18](#_Toc473632302)

[4.3.1 Using Temporary tables 18](#_Toc473632303)

[5 - Informatica PowerCenter options for performances 19](#_Toc473632304)

[5.1 Partitioning option: Process Massive Data Volumes with High Performance 19](#_Toc473632305)

[5.1.1 Key Features 20](#_Toc473632306)

[5.1.2 Key Benefits 20](#_Toc473632307)

[5.2 Grid option 21](#_Toc473632308)

[5.2.1 Cost-Effectively Scale Data Integration Environments with Grid Computing 21](#_Toc473632309)

[5.2.2 Key Features 21](#_Toc473632310)

[5.2.3 Key Benefits 22](#_Toc473632311)

[5.3 Informatica PowerCenter Pushdown Optimization 22](#_Toc473632312)

[5.3.1 Increase Performance with Optimal Use of Database Resources 22](#_Toc473632313)

[5.3.2 Key Features 23](#_Toc473632314)

[5.3.3 Key Benefits 24](#_Toc473632315)

[5.3.1 Feedback on this options 24](#_Toc473632316)

[6 - Project phases and performances 25](#_Toc473632317)

[6.1 Phase of analyze 25](#_Toc473632318)

[6.1.1 Analyze of the technical environment 25](#_Toc473632319)

[7 - Matrix of Teradata connections types available on PowerCenter 26](#_Toc473632320)

[7.1 Matrix for relational connections: 26](#_Toc473632321)

[7.2 Matrix for Loader connections: 26](#_Toc473632322)

[8 - Matrix between Issues and Solutions 27](#_Toc473632323)

[9 - General Guidelines for optimizations 29](#_Toc473632324)

[9.1 General Guidelines for Optimizing Mappings 29](#_Toc473632325)

[9.2 Operations and Expression Optimization Tips 30](#_Toc473632326)

[9.3 Lookup Transformation Optimization Guidelines 31](#_Toc473632327)

[10 - How to analyze 33](#_Toc473632328)

[10.1 Source Bottlenecks 33](#_Toc473632329)

[10.1.1 Relational sources 33](#_Toc473632330)

[10.1.2 Flat file sources 33](#_Toc473632331)

[10.2 Target Bottlenecks 34](#_Toc473632332)

[10.2.1 Relational Targets 34](#_Toc473632333)

[10.2.2 Flat file targets 34](#_Toc473632334)

[10.3 Mapping Bottlenecks 34](#_Toc473632335)

[10.4 Session Bottlenecks 35](#_Toc473632336)

[10.5 System Bottlenecks 36](#_Toc473632337)

# Introduction

The relative ease in which mappings can be built often leads to underestimating the challenge of designing an efficient mapping that will process the data in the optimal way. Sub-standard mapping design may be the cause of performance issues in production environments; it can lead to challenges in maintenance and operational support.

Although PowerCenter environments vary widely, most sessions and/or mappings can benefit from the implementation of common objects and optimization procedures. Follow these guidelines and general principles when creating mappings to ensure optimization in design, maintainability and performance.

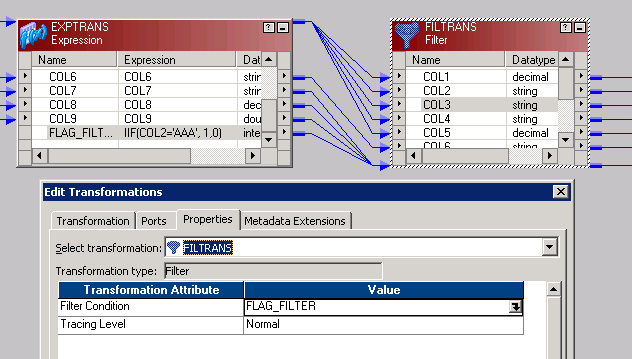
# Design

The following lists some performance considerations when building mappings.

## Filter expressions

### Evaluation upstream of filter conditions

It is better to create the filter answer TRUE or FALSE inside a port expression upstream. Complex filter expressions slow down the mapping. Expressions and conditions operate fastest in an Expression Object with an output port for the result.  It turns out that longer or more complex is the expression, more severe is the speed degradation. Place the actual expression (complex or not) in an EXPRESSION OBJECT upstream from the filter. Compute a single numerical flag: 1 for true and 0 for false as an output port. Fill this into the filter expression. You should see the maximum performance ability with this configuration.

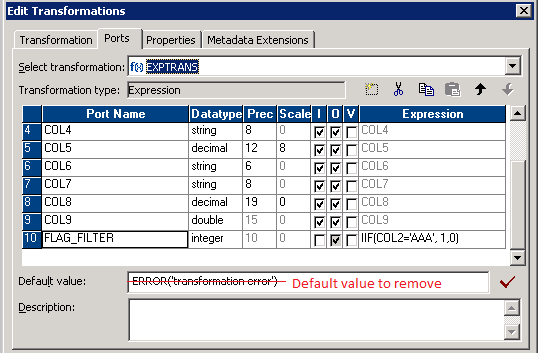


## Expression transformation

### Remove "DEFAULT" values

Having a default value, even the "ERROR(xxx)" command slows down the session. It causes an unnecessary evaluation of values for every data element in the map. The only time you want to use "DEFAULT value is when you have to provide a default value for a specific port.

Another method is to use an expression like an IIF(xxxx, DEFAULT VALUE, xxxx) condition within an expression transformation. This will always be faster (if assigned to an output port) than a default value.

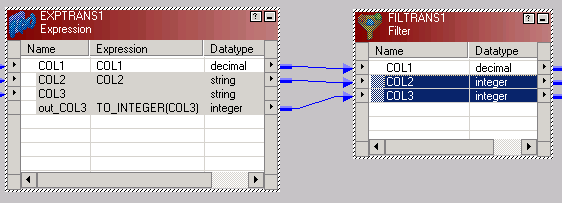


### Variable Ports are "slower" than Output Expressions.

Whenever possible, use output expressions instead of variable ports. The variables are good for "static and state driven" but do slow down the processing time as they are allocated and reallocated each pass of a row through the expression transform.

### Datatype conversion

Simply mapping a string to an integer, or an integer to a string will perform the conversion. However it will be slower than creating an output port with an expression like: TO\_INTEGER(xxxx) and mapping an integer to an integer. It's because PMServer is left to decide if the conversion can be done mid-stream which slows things down.



In the case below, we have

* An implicit conversion of the field COL2 from string to integer. PowerCenter will perform by itself the datatype conversion.
* An explicit conversion of the field COL3 which is the good way to do as we inform PowerCenter of the change of datatype to perform

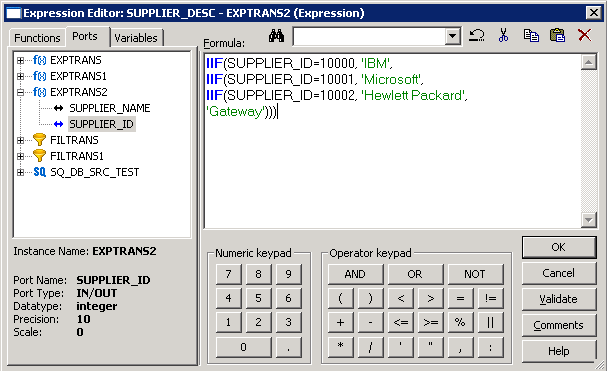
### String Functions

String functions definitely have an impact on performance and particularly those that change the length of a string (substring, ltrim, rtrim, etc.). These functions slow the mapping down considerably. The operations behind each string function are expensive (de-allocate, and re-allocate memory within a READER block in the session). String functions are a necessary and important part of ETL. It is not recommended to remove their use completely. Only try to limit them to necessary operations. One of the ways to tune these is to use "varchar/varchar2" data types in your database sources, or to use delimited strings in source flat files (as much as possible). This will help reduce the need for "trimming" input. If your sources are in a database, perform the LTRIM/RTRIM functions on the data coming in from a database SQL statement, this will be much faster than operationally performing it mid-stream.

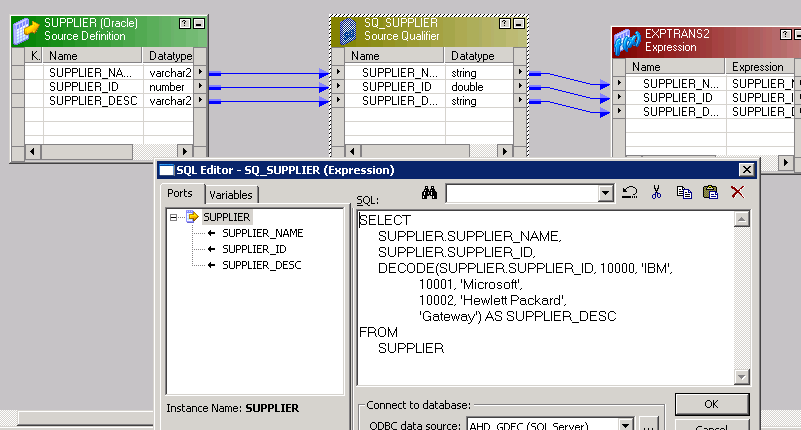
### IIF Conditionals are costly

When possible, arrange the logic to minimize the use of IIF conditionals. This is not particular to Informatica.  It is costly in ANY programming language. It introduces "decisions" within the tool; it also introduces multiple code paths across the logic (thus increasing complexity). Therefore, when possible, avoid utilizing an IIF conditional. The only possibility here might be (for example) an ORACLE DECODE function applied to a SQL source.

The screenshot below shows a common way to perform transcodifications into an expression.



When possible, another way should be to perform it at SGBD level like below:



### Test Expressions slow down calculation

Expressions such as IS\_SPACES tend to slow down mappings. This is a data validation expression which has to run through the entire string to determine if it is spaces, much the same as IS\_NUMBER has to validate an entire string. These expressions (if at all avoidable) should be removed in cases where it is not necessary to "test" prior to conversion. Be aware however, that direct conversion without testing (conversion of an invalid value) will kill the transformation. If you absolutely need a test expression for a numerical value, try this: IIF(<field> \* 1 >= 0,<field>,NULL) preferably you don't care if it's zero. An alpha in this expression should return a NULL to the computation. Yes - the IIF condition is slightly faster than the IS\_NUMBER - because IS\_NUMBER parses the entire string, where the multiplication operator is the actual speed gain.

### Unused Ports

Unused output ports have no affect on performance. However in general it is good practice to remove any unused ports in the mapping, including variables. Unfortunately, there is no "quick" method for identifying unused ports.

### IIF and Datatypes

If the precision and scale of value1 in an IIF statement is different than value 2, the precision for value 2 is rounded.

For example, if you have a value for sales of 7.75, and you have the following statement, a value of 7.7 will be returned:

*IIF (ISNULL (Sales), 0.0, Sales)*

If you had wanted two decimal places returned for sales, you would need to make your IIF statement look like the following:

*IIF (IS Null (Sales), 0.00, Sales)*

### Unexpected results when a NULL value is in a calculation

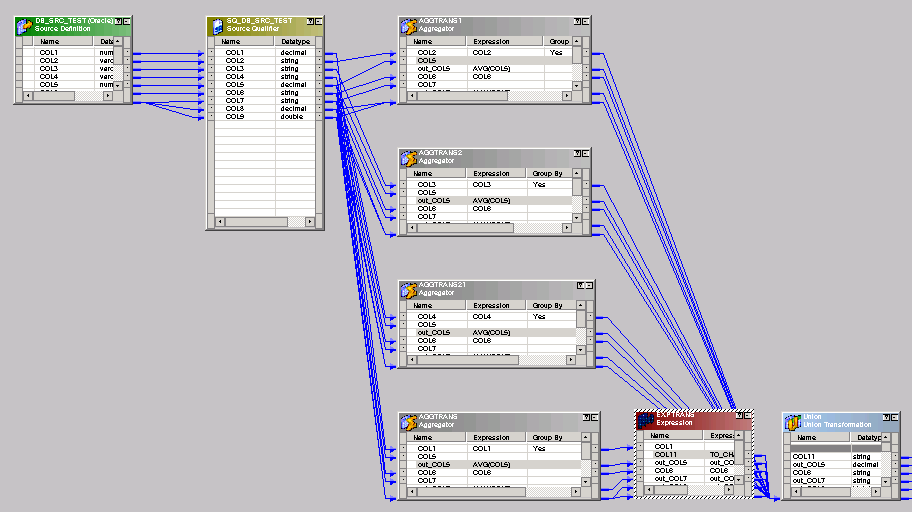
If you have an expression transformation that adds two values together (e.g.  A+B), if one of the values happens to contain a null value, the resulting sum will be set to null.

One solution to this problem is to make sure you always set your default value for your port to 0 for numeric values.

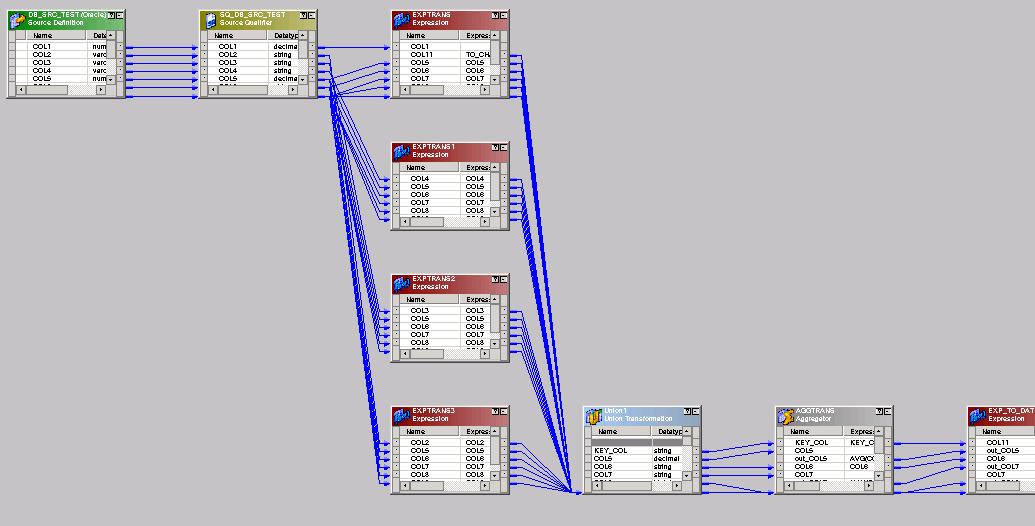
## Aggregator transformation

### Too Many Aggregators

If your mapping has more than 1 aggregator, chances are the session will run very slowly. Unless the CACHE directory is extremely fast and your drive seek / access times are very high.  Even still, placing aggregators end-to-end in mappings will slow the session down by factors of at least 2.  This is because of all the I/O activity being a bottleneck in Informatica.



A solution for this mapping below could be the normalization of input data like below:



### Mapplets containing Aggregators

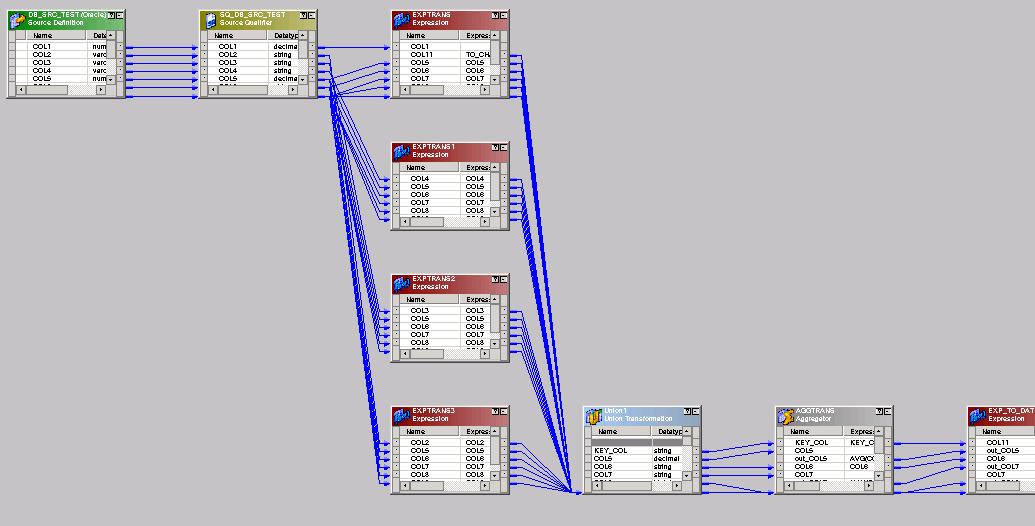
Mapplets are a good source for replicating data logic. But just because an aggregator is in a mapplet doesn't mean it will not affect the mapping. A maplet is treated as a part of the mapping once the session starts.  In other words, if you have an aggregator in a mapplet, followed by another aggregator in a mapping you will still have the problem mentioned above. Reduce the number of aggregators in the entire mapping (including mapplets) to 1 if possible. If necessary split the mapping up in to several different mappings, use intermediate tables in the database if required to achieve processing goals.

### Sorted input option

Inputs of aggregators have to be sorted to improve performances.

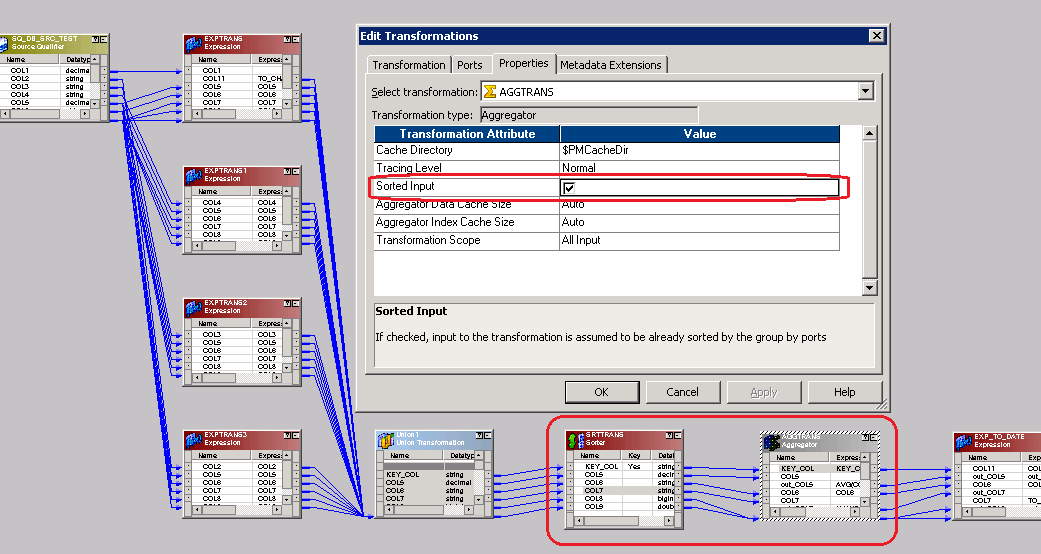
Aggregators generate caches in order to perform its job. When data in input of these transformations is not sorted, Informatica needs 2 distinct passes to perform it and the allocated resource is shared for these 2 passes. First pass is dedicated to sort records on the set key. The second one is dedicated to the tasks awaited (aggregation or join). By sorting data before sending it to the aggregator or to the joiner, these objects dedicate their whole cache to their aggregation task. So by sorting input data make the mapping is more efficient.

Below a case that can be improved:



Data going to the aggregator are not sorted. The cache will be splitted in 2 parts. First one will sort data and second one will be used to aggregate.

To improve it, only adding a sorter just before the aggregator which sort data on the key field is enough.



### Lookup and Aggregators conflicts

The lookups and the aggregators fight for memory space as discussed above. Each requires Index Cache, and Data Cache and they "share" the same HEAP segments inside the core.   These memory areas become critical. When dealing with many rows, the session is almost certain to cause the server to "thrash" memory in and out of the OS Swap space. If possible, separate the mappings, perform the lookups in the first section of the mappings, position the data in an intermediate target table. Then a second mapping reads the target table and performs the aggregation (also provides the option for a group by to be done within the database)... Another speed improvement...

## Sequence transformation

### Alternatives

There is no fast and easy way to create sequence generators. The cost is not that high for using a sequence generator inside of Informatica, particularly if you are caching values (cache at around 2000) seems to be the suite spot. However, if at all avoidable, this is one "card" up a sleeve that can be played. If you don't absolutely need the sequence number in the map for calculation reasons, and you are utilizing Oracle, then let SQL\*Loader create the sequence generator for all Insert Rows. If you're using Sybase, don't specify the Identity column as a target - let the Sybase Server generate the column. Also - try to avoid "reusable" sequence generators - they tend to slow the session down further, even with cached values.

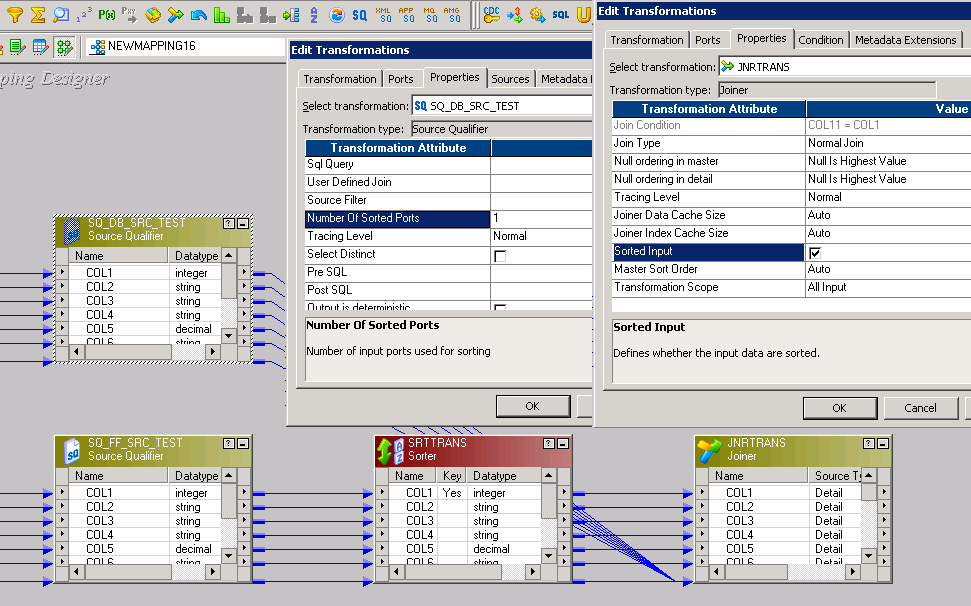
## Joiner transformation

### Sorted input option

Inputs of joiners have to be sorted to improve performances.

Joiners generate caches in order to perform its job. When data in input of these transformations is not sorted, Informatica needs 2 distinct passes to perform it and the allocated resource is shared for these 2 passes. First pass is dedicated to sort records on the set key. The second one is dedicated to the tasks awaited (aggregation or join). By sorting data before sending it to the aggregator or to the joiner, these objects dedicate their whole cache to their join task. So by sorting input data make the mapping is more efficient.

In the screenshot below, data coming from Oracle are sorted by setting “Number Of Sorted Ports” to 1 (meaning sorting on Port 1 of the source). Data coming from the flat file are sorted by a sorter tansformation. At joiner level, “Sorted Input” option is checked.



### Define the master table

The master table of a joiner is cached and the detail one is processed on the fly. In order to consume less resource, the table with the lower volume has to be the master table.

## Update transformation

### Session set to “Update Else Insert“

If you have this switch turned on, it will definitely slow the session down. Informatica performs 2 operations for each row: update (w/PK), then if it returns a ZERO rows updated, performs an insert. The way to speed this up is to "know" ahead of time if you need to perform a DD\_UPDATE or a DD\_INSERT inside the mapping. Then tell the update strategy what to do. After which you can change the session setting to: INSERT and UPDATE AS UPDATE or UPDATE AS INSERT.

## Lookup transformation

### Eliminate "too many lookups"

With too many lookups, your cache is eaten in memory. The end result is that there is no memory left for the sessions to run in. The DTM reader/writer/transformer threads are not left with enough memory to be able to run efficiently. When the cache is full, PowerCenter is caching some of these lookups out to disk and is running slower. But you still end up with contention. With too many lookups, you're trading in Memory Contention for Disk Contention.  The memory contention may be worse than the disk contention because the system OS end's up thrashing (swapping in and out of TEMP/SWAP disk space) with small block sizes to try and locate your lookup row.   As the row goes from lookup to lookup, the swapping / thrashing gets worse.

### Lookup SQL performance considerations

A lookup SQL query always places an order by on the SQL for every lookup port.   For example, if the Lookup transformation includes three lookup ports used in the mapping, ITEM\_ID, ITEM\_NAME, and PRICE, the lookup query is:

*SELECT ITEM\_NAME, PRICE, ITEM\_ID*

*FROM ITEMS\_DIM ORDER*

*BY ITEM\_ID, ITEM\_NAME, PRICE*

Therefore, if your lookups are slow in loading, you may need to have an index that your ORDER BY statement can utilize.

### LKP conditions

Equal conditions (“=”) have to be first ones of the LKP SQL query. Then are coming other ones (“>”, “<”, “!=”)

### Unconnected Lookup port default

If you have an unconnected lookup, Informatica ignores what is in the default port.  It always returns a null if the lookup is unsuccessful.

### Cached Lookup vs Uncached lookup

|  |  |
| --- | --- |
| **Case** | **Type of lookup** |
| LKP with less than 1000 records | Uncached |
| LKP with 1000 to 500000 records | Cached |
| LKP called more than  (count(lookup table) / 10) times | Cached |

### Persistent LKP caches

|  |  |
| --- | --- |
| **Case** | **Type of cache (persistent / not persistent)** |
| Same LKP used in different mappings | Persistent |
| Referential tables rarely updated | Persistent |
| Otherwise | Not persistent |

## Which would be better: Join in source qualifier or Lookup?

It depends on 2 things :

1. how the DB server and the PowerCenter machine compare in terms of performance. Give it a try, there's no other way to find out.
2. how well the session is tuned.  For example, if you use a Lookup cache with a large amount of data, it is going to create a large cache size and if the session is not tuned to take advantage of the most amount of memory then this will \*usually\* be slower than a SQL override.

## Source and target transformation

### Multiple Targets are too slow

Frequently mappings are generated with multiple targets and sometimes with multiple sources. This (despite first appearances) can really burn up time. If the architecture permits change, and the users support re-work, then try to change the architecture. One mapping per target is the general rule of thumb. Once reaching one mapping per target, the tuning gets easier. Sometimes it helps to reduce it to 1 source and 1 target per mapping. But if the architecture allows more modularization 1 mapping per target usually does the trick. Going further, you could break it up: 1 mapping per target per operation (such as insert vs update). In doing this, it will provide a few more cards to the deck with which you can "tune" the session, as well as the target table itself. Going this route also introduces parallel operations. For further info on this topic, see my architecture presentations on Staging Tables, and 3rd normal form architecture.

## Mapping design

### Reduce the number of OBJETS in a mapping

Frequently, the idea of these tools is to make the "data translation map" as easy as possible. All to often, that means creating "an" expression for each throughput/translation (taking it to an extreme of course). Each object adds computational overhead to the session and timings may suffer. Sometimes if performance is an issue / goal, you can integrate several expressions into one expression object, thus reducing the "object" overhead. You could speed up the mapping by this way.

It is recommended not to have more than 20 to 30 objects into a mapping.

### Variables are better than hard coding

Hard coding is not a good way to develop and in case of rule changes, interfaces have to be updated in order to be aligned with the functional needs.

### Filter data at the earliest

Usage of resources and performance are depending of the volume to be processed. So it is recommended to filter data at the earliest possible.

You can for example add a SQL filter on the source qualifier based on criteria just to extract the scope of data you want to process.

### Link only needed fields between the source and the target of the mapping

Link only field you need between transformation. So PowerCenter will need less memory to process a record.

### Calculate values only once

A same value can be necessary at different level of a mapping. No need to calculate it multiple times. Calculate it once and propagate it by a output prot from a transformation to the next ones.

# Settings

## Session settings

Most of the time, default parameters are enough and we don’t need to modify them.

But someones can be tuned in order to improve performances.

|  |  |  |  |
| --- | --- | --- | --- |
| **Subject / Case** | **Settings by default** | **Settings to apply** | **Comments** |
| Retention of logs | Session logs save by “Session runs”  AND  Save session log for these runs : 0 | 1 - Session logs save by “Session runs”  2 - Save session log for these runs : $PMSessionLogCount | Need to defined a value for $PMSessionLogCount parameter at IS level. By default, it is 0. |
| Session logs save by “Session timestamp” | /!\ Need to implement a cleansing / archiving job no to fill full the FS |
| Commit Interval | 10000 | TBD based on volume to process | For huge volumes, it is better to have a bigger commit interval. |
| Error handling > Override tracing | None | None | Other override tracking are only used for debugging |
| Line Sequential buffer length | 1024 | Max length of a record into the mapping | Has to be set greater than the max length of a record in order to process it in one time. Else the record is split into multiple records with a length equals to the sequential buffer length. |
| Target load type | Bulk | Bulk | When inserting data into a table without indexes or keys |
| Normal | When inserting / updating / deleting records into a table using indexes and keys |

**/!\** In some case, caches can be tuned in order to improve performances. But this has to be done with a clear view of the future evolution of volumes to be processed. We have to keep in mind that a cache size defined today can be wrong later.

# Applicative architecture and infrastructure

## Network

### Slower performances across networks

If you have got slow sources, you can look at some of the following possibilities.

* If the flat file sources reside on a different machine, and you've opened a named pipe to get them across the network, then you have opened (potentially) a can of worms. You have introduced the network speed as a variable on the speed of the flat file source. Try to compress the source file, FTP PUT it on the local machine (local to PMServer), decompress it, and then utilize it as a source.
* If you're reaching across the network to a relational table, and the session is pulling many rows (over 10,000) then the source system itself may be slow.  You may be better off using a source system extract program to dump it to file first. Then follow the above instructions. However, there is something your SA's and Network Ops folks could do (if necessary).  They could backbone the two servers together with a dedicated network line (no hubs, routers, or other items in between the two machines).  At the very least, they could put the two machines on the same sub-net.
* If your file is local to PMServer but is still slow, examine the location of the file (which device is it on). If it's not on an INTERNAL DISK then it will be slower than if it were on an internal disk.  This doesn't mean a unix file LINK exists locally, and the file is remote - it means the actual file is local.

One good example is GMM project (ex-GMC project).

* Database was located into zone L3
* Informatica server was into the CITS

Because of 3 firewalls between the 2 zones, performances were decreased. Before migration, some interfaces of GMC last 15 min. After migration, it lasts 50 min.

## System

### Sizing of File systems

The volume of data to be processed affects resources allocated to the project. So it is recommended to have an idea of this volume in order to size the FS the best as caches, indexes and files are stored on it.

## Teradata

### Using Temporary tables

To avoid issues in archiving, it is recommended to use distinct schemas for temporary tables and target ones.

# Informatica PowerCenter options for performances

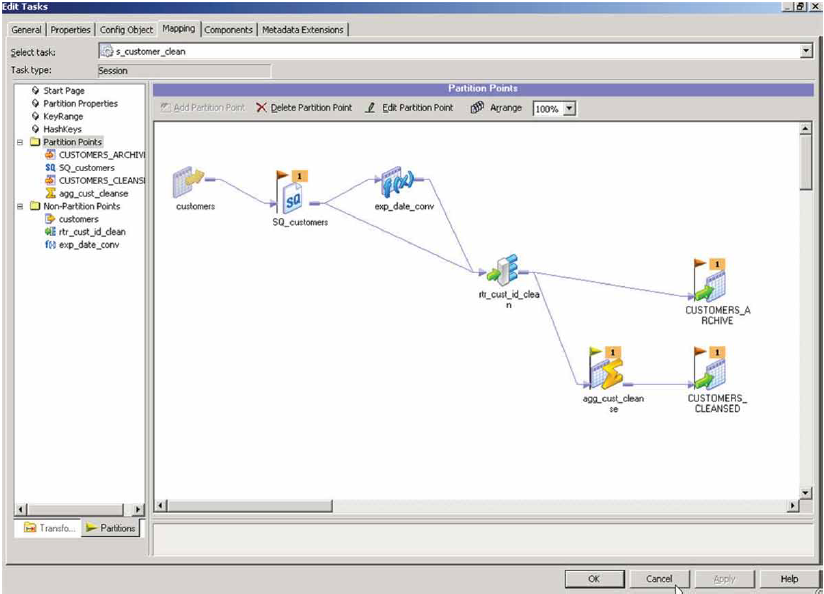
This part describes some Informatica options which improve performances of interfaces. But these options generate extra costs in terms of licensing. So it cannot be implemented on demand on each projects.

## Partitioning option: Process Massive Data Volumes with High Performance

The Informatica® PowerCenter® Partitioning Option helps your IT organization take advantage of parallel data processing in multiprocessor and grid-based hardware environments. This option increases PowerCenter’s performance with a thread-based architecture and automatic data partitioning capabilities. The option executes optimal parallel sessions by dividing data processing into subsets, which run in parallel and are spread among available CPUs in a multiprocessor system.

Unlike approaches that require manual data partitioning, the Partitioning Option automatically guarantees data integrity because PowerCenter’s parallel engine dynamically realigns data partitions for set-oriented transformations.Configurable session options, such as error handling, recovery strategy, memory allocation, and logging, make it easier to gather statistics used to maximize performance.

By enabling hardware and software to jointly scale to handle large volumes of data and users, the Partitioning Option helps your IT organization maximize its technology investments, boost productivity, and optimize system performance.



The Partitioning Option dynamically optimizes jobs for parallel processing at run time based on data-driven, key-driven,or databasesupplied partitioning schemes, dramatically increasing PowerCenter’s performance.

### Key Features

Data Smart Parallelism

This feature improves system performance by automatically aligning PowerCenter partitions with database table partitions.

Guaranteed Data Integrity

Leveraging PowerCenter’s parallel engine to dynamically realign data partitions for set-oriented transformations, such as aggregators or sorters, this option guarantees data integrity.

Session Design Tools

Intuitive, GUI-based session design tools are available for creating partitioning schemes, determining the best partitioning points, and gathering statistics on error handling, recovery strategy, memory allocation, and logging.

Integrated Monitoring Console

All session execution and dependency details can be viewed from an integrated monitoring console, where your IT team can gather session statistics—such as throughput, rows/second, error details, and performance—to identify potential bottlenecks and recognize trends.

Multiple Partition Schemes

This option provides multiple partition schemes to support parallelization through key range, hash algorithm-based, round robin, or file partitions and through concurrent processing of partitions along the data transformation pipeline to maximize data throughput.

### Key Benefits

Maximize Return on IT Investments

With this option, your IT team can divide data processing into subsets that are run in parallel and spread among available CPUs in a multiprocessor system. Large data volumes can be processed faster. By scaling hardware and software to handle large volumes of data and users, this option maximizes return on your IT investments.

Boost IT Productivity

This option’s session design tools simplify and streamline repetitive configuration and performance tuning tasks to boost your IT team’s productivity. You can easily create userdefined partitioning schemes. A graphical partitioning map helps you determine the best points of partitioning. Configurable session options make it easier to gather statistics used to maximize performance.

Optimize System Performance in Response to Changing Business Requirements

This option optimizes PowerCenter performance by empowering your IT team to gather in-depth session statistics to recognize trends and identify potential bottlenecks. Data transformation logic is abstracted from the physical execution plan, which enables IT to rapidly tune performance without compromising the logic and design of the original data mappings. With this option, you can easily optimize system performance in the face of increasing data loads and changing business requirements.

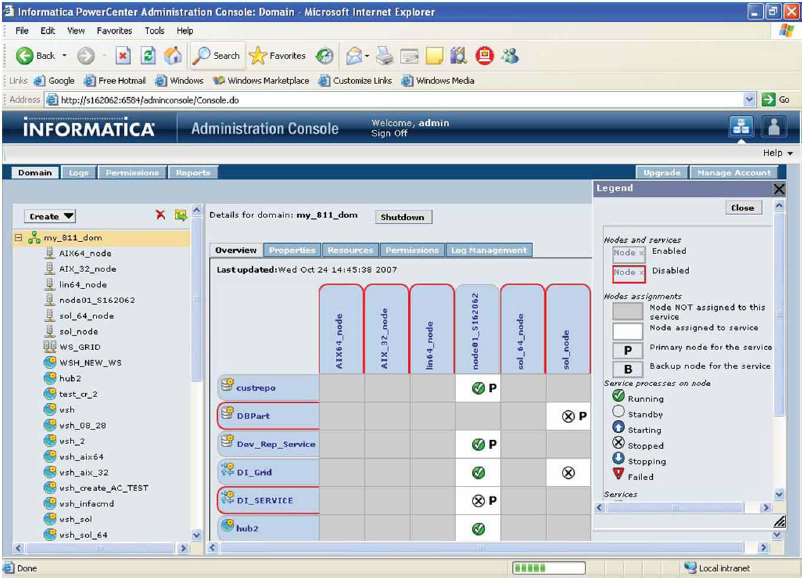
## Grid option

### Cost-Effectively Scale Data Integration Environments with Grid Computing

Commodity hardware, such as blade servers, has shown a great deal of promise in scalable processing to dramatically lower costs. But these grid environments are dynamic. Nodes are constantly added to and removed from the grid. In addition, parallelization schemes require constant tuning to optimize performance.

The Informatica® PowerCenter® Enterprise Grid Option exploits grid computing architectures for data integration. This option minimizes the amount and frequency of data mapping modifications needed to respond to changing conditions. The parallel execution plan can be determined dynamically at run time. With this option, your IT organization can ensure uptime through self-monitoring of Informatica PowerCenter services for seamless resilience, failover, and recovery.

With the Enterprise Grid Option, your IT organization can reduce the administrative overhead of supporting grid computing environments and cost-effectively scale data integration initiatives using low-cost, commodity hardware.



The Enterprise Grid Option allows parallelization of a single session across a grid and offers high availability of the entire data integration platform, increasing both scalability and reliability.

### Key Features

Workflow and Session-on-Grid

The Enterprise Grid Option distributes processing of a single session across multiple server nodes on a grid. It allows incremental investments in processing capacity. It helps to costeffectively scale data integration initiatives using lower-cost commodity hardware, such as blade servers. This feature extends PowerCenter’s scalability, rendering it virtually unlimited.

Adaptive Load Balancing

This feature ensures smooth operations and reliable scale out when facing spikes in data volumes or unexpected utilization loads on the grid. You can select the best node for session execution based on resource requirements and availability. This feature factors in cost-based metrics such as CPU utilization, process size, and memory.

Dynamic Partitioning

This feature automatically adjusts the parallel execution plan in response to additions/ removals of nodes from the grid without changing the logic. It exponentially improves PowerCenter performance by automatically adapting to exploit available grid resources.

Web-Based Administration Console

This console provides secure, centralized control over the provisioning of server nodes, PowerCenter resources, and PowerCenter services both into and out of the enterprise data integration grid.

Robust Interoperability

This option allows heterogeneous hardware operating systems to be used across the grid and to interoperate with enterprise grid resource software. It automatically applies all high availability features to all data integration processing available on PowerCenter.

### Key Benefits

Enhance IT’s Agility and Responsiveness to Changing Business Demands

Your IT organization often needs to address sudden and dramatic increases in data volumes. With this option, your IT organizations can cost-effectively scale your computing capacity as business needs grow.

Scale Performance with Available Hardware

This option provides unparalleled support for grid computing environments. In turn, IT organizations can cost-effectively scale to adjust and optimize data delivery to the |enterprise. This option makes the best use of existing hardware assets or new cost-effective hardware options.

Increase System Reliability and IT Productivity

With PowerCenter’s unique architecture, the mapping specification is abstracted from the parallel execution plan, paving the way for the Enterprise Grid Option to offer dynamic partitioning and load balancing. Dynamic partitioning automatically adjusts the parallel execution plan in response to additions or removals of nodes from the grid or to changes in RDBMS table partitioning schemes. The partitioning capability boosts productivity by freeing developers and administrators from having to constantly refactor the parallelization scheme.

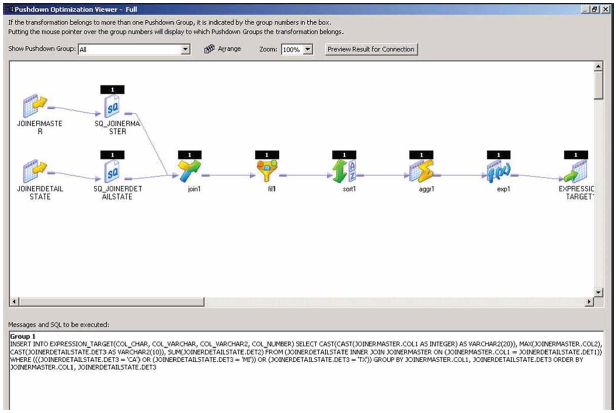
## Informatica PowerCenter Pushdown Optimization

### Increase Performance with Optimal Use of Database Resources

The Informatica® PowerCenter® Pushdown Optimization Option enables data transformation processing, where appropriate, to be pushed down into any relational database to make the best use of database assets. With this option, PowerCenter is the first data integration software on the market to offer a choice of where data processing occurs.

In most instances, the PowerCenter data server functions as the engine behind data processing. When both source and target data are co-resident in a relational database, processing can be pushed down into that database. This option optimizes the performance of your IT systems in response to changing run-time demands, peak processing needs, or other dynamic aspects of the production environment.

With the Pushdown Optimization Option, you enhance your IT organization’s agility and flexibility by being able to adapt easily to changing business conditions. And you boost productivity by empowering developers to control where processing takes place and create and execute transformation mappings easily using a single interface.



The Pushdown Optimization Option improves system performance by enabling processing to be pushed down to a relational database, maximizing flexibility and minimizing unnecessary data movement.

### Key Features

Automatic Generation and Pushdown of Mapping Logic

This feature generates database-specific logic that represents the overall data flow. It automatically shifts data transformation processing to the most appropriate processing resource.

Database Neutrality

By exploiting the processing power of the database in which the data resides, this feature helps IT organizations reap greater scalability and performance. It ends reliance on database-specific programming.

Single Design Environment

A user-friendly interface makes it easy to decouple data transformation logic from the physical execution plan. With this feature, your IT team can control where processing takes place and dynamically create and execute database-specific transformation language.

### Key Benefits

Increase IT Systems’ Flexibility, Cost-Effective Scalability, and Performance

In most instances, the PowerCenter data server functions as the engine behind data processing. If both source and target data are co-resident in a relational database, then processing can be pushed down into that database. Pushdown optimization is abstracted from data mapping, which optimizes IT systems’ performance in response to changing run-time demands, peak processing needs, or other dynamic aspects of the production environment.

Enhancing IT’s Agility and Flexibility

Even when your IT organization can standardize on a single relational database management system, you need to be prepared to support multiple relational databases as business conditions change. With the Pushdown Optimization Option, you get a database-neutral approach and maximum flexibility. Your IT organization can fully leverage the capabilities of each type of database and stay agile enough to rapidly integrate other types of databases as the need arises.

Boost Developer Productivity

The Pushdown Optimization Option boosts productivity by allowing your development team to control where processing takes place. Your team doesn’t have to resort to time-consuming database-specific programming to exploit database processing power. You can create and execute database-specific transformation language by simply selecting pushdown optimization in the PowerCenter interface—a huge time savings.

### Feedback on this options

Below is the feedback from BYQ on this option:

* When applying this option on an existing project, we are obliged to re-develop all Informatica interfaces.
* The processing last of partitioned sessions has been divided by 2.
* Some sessions which were running during 2 hours before implementing this option, last 8 min after its installation.

# Project phases and performances

To ensure less performances troubles will happen all along project phases, some reflections have to be done before launching developments.

## Phase of analyze

### Analyze of the technical environment

Where are servers? Where is stored the database? Where are located tier-applications? Are these applications on the same network? What are resources available on systems? What is the volume to be processed?

Before starting developments, we have to know answers of these questions. This imposes a reflection with performances as a pre-requisite. So it is recommended to plan a meeting with GTS, APS, API, BscDco and ME in order to discuss different available solutions.

Based on this can be defined the locations of servers and a strategy in developments like working more either at DB level or at Informatica one.

# Matrix of Teradata connections types available on PowerCenter

This matrice described connections available for Teradata on PowerCenter

## Matrix for relational connections:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Connection type** | **Description** | **Strength** | **Weakness** | **Use cases** |
| Teradata |  |  |  |  |
| Teradata PT Connection |  |  |  |  |

## Matrix for Loader connections:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Connection type** | **Description** | **Strength** | **Weakness** | **Use cases** |
| Teradata Mload External Loader | Performs insert, update, delete and upsert operations for large volume incremental loads.  Use this loader when you run a session with a single partition. Multiload acquires table level locks, making it appropriate for offline loading. |  |  |  |
| Teradata Tpump External Loader | Performs insert, update, delete and upsert operations for relatively low volume updates.  Use this loader when you run a session with multiple partitions. TPump acquires row-hash locks on the table, allowing other users to access the table as TPump loads to it. |  |  |  |
| Teradata FastLoad External Loader | Performs insert operations for high volume initial loads, or for high volume truncate and reload operations. Use this loader when you run a session with a single partition. Use this loader on empty tables with no secondary indexes. |  |  |  |
| Teradata Warehouse Builder External Loader | **/!\ Connection deprecated. Cannot be created anymore.** |  |  |  |

# Matrix between Issues and Solutions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue** | **Environment setting** | **Check to perform** | **Resolution** | **Comments** |
| Slow Flat File sources / target | The flat file is stored on the local disk of the server |  | Review your Informatica interface. | You can have a bottleneck into your mapping |
| The file is stored on a NAS, a SAN or a GPFS | Check issues with the NAS, the SAN or the GPFS | Contact Linux team |  |
| Check network issues | Contact Network team |  |
|  | Review your Informatica interface | You can have a bottleneck into your mapping |
| Slow DB sources / target | The database is stored on the same server or on the same network | Check performance on DB server | Contact DBA |  |
| Review your Informatica interface. | Review select query and add indexes on the table |  |
| Review update query and add indexes on the table |  |
| Updates in target have to be done by an update strategy |  |
| Review the target commit point value |  |
| The database is on another network | Check performance on DB server | Contact DBA |  |
| Review your Informatica interface. | Review select query and add indexes on the table |  |
| Review update query and add indexes on the table |  |
| Updates in target have to be done by an update strategy |  |
| Review the target commit point value |  |
| Check network issues | Contact network team |  |
| Slow processing |  | Check logs to identify transformations working a long time | Ensure aggregator inputs are sorted |  |
| Ensure joiner input are sorted |  |
| Ensure you don’t have too many lookups | To improve performances, cache your lookups  LKP targeted tables should have good indexes |
| If you are using normalize transformations, replace it by expression and union transformations |  |
| Check logs to identify transformations using more cache than default defined by the IS | Update caches size based on logs and usual processed volume |  |
| You process a huge number of records |  |  |

# General Guidelines for optimizations

## General Guidelines for Optimizing Mappings

1. Reduce the number of transformations. The more transformations, the more complicated it is to understand and maintain the mapping.
2. The default ‘Auto’ setting is applicable for most sessions. For mappings that process small amounts of data in an environment with memory constraints, consider changing the ‘Auto’ setting to 40MB or less.
3. Calculate once, use many times.

* Avoid calculating or testing the same value over and over.
* Calculate it once in an expression, and set a True/False flag or assign the value to an output port for downstream use in the mapping.
* Within an expression, use variable ports to calculate a value that can be used multiple times within that transformation. This is especially important for results of unconnected lookups, to avoid these being executed multiple times with sometimes serious performance consequences.

1. Only connect what is used.

* When a port is not needed downstream in the mapping, uncheck the Output property to avoid unnecessary links between transformations. This is especially important in the Source Qualifier, where the SQL is generated according to the ports linked to downstream transformations. Only reading what is needed saves unnecessary transport of data from the source table or file to the transformation buffer.
* This is also helpful for maintenance. The smaller the number of ports in each transformation, the easier it is to understand the mapping.
* In lookup transformations, change unused ports to be neither input nor output. This makes the transformations cleaner looking. It also reduces the number of columns cached and thereby improves performance.

1. Watch the data types.

* The engine automatically converts compatible types.
* Sometimes data conversion is excessive, which can reduce performance significantly and may evencause loss of data. Data types are automatically converted when types differ between connected ports. Minimize data type changes between transformations by planning data flow prior to developing the mapping.

1. Facilitate reuse.

* Plan for reusable transformations upfront. Common candidates for reusable transformations are Lookups, Expressions doing common calculations or datatype conversions, Sequence Generators, and transformations calling external procedures.
* Use mapplets to encapsulate multiple reusable transformations.
* Use mapplets to leverage the work of critical developers and minimize mistakes when performing similar functions.

1. Only manipulate data that needs to be moved and transformed.

* Apply source filters in Source Qualifiers to limit the number of rows read from the sources.
* Use active transformations that reduce the number of records as early in the mapping as possible (i.e., placing filters and aggregators as close to source as possible).

1. Improve Joiners and Aggregator configuration

* Sorted Joiners and Aggregators are faster and use less cache than unsorted. Try to sort the input data by using ‘Number of Sorted Ports’ in the source qualifier, or add a Sorted Transformation (Note: adding a Sorted Transformation adds extra processing; a test should be conducted to ensure this is faster than using an unsorted input for the Joiner or Aggregator).
* Select the appropriate driving/master table while using joins. The table with the lesser number of rows should be the driving/master table for a faster join on unsorted joiners; a sorted join should take the pipeline with the least amount of duplicates in the join key as master.

1. Utilize single-pass reads.

* Redesign mappings to utilize one Source Qualifier to populate multiple targets. This way the server reads this source only once. If you have different Source Qualifiers for the same source (i.e., one for delete and one for update/insert), the server reads the source for each Source Qualifier.
* A (sorted) self-join can be used when two pipelines originating from the same source need to be combined for further processing.

1. Be aware of performance of external calls.

* Avoid connected Stored Procedure transformations since they are a performance bottleneck.
* Limit the number of Java transformation, since each transformation creates its own runtime environment allocating large amounts of memory.
* When using a SQL transformation, consider if this can also be done using a Lookup. In general a lookup performs better than SQL transformations, especially if the Lookup has caching enabled.

1. Utilize Pushdown Optimization.

* Design mappings so they can take advantage of the Pushdown Optimization feature. This may improve performance by allowing the source and/or target database to perform the mapping logic.

## Operations and Expression Optimization Tips

* Use the Source Filter and Sorted Ports instead of the SQL override in Source Qualifiers. This ensures the generated SQL at runtime always matches the connected ports from the Source Qualifier.
* Numeric operations are faster than string operations.
* Optimize char-varchar comparisons (i.e., trim spaces before comparing).
* Operators are faster than functions (i.e., || vs. CONCAT).
* Optimize IIF expressions.
  + DECODE is faster and easier to read and maintain than nested IIFs.
  + Use variable ports to store intermediate results used in multiple branches of the IIF or DECODE.
* Test expression timing by replacing with a constant value if suspicion exists that the expression is a performance bottleneck.
* Use of flat files.
  + Using flat files located on the server machine loads faster than a database located in the server machine.
  + Fixed-width files are faster to load than delimited files because delimited files require extra parsing, but will require more disk space.
  + Consider using a flat file target and external loader when loading large amounts of data.
  + Complex flat files with multiple record types may better be pre-processed using PowerExchange for flat files or B2B Data Transformation.
* If working with data that is not able to return sorted data (e.g., Web Logs), consider using the Sorter transformation.
* Use a Router Transformation to separate data flows instead of multiple Filter Transformations.
* Use a Normalizer Transformation to pivot rows rather than multiple instances of the same target.
* Rejected rows from an update strategy are logged to the bad file. Consider filtering before the update strategy, instead of using DD\_REJECT, if retaining these rows is not critical because logging causes extra overhead on the engine. Choose the option in the update strategy to discard rejected rows.
* If an update override is necessary in a load, consider using a Lookup transformation just in front of the target to retrieve the primary key. The primary key update is much faster than the non-indexed lookup override.

## Lookup Transformation Optimization Guidelines

The optimal lookup configuration depends on many factors such as source table size, lookup table size, match strategy, etc. Therefore the guidelines below should be seen as general guidelines and not as laws; there are always exceptions in which these guidelines may not result in optimal performance. Informatica recommends always testing different strategies to determine the optimal solution. A solution that works well in a small test environment may not perform well in a large-scale production environment. Tests on lookup strategies should therefore be performed on production-like amounts of data.

1. In general, cache lookup table columns for those lookup tables of 500,000 rows or less. Only when the number of source rows processed in the mapping is very small (less than 1,000 rows), may uncached lookups be faster.
2. As a rule of thumb, do not cache any table over 1,000,000 rows, although many examples exist where much larger tables are cached. Try running the session with a large lookup both cached and not cached. Caching is often faster on very large lookup tables, but the time it takes to create the cache must be taken into account. Since memory is generally not an issue anymore in modern server environments, the build time of the cache is the determining factor on the decision to cache or not.
3. When using a Lookup Table Transformation, improve lookup performance by placing all conditions that use the equality operator = first in the list of conditions under the condition tab.
4. Cache only lookup tables if the number of lookup calls is more than 10 to 20 percent of the lookup table rows. For a fewer number of lookup calls, do not cache if the number of lookup table rows is large; an uncached lookup may perform better in this case.
5. Persistent caching is very useful for large caches used in multiple mappings. Persistent caches avoid building a large cache multiple times in multiple subsequent mappings. Ensure that the cache always reflects the contents of the lookup source by setting up a cache refresh strategy (i.e., rebuild the cache after updates of the lookup source have occurred).
6. Create persistent caches for reference tables with a low refresh interval. Reference tables are often used in multiple mappings so persistent caching will avoid rebuilding the cache many times.
7. For overly large lookup tables, use dynamic caching along with a persistent cache. Cache the entire table to a persistent file on the first run, enable the "update else insert" option on the dynamic cache and the engine never has to go back to the database to read data from this table. You can also partition this persistent cache at run time for further performance gains.
8. When handling multiple matches, use the "Return any matching value" setting whenever possible. Also use this setting if the lookup is being performed to determine that a match exists, but the value returned is irrelevant. The lookup creates an index based on the key ports rather than all lookup transformation ports. This simplified indexing process will reduce the cache size and may improve performance.

# How to analyze

## Source Bottlenecks

### Relational sources

If the session reads from a relational source, you can use a filter transformation, a read test mapping, or a database query to identify source bottlenecks.

**Using a Filter Transformation**.

Add a filter transformation in the mapping after each source qualifier. Set the filter condition to false so that no data is processed past the filter transformation. If the time it takes to run the new session remains about the same, then you have a source bottleneck.

**Using a Read Test Session**.

You can create a read test mapping to identify source bottlenecks. A read test mapping isolates the read query by removing any transformation logic from the mapping. Use the following steps to create a read test mapping:

1. Make a copy of the original mapping.
2. In the copied mapping, retain only the sources, source qualifiers, and any custom joins or queries.
3. Remove all transformations.
4. Connect the source qualifiers to a file target.

Use the read test mapping in a test session. If the test session performance is similar to the original session, you have a source bottleneck.

**Using a Database Query**

You can also identify source bottlenecks by executing a read query directly against the source database. To do so, perform the following steps:

* Copy the read query directly from the session log.
* Run the query against the source database with a query tool such as SQL Plus.
* Measure the query execution time and the time it takes for the query to return the first row.

If there is a long delay between the two time measurements, you have a source bottleneck.

If your session reads from a relational source and is constrained by a source bottleneck, review the following suggestions for improving performance:

* Optimize the query.
* Create tempdb as in-memory database.
* Use conditional filters.
* Increase database network packet size.
* Connect to Oracle databases using IPC protocol.

### Flat file sources

If your session reads from a flat file source, you probably do not have a read bottleneck. Tuning the line sequential buffer length to a size large enough to hold approximately four to eight rows of data at a time (for flat files) may improve performance when reading flat file sources. Also, ensure the flat file source is local to the Integration Service.

## Target Bottlenecks

### Relational Targets

The most common performance bottleneck occurs when the Integration Service writes to a target database. This type of bottleneck can easily be identified with the following procedure:

1. Make a copy of the original workflow
2. Configure the session in the test workflow to write to a flat file and run the session.
3. Read the thread statistics in session log

If session performance increases significantly when writing to a flat file, you have a write bottleneck. Consider performing the following tasks to improve performance:

* Drop indexes and key constraints
* Increase checkpoint intervals
* Use bulk loading
* Use external loading
* Minimize deadlocks
* Increase database network packet size
* Optimize target databases

### Flat file targets

If the session targets a flat file, you probably do not have a write bottleneck. If the session is writing to a SAN or a non-local file system, performance may be slower than writing to a local file system. If possible, a session can be optimized by writing to a flat file target local to the Integration Service. If the local flat file is very large, you can optimize the write process by dividing it among several physical drives.

If the SAN or non-local file system is significantly slower than the local file system, work with the appropriate network/storage group to determine if there are configuration issues within the SAN.

## Mapping Bottlenecks

If you have eliminated the reading and writing of data as bottlenecks, you may have a mapping bottleneck. Use the swap method to determine if the bottleneck is in the mapping.

Begin by adding a Filter transformation in the mapping immediately before each target definition. Set the filter condition to false so that no data is loaded into the target tables. If the time it takes to run the new session is the same as the original session, you have a mapping bottleneck. You can also use the performance details to identify mapping bottlenecks: high Rowsinlookupcache and High Errorrows counters indicate mapping bottlenecks.

Follow these steps to identify mapping bottlenecks:

**Create a test mapping without transformations**

Make a copy of the original mapping.

In the copied mapping, retain only the sources, source qualifiers, and any custom joins or queries.

Remove all transformations.

Connect the source qualifiers to the target.

**Check for High Rowsinlookupcache counters**

Multiple lookups can slow the session. You may improve session performance by locating the largest lookup tables and tuning those lookup expressions.

**Check for High Errorrows counters**

Transformation errors affect session performance. If a session has large numbers in any of the Transformation\_errorrows counters, you may improve performance by eliminating the errors.

For further details on eliminating mapping bottlenecks, refer to the Best Practice: [Tuning Mappings for Better Performance](https://velocity.informatica.com/index.php/best-practices-all/156-performance-tuning/427-m1bp215)

## Session Bottlenecks

Session performance details can be used to flag other problem areas. Create performance details by selecting Collect Performance Data in the session properties before running the session.

View the performance details through the Workflow Monitor as the session runs, or view the resulting file. The performance details provide counters about each source qualifier, target definition, and individual transformation within the mapping to help you understand session and mapping efficiency.

To view the performance details during the session run:

* Right-click the session in the Workflow Monitor.
* Choose Properties.
* Click the Properties tab in the details dialog box.

To view the resulting performance data file, look for the file session\_name.perf in the same directory as the session log and open the file in any text editor.

All transformations have basic counters that indicate the number of input row, output rows, and error rows. Source qualifiers, normalizers, and targets have additional counters indicating the efficiency of data moving into and out of buffers. Some transformations have counters specific to their functionality. When reading performance details, the first column displays the transformation name as it appears in the mapping, the second column contains the counter name, and the third column holds the resulting number or efficiency percentage.

**Low buffer input and buffer output counters**

If the BufferInput\_efficiency and BufferOutput\_efficiency counters are low for all sources and targets, increasing the session DTM buffer pool size may improve performance.

**Aggregator, Rank, and Joiner readfromdisk and writetodisk counters**

If a session contains Aggregator, Rank, or Joiner transformations, examine each Transformation\_readfromdisk and Transformation\_writetodisk counter. If these counters display any number other than zero, you can improve session performance by increasing the index and data cache sizes.

If the session performs incremental aggregation, the Aggregator\_readtodisk and writetodisk counters display a number other than zero because the Integration Service reads historical aggregate data from the local disk during the session and writes to disk when saving historical data. Evaluate the incremental Aggregator\_readtodisk and writetodisk counters during the session. If the counters show any numbers other than zero during the session run, you can increase performance by tuning the index and data cache sizes.

**Note:** PowerCenter versions 6.x and above include the ability to assign memory allocation per object. In versions earlier than 6.x, aggregators, ranks, and joiners were assigned at a global/session level.

For further details on eliminating session bottlenecks, refer to the Best Practice: [Tuning Sessions for Better Performance](https://velocity.informatica.com/index.php/best-practices-all/156-performance-tuning/428-m1bp219) and [Tuning SQL Overrides and Environment for Better Performance](https://velocity.informatica.com/index.php/best-practices-all/156-performance-tuning/429-m1bp288).

## System Bottlenecks

After tuning the source, target, mapping, and session, you may also consider tuning the system hosting the Integration Service.

The Integration Service uses system resources to process transformations, session execution, and the reading and writing of data. The Integration Service also uses system memory for other data tasks such as creating aggregator, joiner, rank, and lookup table caches.

You can use system performance monitoring tools to monitor the amount of system resources the Server uses and identify system bottlenecks.

* **Windows Server**. Use system tools such as the Performance and Processes tab in the Task Manager to view CPU usage and total memory usage. You can also view more detailed performance information by using the Performance Monitor in the Administrative Tools on Windows.
* **UNIX**. Use the following system tools to monitor system performance and identify system bottlenecks:
  + lsattr -E -I sys0 - To view current system settings
  + iostat - To monitor loading operation for every disk attached to the database server
  + vmstat or sar w - To monitor disk swapping actions
  + sar u - To monitor CPU loading.