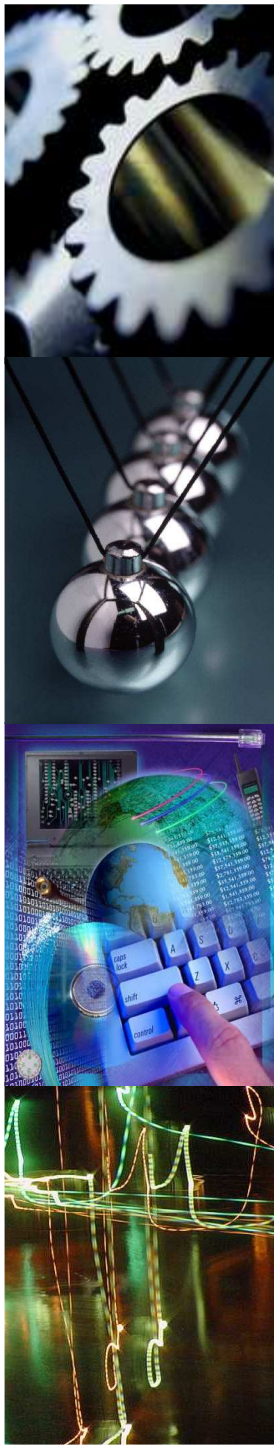


instant business intelligence

What you need, when you need it

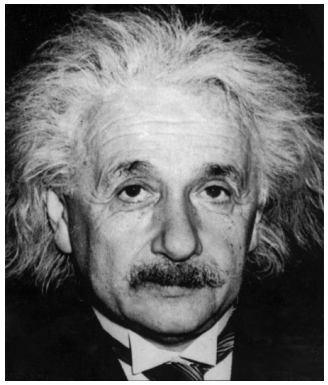
SeETL
in a Large Scale Environment

Public Information
Version 3.1: 1/1/2012

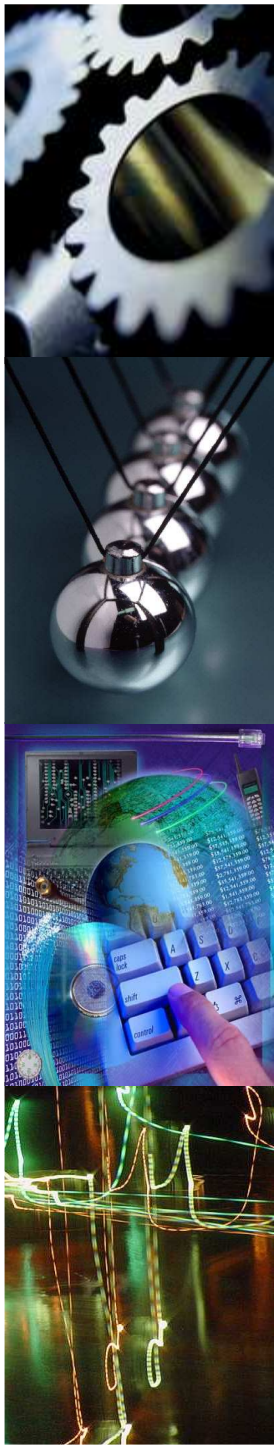
instant BUSINESS INTELLIGENCE

Why Instant Business Intelligence?

"The significant challenges we face today cannot be resolved by the same level of thinking that created them."



Albert Einstein



Introduction

instant BUSINESS INTELLIGENCE

Introduction

The early versions of the SeETL were in Cobol and they were deployed on mainframes or the largest UNIX machines available at the time, Sequent NUMA-Qs.

One of the key advantages of performing dimensional model ETL on mainframes was that there were a very small number of very fast engines available to you. And each engine looked at 'memory' as if there was 'plenty of it'. So the process of attribution, translating real keys with integer keys, could be sped up by loading as much data into memory as possible and binary searching the arrays of lookup tables. We had to be careful not to waste memory but we could reliably build extremely large data warehouses without being too concerned about how much memory was being used for the attribution process.

With the move to 'commodity' UNIX machines like Sun, HP, IBM and windows servers we have the situation of being able to have lots of small processors for much less money than mainframes. However, for ETL processing this comes at a significant cost with some significant restrictions for large companies.

It has become very difficult for companies with large amounts of data to write their own ETL code to maximum efficiency because of the complexities of trying to get around the issue of attribution and the use of memory across many processors and running processes.

Indeed, only the very highest end ETL tools provided some sort of support for this kind of processing and then only at a significant cost.

SeETL has been upgraded to support shared memory between attribution processes on both windows 2003+ and UNIX/Linux machines.

This ability to share memory delivers unlimited scalability at an extremely low cost which can be easily implemented by any company. Gone are the days where if you had a large amount of data you had to spend a large amount of money to get an ETL tool to make use of all those small processors.

SeETL also has features which will allow the attribution process to run on a machine other than the target data warehouse machine. This provides the ability to avoid database licenses to support the attribution processing.

This presentation describes in some detail

- The issues of attribution processing
- The issues of memory management for large clients
- The issues of database licensing for large clients

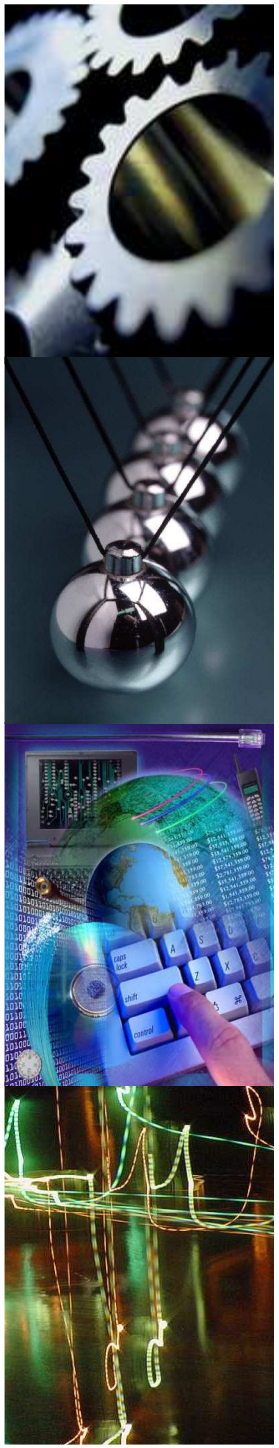
And how SeETL can meet these challenges for a fraction of the cost of any other tools.

We hope you enjoy reading this paper.

Best Regards

The Development Team

Instant Business Intelligence



Problem Definition

instant BUSINESS INTELLIGENCE

Problem Definition

Problem Definition

The problems of performing the attribution process for a dimensional model can be summarised into three main points. Of course there are more, but these are the ones we want to discuss in this presentation.

- Ability to use memory efficiently for large lookup tables.
- Ability to use all the processing power available.
- The cost of database licenses required to support the attribution processing.

Before we discuss each problem in detail let us describe briefly what the attribution process is.

What is the Attribution Process?

It is assumed the reader knows what we mean by 'the attribution process'. You will not know the problems of the attribution process if you do not know what the process is. However, just in case you are new to this and have not heard the term before let us just define the 'attribution process' in the ETL for a dimensional model.

One of the key activities in the ETL for a dimensional model is the translation of 'real keys' such as customer number, account number, dates, transaction types, currencies, languages, departments names, geographical regions and the like into integers. In a dimensional model, all real keys that may be used to link between a dimension table and a fact table are translated into integers.

Let's define an example. You go to an auto teller and withdraw EUR100. Your bank has a data warehouse and your transaction is going to find its way into the data warehouse. You are customer number 'CUST1978354' and you accounts is 'CASH97836486'.

The attribution process is the process that will take the customer number 'CUST1978354' and translate it into an integer such as 1234567 and take the account number and translate and translate it into an integer such as 7654321. These two integers will then be put into the fact transaction record to be stored away in the data warehouse.

The way this process is performed is that a table is maintained with both the 'real' key and the integer key and the translation is performed through a 'lookup' process.

For those of you who are new, the problem is that many transactions are performed each day and the lookup tables can be quite large. In a large bank there might be 10M customers and 10M accounts. This process is usually too slow if performed at the database level so we prefer to load these lookup tables into memory.

Alas, this means loading 20M rows into memory. With machines that have lots of smaller processes we then need to load a copy of each table into memory for each attribution process. So if we have four attribution processes running we need to load 80M rows into memory. Pretty soon we run out of memory!!

This, in a nutshell, is the attribution process. The process of translating real keys into integer keys for all data flowing into the dimensional data warehouse.

Problem Discussion in Detail

Problem Discussion in Detail

Let us discuss each of the problems in detail.

Ability to Use Memory Efficiently

One of the biggest issues has been that even though we now have lots of cheap processing power available it has been hard to actually run many attribution processes across the processors and effectively share the memory.

Some ETL tools introduced features such as interprocess communication or shared memory. However, for the average ETL programmer without access to these expensive tools writing such code is often beyond their capability. After all, it requires call level access to system functions, usually in C/C++.

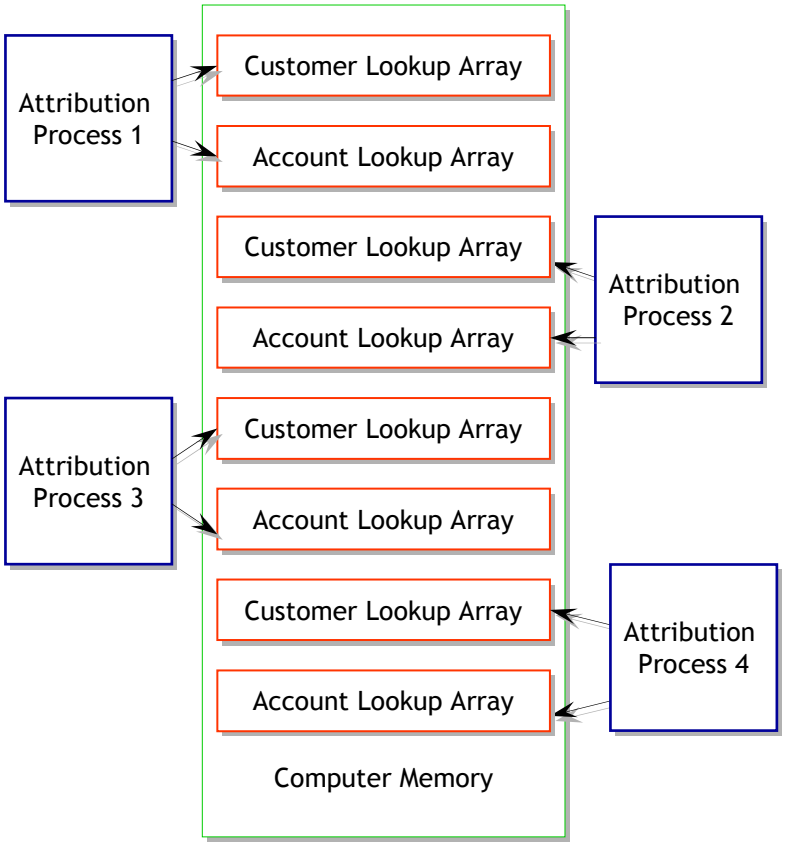
If memory is not effectively shared it is possible to create a situation as described in the diagram. The diagram represents four attribution processes running at the same time that have each loaded their own version of the customer and the account lookup tables.

Of course, if you want to run your attribution processes on a large UNIX machine and get the most out of it you would want to run 2 attribution processes per processor and that might end up loading many copies of the customer and the account lookup tables.

You can see from the diagram that with four attribution processes running we have loaded four copies of each of the account and customer lookup arrays into the memory of the machine. Effectively, we are wasting the memory for the extra three copies of the account and customer lookup arrays.

In some cases clients have more than two dimension tables that are very large. We recently worked at a client that had four large dimension tables. Each of the four dimension tables was more than 15M rows and we had to perform a lot of work to get around this problem. Even though we were using one of the worlds leading ETL tools for the production ETL processing.

How people respond to this specific problem is discussed on the next page…



instant BUSINESS INTELLIGENCE

Ability to Use Memory Efficiently (contd)

In order to avoid this waste of memory on large scale UNIX machines people implementing these solutions for themselves were forced to create much more complex and time consuming ETL processing. The most common method being to write programs to 'split' the transaction coming into the data warehouse via some mechanism that would also 'split' the largest lookup tables. The ETL programmer could then feed the 'split' file into many processors and because each attribution process knew which area of the lookup table it needed it could limit itself to loading just the portion that it needed.

In this way, ETL programmers can manually partition the attribution process to make the most efficient use of memory and processors which can maximise throughput.

The ETL System has just as many attribution processes running, and just as many in memory lookup arrays. It is just that each in memory array only contains the data that will be looked up by the attribution process because of the data that has been sent into the attribution process.

All the arrays, taken together, only contain one image of the entire customer and account lookup table each.

Clearly, such an approach requires the scanning of the input file and the splitting of the file on some basis as well as good co-ordination between all the processes to make sure all the transactions get loaded and none are 'lost' along the way.

Building this capability into an ETL system is not a simple thing to do. This becomes especially so if there are more than two very large dimension tables. The more large dimension tables that need to be split, the harder it is to make the ETL system manage all the 'split' files.

Another issue has been that the real keys coming into a data warehouse are often character strings. A great example is credit cards. Our visa cards have 16 numbers. Too long for an integer field. And credit card numbers are therefore often stored as character strings. In addition, the various real keys may actually vary in length within a single table. In the past we have usually been forced to define the 'maximum' length of a real key for a specific table and we consume that many bytes for each row whether we need them or not.

Clearly, a far better alternative to forcing ETL developers to code the attribution processing to partition memory and to consume the maximum length of real keys for each row is to:

1. Have the ETL system only ever load one image of a lookup table and share it across all attribution processes no matter how many are running
   and
2. Allow the ETL system to understand the maximum length of a real key, or define it to be some length (such as 255) but make the ETL system smart enough to only ever allocate the memory required to store the size of key that is actually in any particular row. If the key is 7 characters in one row and 21 in another the ETL system allocates 'just enough' memory in each case.

This is the approach taken by SeETL .

Problem Definition

Ability to use All the Processing Power Available

ETL developers who have not been able to implement the rather complex code of partitioning transaction files and dimension tables or the 'shared memory' code in their ETL are often forced to go back to 'let the database perform the lookup'.

There are ETL tools out there today that only allow the developer to use the database to perform the attribution process. Large companies should strike from their list of 'possible' vendors any ETL tool that does not allow in memory binary searching of lookup tables. The processing cost of performing such lookups in the database is far in excess of performing the lookup in memory. Your machines will spend all their time doing database lookups rather than something more useful. (Note. We have found Netezza can do this.)

Simply put, any solution that does not allow in memory lookups in support of the attribution process will not make efficient use of all the processing power available because it is spending all its time in the database.

Some database vendors will tell you that the database has a 'pinning' option that will 'keep the data in memory' and therefore be 'just as efficient'. This is not true.

A big problem for any company with large scale data volumes they want to place into the data warehouse is making 100% use of all the processors available doing useful, non-repetitive work.

SeETL imbeds a number of features to achieve this.

Database Licenses to Support ETL

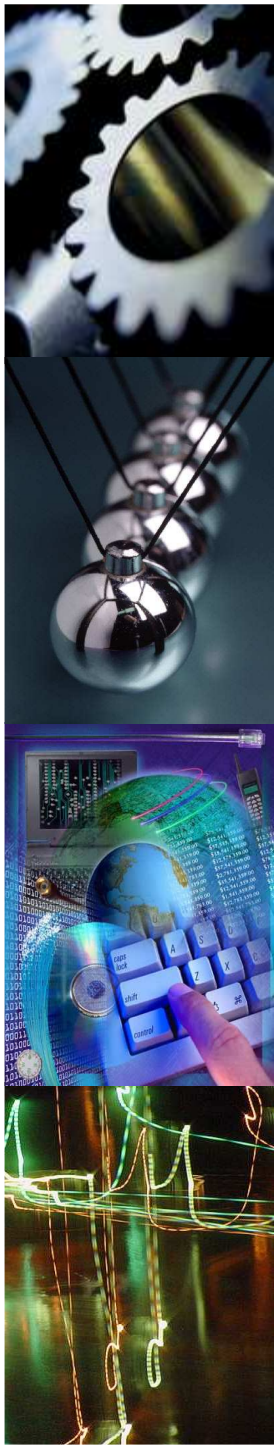
Attribution is the greatest consumer of processors and memory. To our amazement, it is closely followed by the re-summarising of data that has already been summarised. Yes, in many clients we visit we see summaries being created by re-summarising large amounts of data. In many cases we see this summarisation being performed inside the database! Amazing!!

One of the problems of pushing the workload of the attribution/ summarisation process onto the database is that you then require database licenses in order to perform all that processing. This may be ok. Your daily query load may be such that you need all those database licenses to support the query load so having them available to support the ETL processing is not an extra 'cost'. However, this is often not true in a large scale environment. In a large scale environment it is often the case that the batch window is so constrained that in order to achieve processing schedules on the time scales required extra database licenses must be bought for the ETL processing. They are then used for query processing because 'faster is always better'. But with the cost of database licenses per cpu for the leading databases buying 2, 4 or 6 CPUs worth of database licenses to speed up the ETL processing is not a 'cheap' exercise, no matter how much discount is allowed!!

A much better approach is to:

- Be able to deploy the ETL system such that the 'T' portion can be run on a machine that does not require database licenses.
- Only summarise that data which has not yet been summarised and to consolidate the newly arrived data with all the summary levels that exist in the data warehouse.

Of course, this is what SeETL does.



The Solution to the Problem

instant BUSINESS INTELLIGENCE

The Solution to the Problem

The Solution

The solution to the problems described are implemented as various features in SeETL .

The three main features which solve the problems defined are:

1. Memory mapped IO (UNIX/Linux and windows)
2. Ability to run attribution processing on a different machine to the data warehouse database.
3. Ability to aggregate 'today's' transactions and then 'consolidate in place' with summary levels from previous periods.

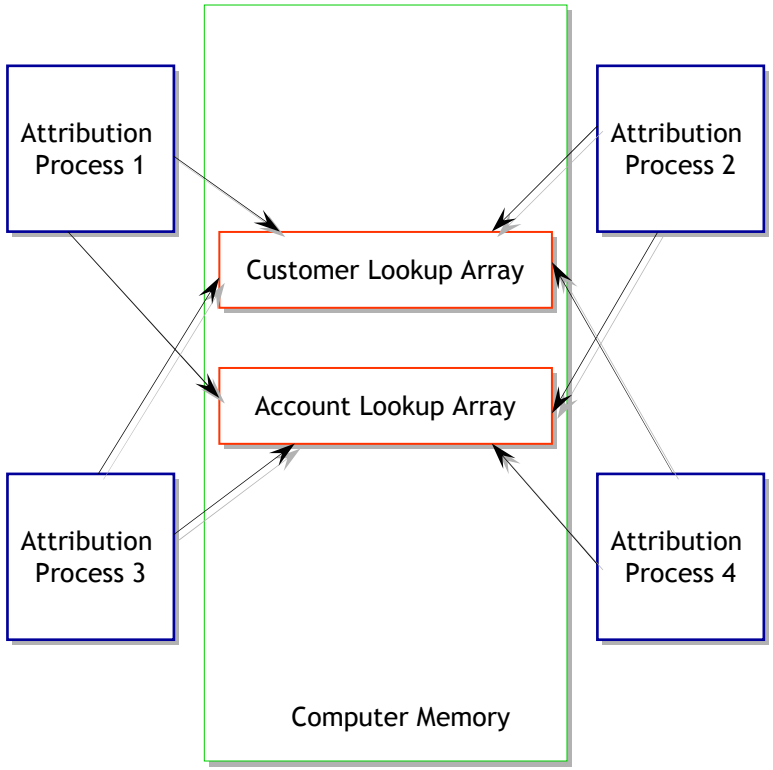
We will explain each feature in some detail here. Complete details are in the user guide.

Memory Mapped IO

Memory mapped IO is a relatively advanced programming practice that allows many running processes to share one copy of data in memory. The best way to think about it is to compare this picture with the example previously provided. In SeETL it is possible to load one copy of any (or all) of the lookup tables into memory and then for all attribution processes to access the one copy of the data in the memory of the computer. This holds true no matter how many attribution processes are running.

Indeed, the memory mapped IO is implemented in the same way as the operating system paging file in windows and in most UNIX implementations so the performance of the in memory lookups is equivalent to the best performance possible on the hardware provided!!

Clearly, this makes the greatest use of the available memory.



Another interesting feature of this style of solution is that in most UNIX solutions the fact that multiple processes are sharing the same memory does not actually slow any single one of the processes down. This enables the ETL Designer to decide how many concurrent attribution processes can run. He/she can 'load up' the machine to the maximum throughput through testing.

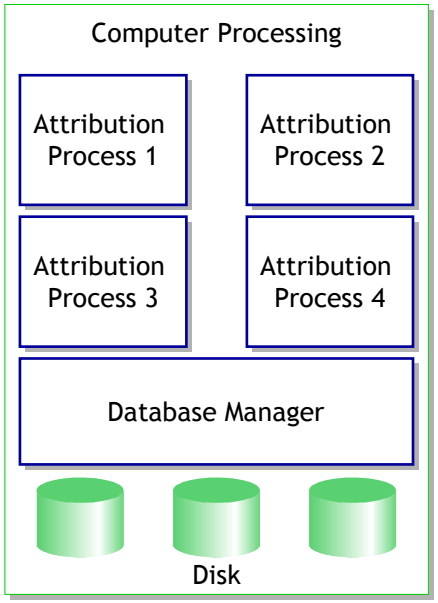
Running Attribution on a Separate Machine

Running Attribution on Separate Machine

It is common to run all ETL processing on the same machine as the data warehouse itself. Some vendors (eg Microsoft) even bundle their ETL tools with the database license.

The diagram below shows an example of how an ETL System might be running four attribution processes on the same machine as the database manager which is attached to the disk that is storing the data warehouse.

Some ETL vendors recognise that there are situations where it is desirable to run the ETL System on a separate physical machine, such as when the ETL tool will be used as a 'hub' to pump data 'between' many applications rather than to just load one large data warehouse.



In these cases they have included some features that can assist in doing so, some more effectively than others.

Another issue pushing some companies in the direction of separating the ETL System and the data warehouse database is the fact that the ETL tools are often licensed only on the basis of all the CPUs visible to the operating system. So if the ETL Designer wants to use four CPUs for the ETL tool and four CPUs for the database work he/she cannot necessarily use an eight CPU machine and make the ETL tool only use four CPUs. For some ETL tools he/she will be required to buy an eight CPU license even though four of those CPUs will be busy doing database work!!

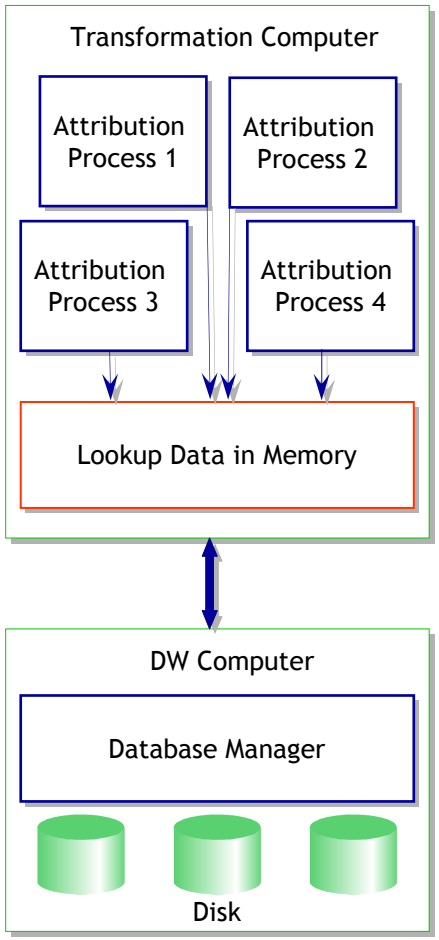
Lastly, moving the 'T' portion of 'ETL' to a different machine is not as simple as just moving the processing. There are a number of problems in doing so:

1. Some ETL tools are very 'chatty' with the database and tend to perform poorly when they are put onto a separate machine and need to 'chat' over the LAN.
2. The scheduling of ETL requires all three components (ETL) and usually the scheduler is in the ETL tool. If the ETL contains the scheduler it is necessary to fire the load job from the ETL machine against the data warehouse machine. It would be far nicer to hand the data off to the data warehouse machine and have the scheduler running on that machine load the data. However this would require another copy of the ETL tool just for the scheduler!
3. If the ETL is advanced enough to support memory mapped IO it must load the large dimension tables from the database machine across the LAN. If there is a failure during processing some ETL tools require that the data be re-loaded across the LAN consuming valuable running time.

Running Attribution on a Separate Machine

Running Attribution on Separate Machine (contd)

SeETL solves all these problems ingeniously. The utility to load data into memory mapped IO can be run independently to the starting up of the attribution processes. Indeed, the attribution processes can be made dependent on the loading of lookup data into memory mapped IO.



Then, once the data is loaded into memory, the attribution processing can be started. During attribution processing, no processing is required on the DW Computer. It is idle as far as the ETL System is concerned.

The lookup data that is in memory is not flushed until all attribution processes have concluded and that flushing is at the control of the ETL Administrator. Some of the most advanced ETL tools that can support data in memory only provide the option that the first attribution process to use the lookup table loads it for all others. (SeETL provides this option in windows too.) However, if there is a problem and the processing fails the memory mapped io files will be deleted. With SeETL , these files will be retained and loaded into virtual memory until the attribution process ends or the ETL Administrator tells SeETL to delete the memory mapped files and release the memory. In this way, SeETL can avoid having to reload the lookup tables across the LAN.

Indeed, because SeETL is licensed as source code the user can put the staging area on one machine, the transformation processing on another and the data warehouse itself on yet another machine. In fact, the staging area and the target DW database can be different flavours of database. You may choose MySQL as the staging area database because it is free and you don't query the staging area and you may deploy the data warehouse on a higher performance data warehousing database.

The point is: SeETL strongly supports the capability of running attribution processing on a different computer to the data warehouse thus giving you the choice of where to place the attribution processing.

Consolidating Aggregations in Place

Consolidating Aggregations in Place

Although the attribution process may be run on a different computer the aggregation and consolidation process is recommended to be run on the data warehouse computer. To enable this the output from the attribution processes can be copied to the data warehouse computer as a part of the schedule. We typically recommend attribution only ever occurs at the most detailed level.

When the attributed file arrives on the DW machine the aggregation processing creates the aggregates defined in a control table for those transactions that have occurred today and are being attributed today. You might create weekly, monthly and/or quarterly aggregate levels out of today's data.

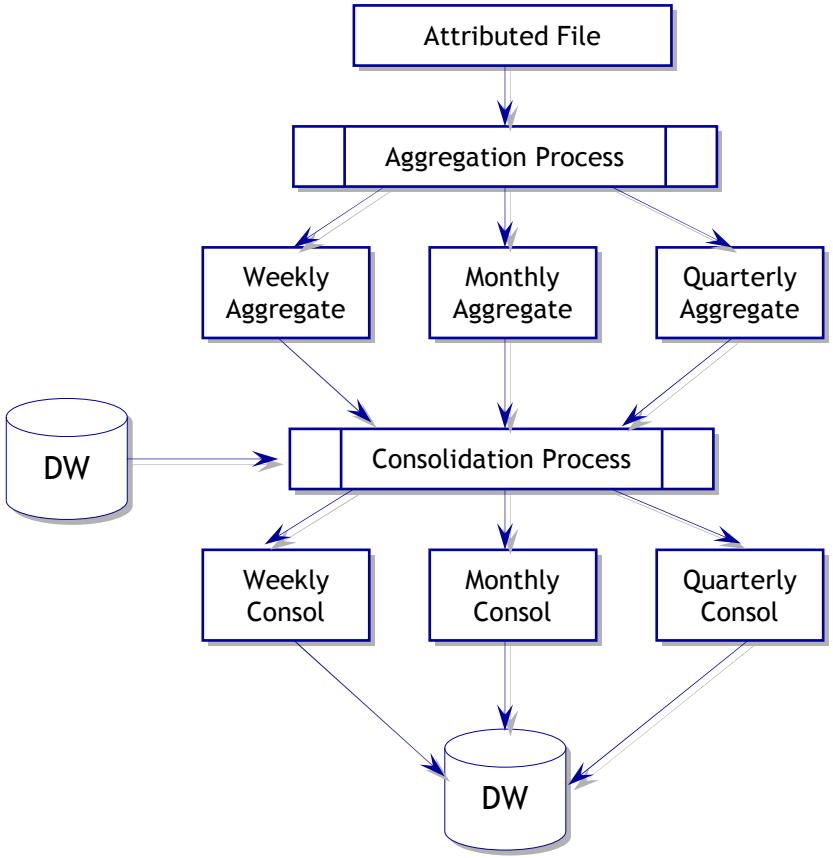
However, it is clear that if I run updates daily (or even more frequently) that when I create weekly, monthly and quarterly aggregates then rows from today's data will need to be consolidated with data already in the data warehouse.

To do this the consolidation process reads each of the aggregated records coming from the aggregation process and looks up the data warehouse to see if today's row is 'new' or whether it needs to be added to (consolidated) with an existing row in the data warehouse.

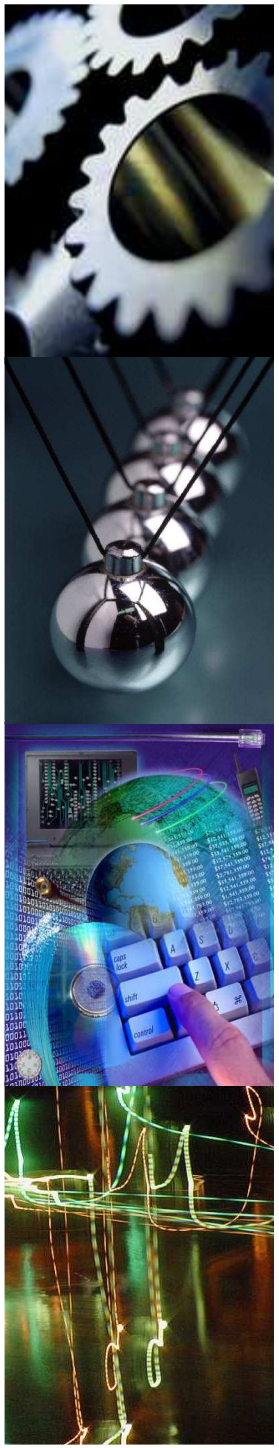
The consolidation process then sends to the output file the 'consolidated' information for each level of aggregation.

From the diagram on the right you can see that the 'aggregation' process only ever sees today's attributed file and that the aggregation process never re-summarises data that is already stored in the data warehouse.

This incremental aggregation and consolidation in place is a key feature of SeETL .



It eliminates the need for large amounts of processing to maintain the aggregates required in a large scale environment. Also, unlike other ETL tools, it is entirely parameter driven. There are no tables to create and no code to change to add a new level of aggregate to an SeETL maintained DW. It is all taken care of for you!!



Summary

instant BUSINESS INTELLIGENCE

Summary

This brief paper has talked about some of the top issues when developing a data warehouse in a large scale environment.

- Ability to use memory efficiently for large lookup tables.
- Ability to use all the processing power available.
- The cost of database licenses required to support the attribution processing.

Of course, these are not all the issues, however, our experience is they are very important. Important enough to cause you serious difficulty in a large scale environment if you are not careful!!

Because we have been working in large scale environments for 30+ years we are very familiar with how to get the best out of them.

We have added features to SeETL to:

- Allow it to use memory as efficiently as possible through a unique and advanced implementation of memory mapped IO to share memory between processes.
- Maximise processor usage by being able to run as many attribution processes as you want because of the memory mapped IO. It is also possible to split input files and run portions of SeETL in parallel.
- Minimise database license requirements by allowing the 'T' portion of ETL to run on a different machine to the data warehouse.
- We have even provided the lowest possible cost maintenance of aggregates.

We hope you have enjoyed this 'brief tour' through some of the most important features required in an ETL tool to be able to support large scale data warehousing requirements.

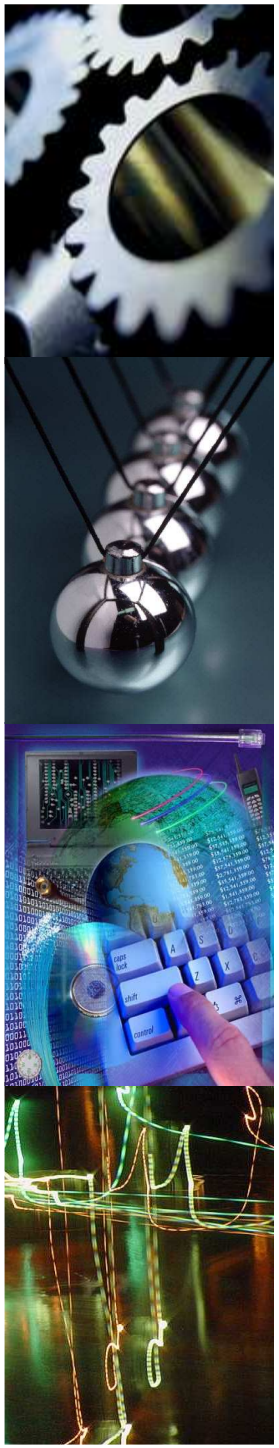
We know that through innovative programming and licensing agreements SeETL is now ready to take on even the largest ETL processing loads.

We look forward to discussing your processing requirements and showing you how SeETL can meet your needs at a fraction of the cost of any other tools.

Best Regards

The Development Team

Instant Business Intelligence



Thank You For Your Time!

instant BUSINESS INTELLIGENCE