Paper 120-2010

Analytical ETL or How to Build a DataPipe

Matthias Kehder, Modern Analytics, San Diego, CA

Abstract

A DataPipe or analytical ETL implementation gives organizations affordable, sustainable and easy-to-use analytical solutions. A DataPipe integrates data from a variety of sources across an organization and provides a self-service data kiosk from which predictive models, forecasts, OLAP cubes and reports can be created. Analytical ETL ensures the predictive needs of an organization can be met, predictive model samples can be easily modified, and all predictive samples are based on one data source – or one version of the truth.

Other key requirements of successful analytical ETL are the abilities: to easily integrate new data sources, produce a reliable and scalable platform based on standardized, business-user approved quantitative variables, and can perform scheduled scoring.

With a proper DataPipe in place, organizations can leverage multiple powerful analytical methods to improve accuracy; can seamlessly integrate Business Intelligence tools; provide many views of the organization (customer, branch, store, teller, etc.); and in addition, will collect and protect intellectual property pertinent and crucial to an organization.

Analytical ETL – the basics

Unlike tradition ETL (Extract, Transform and Load) processes, which link transactional data from such systems as Points-of-Sales (POS) into central databases, analytical ETL leverages those central databases to link the combined and cleaned central warehouse data to the reporting and statistical processes of an organization. A high-level overview of this concept is shown in figure 1.1 below.

Analytical ETL may draw on multiple databases, such as marketing and sales instances, 3rd party and/or economic data. The goal of analytical ETL is to prepare the data so it is useful and correct for the final application. Typically, when the final application is reporting, the analytical ETL (DataPipe) requirements are easier. For statistical applications, especially predictive modeling, the DataPipe requirements are much more stringent. Examples for consideration in statistical predictive modeling are such common central database variables as first/last purchase dates, life-2-date invoice amounts and most recent store visited.

A proper DataPipe will have to time censor variables such as first/last purchase date; while life-2-date invoice amounts will most likely have to be dropped. The variable could be fixed, but a proper DataPipe will provide those type of variables based on the correct time windows with additional detail.

Time windows are an important concept in building DataPipes. Similar to survival analysis, statistical samples for predictive modeling (logistic, OLS, neural networks) have to make sure they are censored to future information. Future information is behavior that occurred after an event took place that is being modeled (a purchase, churn, upgrades, etc.). Consider customer A, who purchased a product in March of 2009, while customer B purchased the same product in September of 2009. When using those two customers in a predictive modeling sample, for customer A, all information until (but not including) March of 2009 may be considered – all other purchases or behavior flags that occurred after that date, will have to be dropped. Similar to customer A, customer B cannot use any purchases or tracked behavior during or after September of 2009. Using information which was created past the modeled event is considered future information and will lead to improper and faulty results and conclusions. Hence an effective analytical ETL or a DataPipe will need to be able to draw on correct time windows to achieve proper results.

Many variables can be affected by future information. Often those effects can be difficult to detect. Consider a variable that captures the last (banking) teller seen at a bank. This would be a typical variable to be part of the customer table in a centralized database. However, if common practice at the bank is to send people to talk to a churn-prevention specialist at each branch when the customer tries to close an account, then every customer who will have attrited will have seen that specialist. At the same time, no one who stayed with the bank will have any records from that same teller, except may be an occasional incident during busy times when everyone helps out at the bank. A logistic regression would pick up on an indicator with such strong correlations to the modeled event (churn) such as in the above described scenario.

Other variables, such as gender or even title are time independent; hence, they can be used without time considerations. However; as soon as variables can change over time, caution needs to be applied.

Once sampling is handled to extract the proper time windows, time building blocks will have to be decided on. Time building blocks can be any time based block, but typically are monthly, quarterly or yearly. To determine the size of the time building blocks depends on how frequent transactions occur. In banking, where transactions typically occur on a daily basis, monthly time blocks are appropriate; in retail the time blocks typically are quarterly; while in high-tech it is not uncommon to aggregate information into yearly building blocks. When creating too small time building blocks on low frequency transactions, too many sparse fields are created; while in a high volume transactional setting, using too large of time building blocks while result in blurring important detail.

The time building blocks are populated with revenue numbers and quantity purchased so that the DataPipe can easily pull the correct time windows based on each customer event for each customer. An example is to take all product categories and produce quarterly revenue numbers for each customer going back for four years. Four years of quarterly revenue numbers would yield 16 building blocks for each product category. Each building block has an inherent start date and end date; hence, if customer A buys in March of 2009, DataPipe knows which is the last quarterly building block that is permitted in the predictive modeling sample without using future information.

Besides picking the right time building blocks, a DataPipe also needs to facilitate lag. Lag is defined to be how much time is needed to make predictions and forecasts meaningful and actionable. As an example, consider churn. It is easier to predict who is going to attrite next month then it is to predict who is going to churn in 3 months. However, with a 3 month notice on likelihood to churn, an organization can try to prevent the attrition. On the other hand, with 1 month notice (or less), an organization does not have many choices in preventing the attrition other than predicting the churn. Hence, for each predictive modeling sample, a DataPipe needs to also be able to subtract the lag from the event date to determine the appropriate time window to base the sample on. In the above example of customer A purchasing in March of 2009, marketing may want to give itself a 3 month window to execute its programs to find additional people who are going to be doing a purchase. So instead of using February of 2009 as the last (or most recent) permitable time building block, the last (or most recent) permitable time building block would be November of 2008.

An example for time window and lag considerations would be tenure, a frequently used variable in modeling exercises. Like life-2-date invoice amount, a variable often found in customer tables, tenure cannot be calculated as the time between the first invoice date and today. Tenure has to be calculated as the time between the first invoice date and the time of the event, including the subtraction of the lag. Extending the example above, modeling customer A with a three month lag on a purchase date of March 2009, and an initial invoice date of January 2008, tenure would not be two years and 3 months (as of today, March of 2010), nor would the correct answer be one year and 2 months (February of 2009), but 11 months (November of 2008).

Intermediate Layer

All of the above information will have to be consolidated into the intermediate layer of the DataPipe. The intermediate layer can be very wide, but consists of one record per entity (customer). This means all the normalizations that were put into the central database will have to be de-normalized; revenues will have to be aggregated by time building blocks and product categories; and all crosssectional variables will have be investigated to see if they integrate time. Figure 1.2 shows a graphical form of the intermediate layer.

A well structured intermediate layer will also support many sampling schemes such as time based, random or seasonal samples.

Event History

The event history file contains all events of interest to the organization. This may be first purchases, most recent purchases, attrition dates, default dates, etc. The event history file, coupled with the intermediate layer allows for merging of the intermediate layer to the event history to create the final predictive modeling samples needed for each predictive modeling project. The event history file contains a date of interest for each event for each customer, while the intermediate layer contains all the necessary building blocks to pull a sample that is free of future information and sampled with the appropriate lag in place to make the predictive models functional from an organizational perspective. Figure 1.2 shows a graphical form of this merge.

Analytical Layer

With the merging of the intermediate layer to the event history completed and the predictive modeling samples pulled, the DataPipe is now ready to apply analytical functions to samples. Since the sample contains all kinds of revenue and product purchases information in a monthly/quarterly/yearly format, it is easy to create current year versus previous year variables, slopes of the revenues over the past year(s), and/or any other kind of quantitative information that may help in predicting the events of interest. Figure 1.2 below highlights how these calculations may look like.

What is important is that the analytical functions are a set of functions to which every analyst contributes and has access to. This has two important characteristics. Firstly, as one analyst discovers a predictive quantitative criterion and encodes it into the analytical layer, all other samples will also be enriched with that insight. Secondly, the quantitative insight gained is now part of the corporate intelligence or the corporate analytical repository. This means that as analysts come and go, their contributed knowledge stays, making the organization smarter as time goes by.

In addition, the end-users and often the sponsors of the predictive models, will be familiar with the analytical variables and hence model presentations will become quite easy as the business users will be familiar with the types of analytical variables that are being used in the modeling processes.

Scoring Universe

The intermediate layer in its raw state without sampling based on time is also the most current “view” of the organization’s customer base and hence serves as the scoring universe.

The intermediate layer is also the underlying table on which to base most OLAP cubes and reporting.

DataPipe

With a complete DataPipe in place, many samples can be pulled using the same logic and processes to support hundreds and thousands of predictive models. DataPipe not only supports the sampling, but can now also accommodate scheduled and on-demand scoring, feed reports and forecasting techniques to validate trends.

When created with flexibility, a DataPipe can handle ever-changing data structures without the need to involve IT. Whether old fields are eliminated or new fields added to an existing data source, a DataPipe can easily handle it. Even completely new data sources can be easily integrated once a common key and process flow exists – eliminating the grunt work and freeing analysts to do what they enjoy – apply modeling techniques.

The overall look of a DataPipe and relevant SAS$^{©}$ tools to leverage is shown in figure 1.1 below.

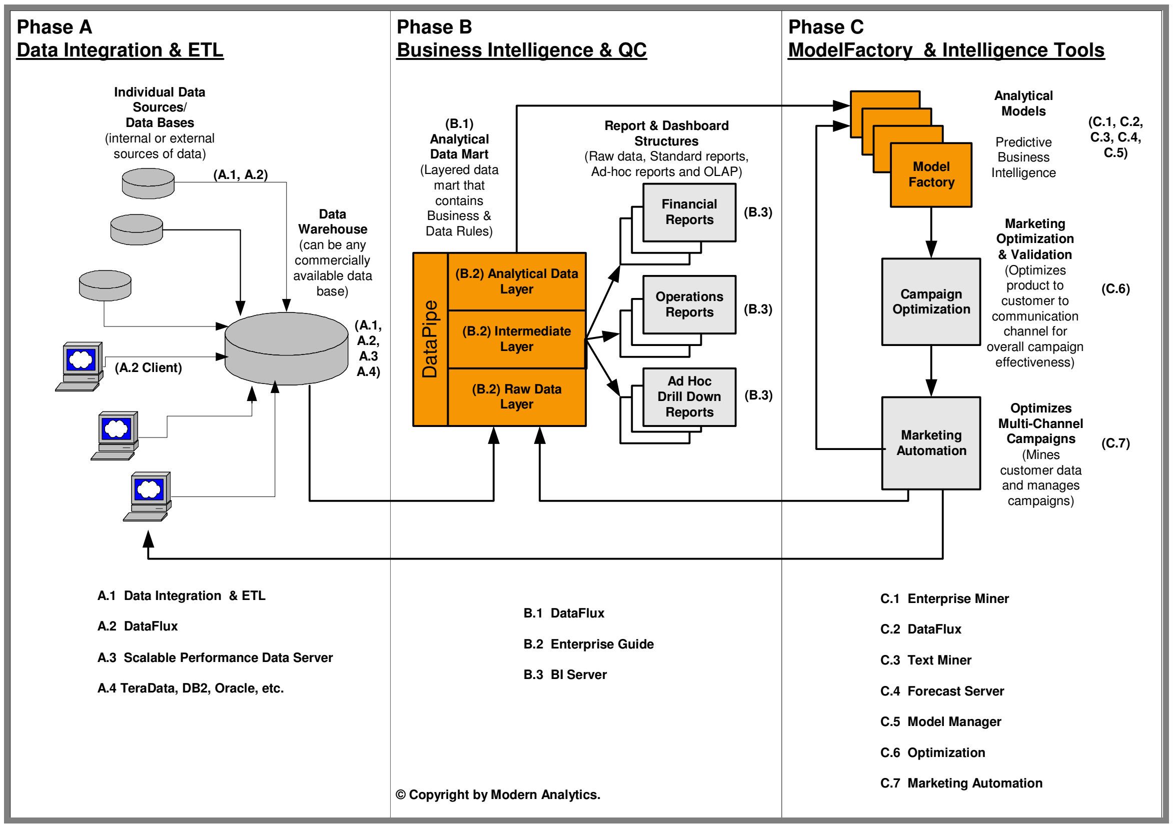
Phase A
Data Integration & ETL
Individual Data Sources/ Data Bases (internal or external sources of data)
(A.1, A.2)
Data Warehouse (can be any commercially available data base)
(A.1, A.2, A.3 A.4)
(A.2 Client)
Phase B
Business Intelligence & QC
(B.1) Analytical Data Mart (Layered data mart that contains Business & Data Rules)
DataPipe
(B.2) Analytical Data Layer
(B.2) Intermediate Layer
(B.2) Raw Data Layer
Report & Dashboard Structures (Raw data, Standard reports, Ad-hoc reports and OLAP)
Financial Reports
Operations Reports
Ad Hoc Drill Down Reports
(B.3)
(B.3)
(B.3)
Phase C
ModelFactory & Intelligence Tools
Model Factory
Analytical Models
Predictive Business Intelligence
(C.1, C.2, C.3, C.4, C.5)
Campaign Optimization
Marketing Optimization & Validation (Optimizes product to customer to communication channel for overall campaign effectiveness)
(C.6)
Marketing Automation
Optimizes Multi-Channel Campaigns (Mines customer data and manages campaigns)
(C.7)
A.1 Data Integration & ETL
A.2 DataFlux
A.3 Scalable Performance Data Server
A.4 TeraData, DB2, Oracle, etc.
B.1 DataFlux
B.2 Enterprise Guide
B.3 BI Server
C.1 Enterprise Miner
C.2 DataFlux
C.3 Text Miner
C.4 Forecast Server
C.5 Model Manager
C.6 Optimization
C.7 Marketing Automation
© Copyright by Modern Analytics.

Figure 1.1

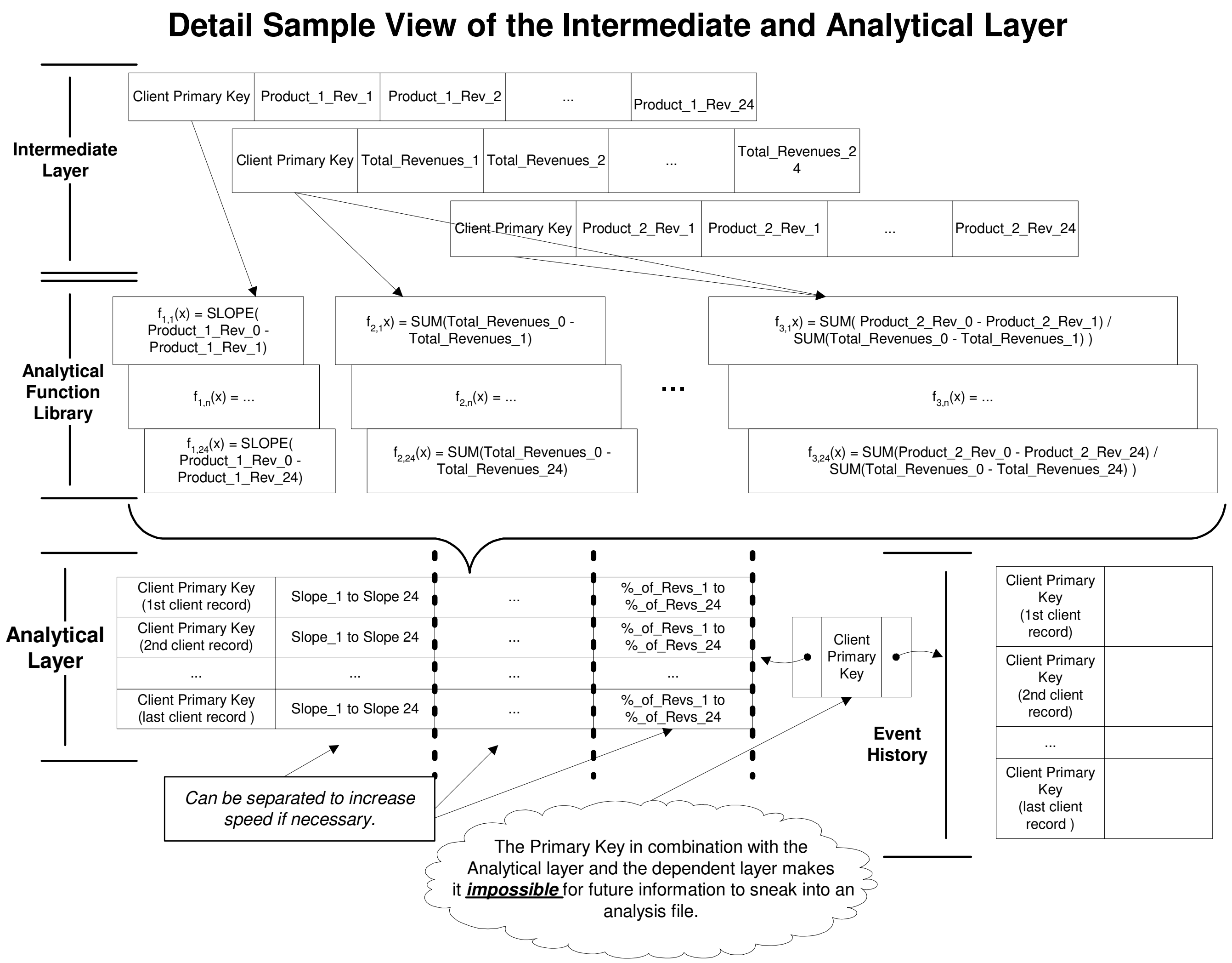
Detail Sample View of the Intermediate and Analytical Layer
Intermediate Layer
Client Primary Key | Product_1_Rev_1 | Product_1_Rev_2 | ... | Product_1_Rev_24
Client Primary Key | Total_Revenues_1 | Total_Revenues_2 | ... | Total_Revenues_24
Client Primary Key | Product_2_Rev_1 | Product_2_Rev_1 | ... | Product_2_Rev_24
Analytical Function Library
f1,1(x) = SLOPE(Product_1_Rev_0 - Product_1_Rev_1)
f1,n(x) = ...
f1,24(x) = SLOPE(Product_1_Rev_0 - Product_1_Rev_24)
f2,1x) = SUM(Total_Revenues_0 - Total_Revenues_1)
f2,n(x) = ...
f2,24(x) = SUM(Total_Revenues_0 - Total_Revenues_24)
...
f3,1x) = SUM(Product_2_Rev_0 - Product_2_Rev_1) / SUM(Total_Revenues_0 - Total_Revenues_1))
f3,n(x) = ...
f3,24(x) = SUM(Product_2_Rev_0 - Product_2_Rev_24) / SUM(Total_Revenues_0 - Total_Revenues_24))
Analytical Layer
Client Primary Key (1st client record) | Slope_1 to Slope 24 | ... | %_of_Revs_1 to %_of_Revs_24
Client Primary Key (2nd client record) | Slope_1 to Slope 24 | ... | %_of_Revs_1 to %_of_Revs_24
... | ... | ... | ...
Client Primary Key (last client record) | Slope_1 to Slope 24 | ... | %_of_Revs_1 to %_of_Revs_24
Can be separated to increase speed if necessary.
Client Primary Key
Event History
Client Primary Key (1st client record)
Client Primary Key (2nd client record)
...
Client Primary Key (last client record)
The Primary Key in combination with the Analytical layer and the dependent layer makes it impossible for future information to sneak into an analysis file.

Figure 1.2

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Name: Matthias Kehder
Enterprise: Modern Analytics
Address: 1010 Turquoise Ave, Suite 250
City, State ZIP: San Diego, CA 92109
Work Phone: 858-488-0771 x112
Fax: 858-488-0775
E-mail: mkehder@modernanalytics.com
Web: www.modernanalytics.com

SAS© and all other SAS© Institute Inc. product or service names are registered trademarks or trademarks of SAS© Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.