

**Research Report**

**Time series: From Data to Model**

Author: Bertocco Thomas-Killian - #13758322

Lecturer: Mr. Jeyaseelan Thirunavukarasu

CP5520 Advanced Databases and Applications| SP53 | 2019

6 January 2020

# Abstract

In the Data Science era, a significant number of scientific fields want to analyze their data. Often in these areas, test data are represented by time series. The latter is a class of time data, including a chronological record of values, considered as a whole and not as a list of individual and independent data. In addition, time series are generally composed of many values and can be stored in traditional databases, sometimes in enormous quantities. In this report, we analyze a way to store time series, by abstracting the series by its model (differential equation).

Table of Contents

[Abstract 2](#_Toc29156338)

[Introduction 4](#_Toc29156339)

[Context 5](#_Toc29156340)

[Essential concepts 6](#_Toc29156341)

[Representation of time series 6](#_Toc29156342)

[Time representations 6](#_Toc29156343)

[Spectral representations 6](#_Toc29156344)

[Detection of similar series 7](#_Toc29156345)

[Numerical resolution of a differential equation 7](#_Toc29156346)

[Euler's methods 7](#_Toc29156347)

[N order Runge-Kutta 8](#_Toc29156348)

[Data warehouses 8](#_Toc29156349)

[The ETL process 9](#_Toc29156350)

[The dimensional model 9](#_Toc29156351)

[The approach 9](#_Toc29156352)

[Experimental prototype 11](#_Toc29156353)

[Storage structure 11](#_Toc29156354)

[Software architecture 12](#_Toc29156355)

[Storage Space 14](#_Toc29156356)

[Execution time 15](#_Toc29156357)

[Errors 16](#_Toc29156358)

[Conclusion 16](#_Toc29156359)

[Conclusion 17](#_Toc29156360)

[References 18](#_Toc29156361)

# Introduction

The latest "advances" in information technology, hardware, high-powered deployment platforms and the maturity of data warehousing technology have motivated a significant number of industrial and research organizations to store their experiments for analysis purposes. Often, the data generated by his experiments are represented by time series.

Time series is a class of time data, including a chronological recording of values. A time series is considered as a whole and not as a list of individual and independent data (Fu ,2011; Esling and Agón, 2012). Indeed, there is a correlation between the value of a point and the values of adjacent points, so no value can be considered totally independent of the other (Shumway and Stoffer, 2015). In addition, the latter are generally composed of a large number of values, sometimes continuously updated. Time series can be represented, stored and processed in their temporal or spectral form (Shumway and Stoffer, 2015).

Time series are of interest in many fields, from medicine (electrocardiogram (Fu, 2011)) to meteorology (daily temperatures (Fu, 2011)), astrophysics (Hetland, 2004) and economics (stock and cost trends (Fink and Gandhi, 2007)). They are increasingly voluminous, and their analysis and processing have been a field of research in its own right for several years now (Fu, 2011). They can be stored in conventional databases, sometimes in vast quantities (Zeira et al, 2004) and can feed incremental learning algorithms, consisting of inferring a model from existing data and updating it with new incoming data. A given model needs context and it may be interesting to keep the history of the models.

The treatment of time series poses three significant problems: representing a series in the most optimal way in terms of storage space, with the minimum loss of accuracy compared to the original series; defining the notion of equality of two series and the notion of distances between series, for comparison purposes; finding an indexing of the elements of a series, which is not cumbersome in terms of storage space and does not introduce too much complexity in terms of calculation. In the face of this explosion of time series, the question we would like to share with the data warehousing community is as follows: Should we continue to store the data or model that generates it?

The storage of the data or model will lead to the same problems as the traditional warehouses (Vaisman and Zimanyi, 2014). However, knowledge of the model provides a better understanding of the phenomenon under study than raw data. Consequently, we propose to store the model, which can also be used to regenerate the data if necessary. One of the difficulties in substituting the data by the model is the ETL part and especially the cleaning algorithms that can use numerical analysis methods such as Euler.

Thus, in this report, we analyze a complete approach to storing time series models, while explaining the following steps: the construction of the model warehouse schema, the ETL phase, and the deployment phase.

# Context

One case study consisted of proposing a system for storing differential equations. The latter are confronted with the use of time series to monitor the evolution of the values returned by a sensor, by sampling, this type of use is also mentioned in Hetland (2004). The measured values must then be analyzed in three steps. First of all, the choice of a type of model.

It is either a differential equation or a representation of states, the choice was made to work on the differential equations. Then, a criterion must be chosen to measure the conformity of the time series with the chosen model, then the comparison between the model and the experimental data (measurement series) is carried out in order to validate or invalidate the chosen model (Yann, 2015).

*u*

FIG. 2: Representation of a model as a block diagram Yann (2015)

*y*

System

|  |  |
| --- | --- |
| TimeStamp | Value |
| 0.0000000e+00 | 0.0000000e+00 |
| 1.0000000e+00 | 9.4105346e-02 |
| 2.0000000e+00 | 1.8452077e-01 |
| 3.0000000e+00 | 2.7139095e-01 |
| 4.0000000e+00 | 3.5485491e-01 |
| 5.0000000e+00 | 4.3504619e-01 |
| 6.0000000e+00 | 5.1209313e-01 |
| 7.0000000e+00 | 5.8611902e-01 |
| 8.0000000e+00 | 6.5724231e-01 |
| 9.0000000e+00 | 7.2557682e-01 |
| 1.0000000e+01 | 7.9123189e-01 |
| 1.1000000e+01 | 8.5431259e-01 |
| 1.2000000e+01 | 9.1491986e-01 |
| 1.3000000e+01 | 9.7315068e-01 |
| 1.4000000e+01 | 1.0290982e+00 |
| 1.5000000e+01 | 1.0828521e+00 |
| 1.6000000e+01 | 1.1344982e+00 |
| … | … |

FIG. 1: Example of a time series  
(First 17 items out of 1000) Yann (2015)

Table 1 shows an example of a time series, the first column represents the time, the second the values. Figure 2 shows how a model is automatically represented, the System block corresponds to a differential equation, which allows connecting input u to output y.

We, therefore, seek to propose a means of storing differential equations, a way of abstracting experimental data by their model and comparing several series of experimental data with the models.

The purpose of the model warehouse idea is to propose an alternative representation of time series (by their model), an alternative storage solution (by storing the model) and to adapt the various data analysis and query tools in order to facilitate the management, analysis, and processing.

# Essential concepts

## Representation of time series

Time series storage and analysis is a vast field of research studied by people in databases and in recognition of shapes or patterns (Fu, 2011). Many techniques for representing and storing time series exist. These are divided into two main subtypes: temporal and spectral representations.

## Time representations

Among the temporal representations, the simplest method is to sample the series using a fixed difference between the sampling points (Fu, 2011). However, this method introduces a significant loss of information for high compression rates. For example, there is a local aggregate approximation method, called "Piecewise Aggregate Approximation" (PAA), which averages between two sampling points. An improvement to the latter method is to use a variable deviation between sampling points, called

Adaptative Piecewise Constant Approximation (APCA), to adapt this deviation to variations in the series.

Another method of compression is to search only for the extremums of the series. An improvement of this method is to keep only the most significant ones, this is a method called "Important Extrema" (IE) detailed in (Fink and Gandhi, 2007).

Time series can also be approximated by straight lines ((Keogh et al., 2001) and (Fu, 2011)), which can be constructed by linear approximation per part, where a set of consecutive points is approximated by the line connecting the first and last point of the set in chronological order, or by linear regression, which consists of approximating a set of consecutive points by a line passing at best through all points of the set. The linear regression method includes a preliminary step of searching for the Perceptually Important Points of the series.

A symbolic representation of the series can also be made by discretizing the series with a set of segments to which a symbol is then assigned ((Hetland, 2004) and (Fu, 2011)). The most effective known method is the symbolic approximation per part, called Symbolic Aggregate ApproXimation (SAX) (Esling and Agón, 2012), which consists in applying the local aggregate approximation method and converting the results into a character string, called symbolic (Fu, 2011).

## Spectral representations

In the spectral domain, the series undergoes a number of operations, which lead to a representation of the series according to a frequency point of view. The Discrete Fourier Transform provides a discrete spectral decomposition of a time series ((Fu, 2011) and (Shumway and Stoffer, 2015)) and it is also possible to decompose the DFT into real and imaginary parts, which are respectively the discrete cosine transform and the discrete sinus transform ((Shumway and Stoffer, 2015) and (Esling and Agón, 2012)).

More recently, the Discrete Wavelet Transform has emerged as an excellent alternative to the DFT.

## Detection of similar series

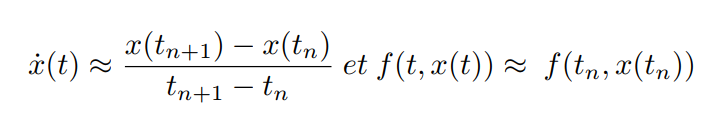
In order to detect similar series, the notion of distance between series must be defined. The choice of a distance is very dependent on the (Shumway and Stoffer, 2015) scientific domain, on which the series depends. The Euclidean distance is the most commonly used distance (Fu, 2011), but is not necessarily the most suitable for all areas. For example, it does not adapt well to the abstract notion of form, which is intuitive in humans, and therefore does not allow the detection of similar forms of evolution (notably periodicity). However, in Esling and Agón, (2012), the authors explain that for large series, the Euclidean distance remains a sufficient method of comparison.

Detection of similar series can be done by four types of methods, each requiring the definition of an appropriate distance: by comparing the overall shape of the series; by comparing the minimum number of operations necessary to transform the series into a third; by extracting the characteristics of the series to compare the characteristics with each other; by comparing on a higher-level structure. One approach is to assign a model to each series and then compare the parameters of the models with each other.

## Numerical resolution of a differential equation

There are various methods of numerical solution of differential equations (Press et al., 2002), which aim to calculate values of the solution function from an initial solution, denoted *x*0 at time *t*0. The time difference between the calculated values is fixed before the method is applied and is denoted as h. Thus we will have: *tn* = *t*0 + *n ∗ h*. We will limit ourselves here to the linear differential equations of order 1, which can be written in the form: *x*˙(*t*) = *f*(*t, x*(*t*)).

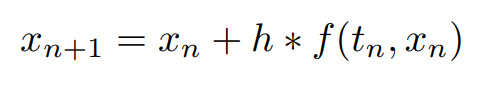
### Euler's methods

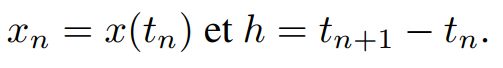
Explicit Euler The explicit Euler method is based on the following approximations (equation 1):

for t in the vicinity of tn.

Thus, using the differential equation:

This gives the formula for the explicit Euler method (equation 2):





With:

**Implicit Euler** The implicit version of the Euler method consists in taking t in the neighborhood of *tn*+1 leading to the approximation of *f*(*t, x*(*t*)) by *f*(*tn*+1*, xn*+1). This gives the formula of the implicit Euler's method (equation 3):



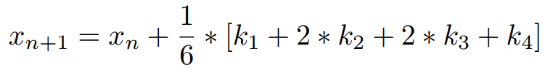
The method is said to be implicit, because the calculation of xn+1 depends on itself.

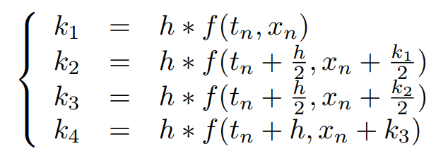
### N order Runge-Kutta

The first-order Runge-Kutta method is equivalent to the explicit Euler method. Higher-order methods consist in using intermediate points for the calculation of *xn*+1 from *xn*. The Runge-Kutta method of order 2 is also called the mid-point method, since the calculation of *xn*+1 is based on an evaluation of *xn*+1/2 with: *xn*+1/2 = *x*(*tn* + *h*2 ).

By using additional steps, the order of the error on the calculated values can be reduced. The explicit Euler method (or Runge-Kutta of order 1) is said to be of order 1 because the error committed *en* = *xn^th - xn* is an O(*h*2), where *xn^th* is the value that *x* would take at point tn if it were possible to solve the equation and obtain the explicit formula of *x* over its domain of definition. In general, a method is said to be of order *N*, when the error *en* is an *O*(*hN*+1).

Thus, Runge-Kutta methods of order higher than 1 have a better accuracy than the explicit Euler method and are more recommended in practice. However, they require more computation at each step than the first-order method. For example, the fourth-order Runge-Kutta method is the application of the following formula (equation 4):



With

This method requires four f evaluations, whereas only one is required for the first-order method. Thus, the order of the method decreases the order of the error, but increases the theoretical complexity of the calculations.

## Data warehouses

### The ETL process

Figure 3 is a simplified version of the diagram given by Elliott (2013). This schema is the general structure of a data warehouse. The ETL process plays an essential role in a data warehouse, as it is designed to read data from sources and extract information useful to the warehouse. It then performs a set of transformation operations that may vary depending on the data sources to be processed ((Elliott, 2013) and (Vassiliadis et al., 2009)). The role of transformations can be to correct errors, resolve conflicts, filter (or clean up) data, put data into warehouse formats, or remove duplicates.

Client-side applications

ETL

DBMS

Sources

FIG. 3: Simplified diagram of an ESD

### The dimensional model

Dimensional modeling, introduced by Ralph Kimball in the 1990s (Adamson, 2006), was created to improve query performance, including facilitating navigability in the model. It thus makes it possible to process large amounts of data ((Ballard et al., 2006) and (Elliott, 2013)). This model also makes it possible to describe the context of a data, for its representation to be as complete as possible. The star schema is an instance of the dimensional model that allows storage in a database.

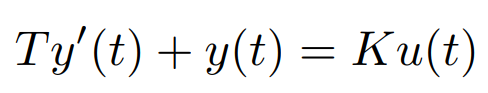
# The approach

We, therefore, seek to extend the notion of data warehouse to that of model warehouse, using differential equations as an example of a study.

The use of dimensional modeling through the star schema allows to store equations and also a certain number of information that revolve around them using dimensions (if it is a model from the analysis of experimental data, the date, the context, the parameters such as constants, the time intervals over which the equation is considered...). In addition, the same equation can be used to represent several different time series, if they differ in their time step, or the time interval considered, or if it is an equation with parameters. In this representation, a time series is an equation necessarily associated with its context.

Thus, a differential equation will be at least defined by its explicit formula, as well as the data necessary for its approximation (Yann, 2015). Indeed, to obtain the corresponding time series, it is necessary to recalculate the values from the equation using a numerical calculation algorithm, those used for the study in (Yann, 2015) being explicit Euler and Runge-Kutta of order 4.

Let us quote the example used by Yann (2015) with a reasonably simple equation such as (equation 5):



Where *T*, *K*, *u*(*t*) and *y*(*t*) are, respectively, the time constant, gain, system input, and output. This equation must be accompanied by a data system comprising: a value of *T*; a value of *K*; a definition of *u*(*t*); a time origin value *t*0 of the time series; an initial value *x*0; a time step.

Thus, several time series can result from the same equation, if they differ only by the associated dataset.

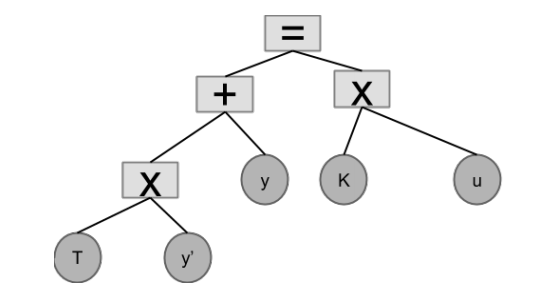


FIG. 4: Storage tree of equation 5

The abstraction of a time series by its model offers a possibility of reducing the amount of data needed for its potentially more powerful storage than other representations. This makes it possible to store more information in a small storage space. This system suffers from a loss of accuracy in the stored data. Indeed, the series regenerated using its model is not necessarily strictly equal to the basic series. However, other representations, whether temporal or spectral, also pose the same kind of problems and require consideration of the errors made. A segmentation algorithm for example (sliding window (Keogh et al., 2001)) is able to approximate a portion of a time series by a line. The portion is initially two consecutive points, then increase point by point as long as the sum of the errors is less than a threshold value. In the case of equation storage, the differences between the regenerated series and the starting series are due, first, to the error made when the series is abstracted (the solution of the equation is not strictly equal to the starting series) and to the error made when applying the numerical algorithms (the calculated values are not strictly equal).

A second notable problem is the cost of calculating to regenerate a time series. It is, in fact, much more important than for the other (Yann methods, 2015). Moreover, the complexity of the calculations, as well as their accuracy, will depend on the numerical algorithm used. On the other hand, data warehouses allow nothing to be deleted in order to keep a history of data (Inmon, 2002). Thus, the storage system could also be an incremental learning medium, or a prediction aid, and indicate a time range within which the model was considered valid.

# Experimental prototype

## Storage structure

Preliminary work has defined a storage structure for a differential equation. The binary tree structure was chosen because this structure allows the representation of a differential equation and the computer tools for storing and manipulating binary trees are numerous. Figure 4 is an example of representation of an equation by a binary tree, it is the tree representing equation 5. The leaves are the different terms of the equation (constants, functions. . .), while the nodes are the operators. The root always contains the operator "=".

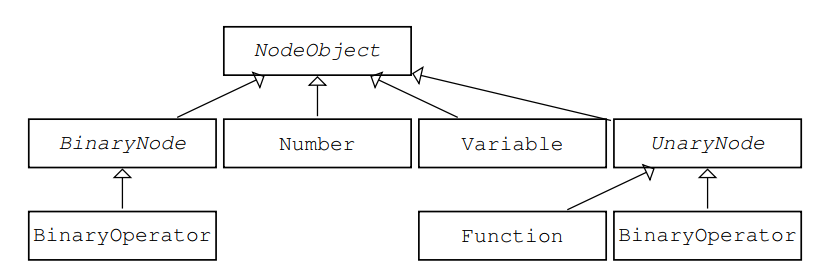


FIG. 5: Simplified UML diagram describing the nodes and leaves of a tree

Figure 5 contains a simplified UML diagram of the objects representing the nodes and leaves contained in the tree.

* The NodeObject class is abstract and is the general representation of a node.
* The classes Number and "Variable" will be sheets that represent either an invariable real number in the equation (e.g. the number 2 in 2*y’* = *y*) or a constant (such as T and K in 5).
* The class Function, represents functions (such as y and u in 5 or other classical functions, such as cosine or exponential).
* The BinaryNode and UnaryNode classes represent a node that can have two threads for the first and only one thread for the second.
* The BinaryOperator and UnaryOperator classes represent operators that can be applied to two terms (binary) or only one (unary).

A series of tests were carried out to compare the storage of time series in the form of equations and other classical representations.

The representations used for the tests are:

* Two compression algorithms, the first-named All-Extrema, consisting in retrieving the extrema of the series and the second, named Important-Extrema, being the improved version of the first;
* Two segmentation algorithms, named Bottom-Up and Slide-Window. These are two segmentation algorithms, the first of which consists in approximating the series as accurately as possible, then the segments are merged, resulting in a loss of precision. The merging process is controlled by a stopping criterion. The second algorithm consists of using a window, in which the points of the series are approximated by a straight line, and then the sum of the errors is calculated, which must be below a certain threshold. This window is then enlarged to the maximum size allowed by the threshold error;
* A direct representation, called "raw", the series is stored as is, without any modification.

## Software architecture

One of the uses of the model repository required by researchers is the comparison of a time series with pre-existing models in the database, in order to identify similar behaviors and avoid looking for a model that is already known.

The comparison activity should also verify whether an existing model allows the series to be generated by simply modifying the associated data system. To perform the comparison, a data generation activity from the model is also required. All these considerations require the addition of new operations in the classical data warehouse.

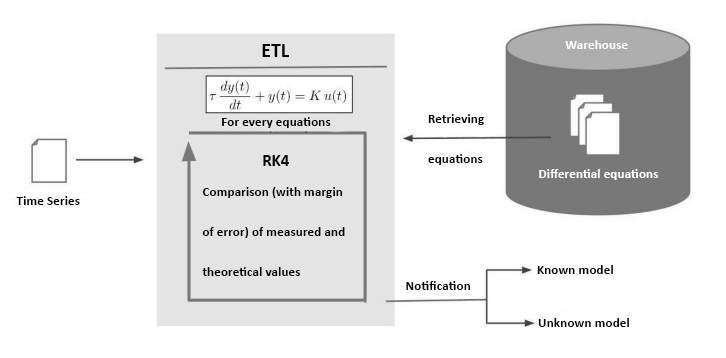


FIG. 6: Model-oriented" ETL prototype (Yann, 2015)

A generation operation, which must be able to retrieve an equation from the database and apply a numerical approximation algorithm to it; A comparison operation of two-time series or of one time series with a set of time series.

Thus, Figure 6 shows a prototype ETL process adapted to a model warehouse.

We can talk about "model-oriented" ETL processes. The process takes a time series as input. In our case, this is the raw series, but in a general case it will be one of the possible temporal or spectral representations for a time series. A "model-oriented" ETL process will therefore have to be able to handle the heterogeneity of the representations in addition to the heterogeneity problems already known from data warehouses. Then, the ETL process retrieves the set of equations contained in the warehouse.

Then, a first series is generated using the Runge-Kutta method of order 4. The generated series is compared with the input series (value-by-value comparison, with a margin of error), if they are equal, a notification is sent indicating that a pattern corresponding to the input series has been found, otherwise another series is generated with another equation.

If no equality is detected, the process notifies this as well.

Figure 7 shows the process definition made using Talend Open Studio.

A prejob is defined in order to retrieve all the equations from the warehouse. Then, the elements time series and creation\_ts retrieve the data of the series, in order to create a Java object representing them. Then the elements foreach\_DifferentialEquation and foreach\_System will iterate on the equations and for the same equation on each of the associated data systems. Finally, the ModelFound and NotFound elements are the notification processes.

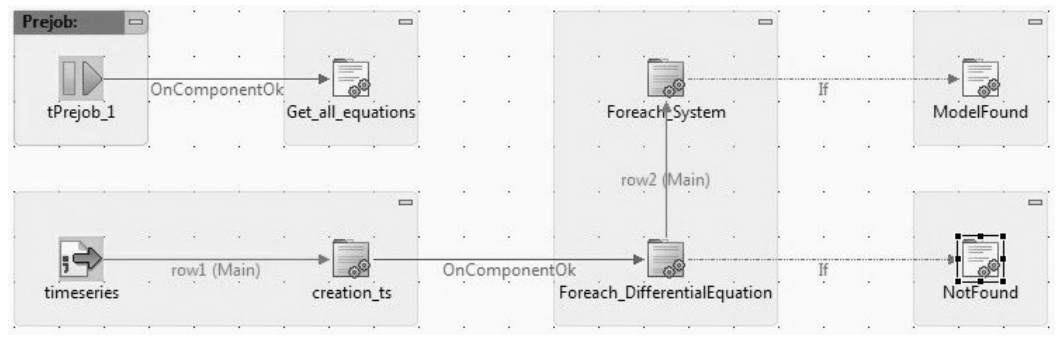


FIG. 7: Talend's definition of the "model-oriented" ETL process (Yann, 2015)

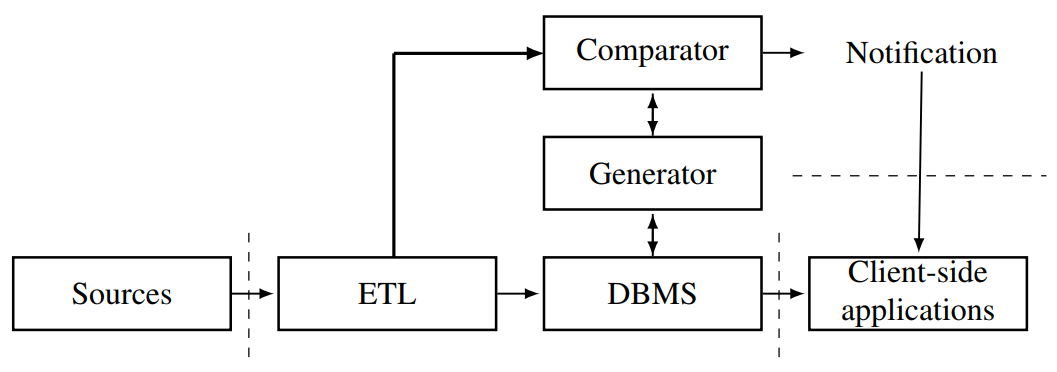


FIG. 8: Schematic of a pattern warehouse

The notion of "model-oriented" ETL induces modifications of the chain shown in figure 3, as classical ETL operations such as filtering or cleaning are not suitable for time series processing and specific comparison algorithms are needed, for example for similarity detection. Furthermore, these algorithms will need a preliminary step consisting of the regeneration of the series values from the models, involving the use of numerical algorithms.

We therefore propose the chain in Figure 8, on which the original chain has been slightly modified to take into account new processes. The chain thus created is no longer completely linear with one-way communication between the different layers, but the two-way communication has been limited to additional processes, which need to retrieve existing data from the database for the generator and to perform generation queries for the comparator. It should also be noted that the chain thus created allows communication from the ETL process to the DBMS without the need to call the comparison process. Thus, a model warehouse can be used as a data warehouse.

## Storage Space

For the tests, a series of 1,000 elements supposed to follow equation 5, was used. The series are stored in text files in CSV format with spaces as separators, while the equation is stored in a file in XML format. The XML format is particularly suitable for storing binary trees.

Table 1 shows the number of elements contained in each series according to the algorithm used. There is no column corresponding to the storage by model, knowing that in this particular case, no element of the series is stored. On the other hand, the elements are recalculated from the equation.

However, Table 2 shows in bytes the memory size required to store the series, according to its representation, with the associated compression ratio. The model storage method allows a compression rate of 97.02%, which is comparable to the rates achieved by the Important-Extrema and Sliding-Window algorithms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Unit | Crude | All-Extrema | Important-Extrema | Bottom-Up | Sliding-Window |
| None | 1000 | 73 | 36 | 74 | 28 |

TAB. 1: Number of elements after application of the algorithms (Yann, 2015)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Unit | Crude | All-Extrema | Important-Extrema | Bottom-Up | Sliding-Window | Model |
| Byte | 88208 | 6398 | 3181 | 6477 | 2540 | 2633 |
| % | 0 | 92.75 | 96.39 | 92.66 | 97.12 | 97.02 |

TAB. 2: Size in memory (Yann, 2015)

|  |  |  |  |
| --- | --- | --- | --- |
|  | All-Extrema | Bottom-Up | Slide-Window |
| u (size) | 72 | 98 | 37 |
| u (B) | 6318 | 9182 | 3260 |
| Total (B) | 8951 | 11815 | 5893 |
| Max error (%) | 0.0 | 0.0 | 32.00 |
| Avg error (%) | 0.0 | 0.0 | 16.00 |
| Compression (%) | 89.85 | 86.61 | 93.32 |

TAB. 3: Summary of model storage performance (Yann, 2015)

However, the indicated value does not take into account the storage of the u entry. This input is, in general, a known parameter of the experiment (here, it is a sum of several steps). The simplest choice, which was made in the study, is to store this input as a series, although other more optimal representations can be used. Thus, storing a differential equation takes more space than the series itself, since the input series (u) must also be stored. However, storing models still has the following advantages: several equations may share the same input; like the example used, the input may be simpler than the series itself, so compression or segmentation algorithms may achieve better compression ratios, with minimal impact on the loss of accuracy of the values (see Table 5). Moreover, Table 3 summarizes the performance obtained for storage by model, depending on the algorithm used for u. The All-Extrema algorithm allows to obtain a compression ratio of 89.85%, without any loss of precision on u.

## Execution time

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Unit | Brute | All-Extrema | Important-Extrema | Bottom-Up | Slide-Window | Model |
| Ms | 285 | 172 | 168 | 174 | 164 | 3415 |

TAB. 4: Run time for series recovery (Yann, 2015)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Brute | All-Extrema | Important-Extrema | Bottom-Up | Slide-Window | Model |
| 0 | 0.035 – 0.577 | 0.334 – 1957 | 0.248 – 0.995 | 0.458 – 1.337 | 0.001 – 0.002 |

TAB. 5: Minimum and maximum deviations (Yann, 2015)

Table 4 contains the results of an analysis of the performance in computing time required to recover the original series or an approximate version, since compression, segmentation and model abstraction algorithms induce losses compared to the original.

We notice that both compression and segmentation algorithms improve performance. Indeed, the computation times are all lower (almost halved for Sliding-Window) than the time needed to recover the raw series. On the other hand, for storage by model, the recovery time is much higher. This is due to the need to use numerical calculation algorithms.

A test has been performed with the previously described model base, containing a single model, to calculate the execution time of the "model-oriented" process defined above. If the series of inputs corresponds to the model, the calculation time is 3,485 ms, otherwise the time has decreased to 1,639 ms. As the comparison is made value by value, when a new value is generated, it is immediately compared to its corresponding value in the input series. Therefore, if the difference between the two values is greater than a threshold value, the process stops immediately. Thus, in order to detect equality between two series, all elements of the series must be generated, but in case of inequality, only the first values will be generated. Note that the 3,485 ms in the case where the series corresponds to the model are slightly higher than the 3,415 ms required for regeneration of the series alone and the time saving is quite significant when the generation is aborted. The most expensive calculation is therefore the application of the numerical calculation algorithms.

## Errors

Modifications made to a series for its storage imply a loss of precision on the values of the series. For example, when comparing the original series with the version regenerated from the stored data, the stored data no longer contain exactly the same values.

Table 5 contains the minimum and maximum deviations between the values of the original and recalculated series. Here, the use of models offers a much better approximation of the original series than other methods. However, this result depends both on the conformity of the model with the original series and on the accuracy of the numerical calculation algorithm used. For example, with a method such as Runge Kutta, increasing the order of the method provides a better approximation of the values of the solution function of the equation.

Thus, if the original series is exactly equal to said function, the approximation of the original series is better.

## Conclusion

Model storage is a priori heavier than the storage of the raw series, however, there are prospects for optimization. Also, the use of models induces a high calculation cost, which it will be necessary to study later. It seems that the loss of precision on the original series is much less, provided that the different sources of approximations (the model, the numerical calculation and the input series) are mastered.

# Conclusion

We have proposed a new vision of data warehousing that consists of substituting data with their models, motivated by the massive use of time series in several scientific fields. The concept of model warehousing and an approach to its development was presented, with an example of its use by specialists. A proof of concept was presented and also identified a problem of performance in computing time.

In terms of perspectives, we focus on the formalization of the notion of "model-oriented" ETL processes.

We propose to add new functionalities to the classical ETL process (time series analysis tools, as well as numerical calculation algorithms). In order to propose a formal definition of the "model-oriented" ETL process, we first looked at the notion of formal model of an ETL process, discussed in the following articles Skoutas and Simitsis (2007), Vassiliadis et al (2009), Muñoz et al (2010) and Vassiliadis et al (2002).

The "model-oriented" ETL process will also need to be able to manage the homogeneity of time series representations. Also, the detection of series whose behavior can be represented by the same differential equation can be performed using a similar series detection operation.

# References

Adamson, C. (2006). Mastering Data Warehouse Aggregates : Solutions for Star Schema. In C. Adamson, *Mastering Data Warehouse Aggregates : Solutions for Star Schema.* Wiley Publishing, Inc.

Ballard, F., Farrell, M., Gupta, A., Mazuela, C., & Vohnik, S. (2006). *Dimensional Modeling : In a Business Intelligence Environment (First ed.).* IBM Corp.

Elliott, R. (2013). *The Data Warehouse Toolkit (Third ed.).* Wiley Publishing,.

Esling, P., & Agon, C. (2012). Time-series data mining. *ACM Comput. Surv. 45*.

Fink, E., & Gandhi, S. (2007). Important extrema of time series. *In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 366–372.

Fu, T. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*.

Hetland, M. (2004). A survey of recent methods for efficient retrieval of similar time sequencies. *Data Mining In Time Series Databases*, pp. 23-42.

Inmon, W. (2002). *Building the Data Warehouse (Third ed.).* John Wiley & Sons, Inc.

Keogh, E. (2001). An online algorithm for segmenting time series. *Mining, In Proceedings of the 2001 IEEE International Conference on Data*, 289–296.

Munoz, L. (2010). A family of experiments to validate measures for UML activity diagrams of ETL processes in data warehouses. *Information & Software Technology 52*, pp. 1188–1203.

Press, W., Teukolsky, S., Vetterling, W., & Flannery, B. (2002). Numerical Recipes in C : The Art of Scientific Computing. Cambridge University Press.

Shumway, R., & Stoffer, D. (2015). *Time Series Analysis and It’s Applications.* Springer.

Skoutas, D., & Simitsis, A. (2007). Ontology-based conceptual design of ETL processes for both structured and semi-structured data. *Int. J. Semantic Web Inf. Syst*, pp. 1–24.

Vaisman, A., & Zimanyi, E. (2014). *Data Warehouse Systems Design and Implementation.* Springer.

Vassiliadis, P., Simitsis, A., & Baikousi, E. (2002). Conceptual modeling for etl processes. In D. Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, *Conceptual modeling for etl processes* (pp. 14-21). New York: ACM.

Vassiliadis, P., Simitsis, A., & Baikousi, E. (2009). A taxonomy of ETL activities. In *Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP, DOLAP* (pp. 25-32). New York: ACM.

Yann, A. (2016). *Model repository, application to data from the automatic.* Geneva School of Economics and Management.

Zeira, G., Maimon, O., Last, M., & Rokach, L. (2004). Change detection in classification models induced from time series data. *induced from time series data*, pp. 101-125.