Entrepôt de Données dans l'ère Data Science : De la Donnée au Modèle

Cyrille Ponchateau, Ladjel Bellatreche, Mickael Baron

LIAS/ISAE-ENSMA, Université de Poitiers
Téléport 2 - 1 avenue Clément Ader - BP 40109,
86961 Futuroscope Chasseneuil Cedex
France
firstname.lastname@ensma.fr

Résumé. Dans l'ère Data Science, un nombre important de domaines scientifiques souhaitent analyser leurs données. Souvent dans ces domaines, les données des tests sont représentées par des séries chronologiques. Ces dernières sont une classe de données temporelles, comprenant un enregistrement chronologique de valeurs, considérées comme un tout et non comme une liste de données individuelles et indépendantes. De plus, les séries chronologiques sont généralement composées d'un grand nombre de valeurs et peuvent être stockées dans des bases de données classiques, parfois en très grande quantité. Dans cet article, nous proposons un moyen de stockage des séries chronologiques, par abstraction de la série par son modèle (équation différentielle), conduisant ainsi à la notion nouvelle d'*entrepôt de modèles* ayant pour but de proposer une autre représentation des séries chronologiques et une solution alternative de stockage. Cependant, cette méthode induit un coût non-négligeable en temps de calcul. Notre proposition est implémentée et validée en utilisant des données réelles issues des expérimentations du domaine de l'automatique.

1 Introduction

Les dernières « avancées » en matière des technologies de l'information, matériels, de plateformes de déploiement de grande puissance et de la maturité de la technologie d'entreposage de données ont motivé un nombre important d'organismes industriels et de recherche afin de stocker leurs expérimentations à des fin d'analyse. Souvent, les données générées par ses expérimentations sont représentées par des séries chronologiques.

Les séries chronologiques sont une classe de données temporelles, comprenant un enregistrement chronologique de valeurs. Une série chronologique est considérée comme un tout et non comme une liste de données individuelles et indépendantes (Fu (2011); Esling et Agón (2012)). En effet, il existe une corrélation entre la valeur d'un point et les valeurs des points adjacents, aucune valeur ne peut donc être considérée comme totalement indépendante des autres Shumway et Stoffer (2015). De plus, ces dernières sont généralement composées d'un grand nombre de valeurs, parfois continuellement mises à jour. Les séries chronologiques peuvent

être représentées, stockées et traitées sous leur forme temporelle ou spectrale Shumway et Stoffer (2015).

Les séries chronologiques trouvent leur intérêt dans de nombreux domaines, de la médecine (électrocardiogramme Fu (2011)) à l'économie (évolution des stocks et des coûts Fink et Gandhi (2007)) en passant par la météorologie (températures journalières Fu (2011)) ou l'astrophysique Hetland (2004). Elles sont de plus en plus volumineuses et leur analyse et leur traitement est un domaine de recherche à part entière depuis plusieurs années Fu (2011). Elles peuvent être stockées dans des bases de données classiques, parfois en très grandes quantités Zeira et al. (2004) et peuvent alimenter des algorithmes d'apprentissage incrémental, consistant à la déduction d'un modèle à partir des données existantes et à la mise à jour de ce dernier à l'aide de nouvelles données entrantes. Un modèle donné a besoin d'un contexte et il peut être intéressant de conserver l'historique des modèles.

Le traitement des séries chronologiques pose trois problèmes majeurs qui sont Esling et Agón (2012) : représenter une série de manière la plus optimale en termes d'espace de stockage, avec le minimum de perte de précision par rapport à la série originale ; définir la notion d'égalité de deux séries et la notion de distances entre séries, à des fins de comparaison ; trouver une indexation des éléments d'une série, qui soit peu encombrante en termes d'espace de stockage et n'introduise pas trop de complexité en termes de calcul. Devant cette explosion de séries chronologiques, la question que nous aimerions partager avec la communauté d'entreposage de données est la suivante : *Devons-nous continuer à stocker la donnée ou le modèle qui la génère ?*

Le stockage de la donnée ou du modèle conduiront aux mêmes problématiques que les entrepôts classiques Vaisman et Zimanyi (2014). Cependant, la connaissance du modèle apporte une meilleure compréhension du phénomène étudié, que les données brutes. En conséquence, nous proposons de stocker le modèle, qui peut, par ailleurs, être utilisé pour régénérer la donnée en cas de besoin. Une des difficultés de substituer la donnée par le modèle est la partie ETL et surtout les algorithmes de nettoyage qui peuvent faire appel à des méthodes d'analyse numérique comme Euler.

Ainsi, nous proposons, dans cet article, une démarche complète d'entreposage des *modèles des séries chronologiques*, tout en explicitant les étapes suivantes : la construction du schéma de l'entrepôt de modèle, la phase ETL (en utilisant Talend Open Studio [1]), et la phase déploiement.

Cet article comporte les sections suivantes : la section 2 présente le contexte de l'étude. La section 3 présente d'une manière détaillée les concepts et les notions fondamentaux pour faciliter la compréhension de notre proposition. La section 4 décrit en détail notre proposition en illustrant ses différentes étapes. La section 5 présente une validation de l'approche à travers des expérimentations obtenues en utilisant des données issues de l'automatique. Enfin, la section 6 récapitule les principaux résultats et donne les perspectives à explorer.

2 Contexte

Un cas d'étude consistait à proposer un système de stockage d'équations différentielles en collaboration avec des automaticiens. Ces derniers sont confrontés à l'utilisation de séries

1. Talend Open Studio est un logiciel open source développé par la société Talend. Il permet de définir un processus ETL à l'aide d'une interface graphique : https ://fr.talend.com/

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: Exemple de série chronologique (17 premiers éléments sur 1000) Albert (2015)

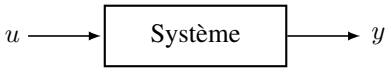

FIG. 2: Représentation d'un modèle sous forme de schéma bloc Albert (2015)

chronologique pour suivre l'évolution des valeurs renvoyées par un capteur, par échantillonnage, ce type d'utilisation est d'ailleurs mentionné dans Hetland (2004). Les valeurs mesurées doivent ensuite être analysées selon trois étapes. Tout d'abord le choix d'un type de modèle. Dans notre cas, il s'agit soit d'une équation différentielle, soit d'une représentation d'états, le choix a été fait de travailler sur les équations différentielles. Ensuite, il faut choisir un critère permettant de mesurer la conformité de la série chronologique au modèle choisi, puis la comparaison entre le modèle et les données expérimentales (séries de mesures) est réalisée afin de valider ou invalider le modèle choisi Albert (2015).

Le tableau 1 montre un exemple de série chronologique (données fournies par l'équipe d'automatique, la série complète contient mille éléments), la première colonne représente le temps, la seconde les valeurs. La figure 2 montre comment un modèle est représenté en automatique, le bloc `Système` correspond à une équation différentielle, qui permet de relier l'entrée u à la sortie y. L'équation correspondant à la série présentée est également fournie par l'équipe d'automatique (voir équation 5, section 4).

Nous cherchons donc à proposer un moyen de stockage des équations différentielles. Un moyen qui permettrait d'abstraire les données expérimentales par leur modèle et de confronter plusieurs séries de données expérimentales aux modèles.

L'idée de l'*entrepôt de modèles* a pour but de proposer une autre représentation des séries chronologiques (par leur modèle), une solution de stockage alternative (par stockage du modèle) et d'adapter les divers outils d'analyse des données et de requêtes, afin d'en faciliter la gestion, l'analyse et le traitement.

3 Notions essentielles

3.1 Représentation des séries chronologiques

Le stockage et l'analyse des séries chronologiques est un domaine de recherche très vaste étudié par les personnes de bases de données et en reconnaissance de formes ou de motifs Fu (2011). De nombreuses techniques de représentation et de stockage des séries chronologiques existent. Ces dernières se scindent en deux principaux sous-types : les représentations temporelles et spectrales.

3.1.1 Représentations temporelles

Parmi les représentations temporelles, la méthode la plus simple consiste à échantillonner la série en utilisant un écart fixe entre les points d'échantillonnage Fu (2011). Cependant cette méthode introduit une perte d'information extrêmement importante pour des taux de compression élevés. Il existe ainsi une méthode d'approximation par agrégats locaux, appelée "Piecewise Aggregate Approximation" (PAA), qui prend la moyenne entre deux points d'échantillonnage. Une amélioration de cette dernière méthode consiste à utiliser un écart variable entre les points d'échantillonnage, appelée "Adaptative Piecewise Constant Approximation" (APCA), afin d'adapter cet écart aux variations de la série. Une autre méthode de compression consiste à rechercher uniquement les points extremums de la série et une amélioration de cette méthode consiste à n'en conserver que les plus significatifs. Il s'agit d'une méthode nommée "Important Extrema" (IE) détaillée dans Fink et Gandhi (2007).

Les séries temporelles peuvent également être approximées par des lignes droites (Keogh et al. (2001) et Fu (2011)), qui peuvent être construites par approximation linéaire par partie, où un ensemble de points consécutifs est approximé par la droite reliant le premier et le dernier point de l'ensemble dans l'ordre chronologique, ou par régression linéaire, qui consiste à approximer un ensemble de points consécutif par une droite passant au mieux par tous les points de l'ensemble. La méthode par régression linéaire comprend une étape préliminaire de recherche des points remarquables de la série (Perceptually Important Points).

Une représentation symbolique de la série peut également être réalisée Hetland (2004) et Fu (2011), en discrétisant la série avec un ensemble de segments à qui un symbole est ensuite attribué. La méthode connue la plus efficace étant l'approximation symbolique par partie, nommée Symbolic Aggregate approXimation (SAX) Esling et Agón (2012), qui consiste à appliquer la méthode d'approximation par agrégats locaux et d'en convertir les résultats en une chaîne de caractères, dite symbolique Fu (2011).

3.1.2 Représentations spectrales

Dans le domaine spectral, la série subit un certain nombre d'opérations, qui conduisent à une représentation de la série selon un point de vue fréquentiel. La transformée de Fourier Discrète (Discrete Fourier Transform), permet de donner une décomposition spectrale discrète d'une série chronologique (Fu (2011) et Shumway et Stoffer (2015)) et il est également possible de décomposer la DFT en partie réelle et partie imaginaire, qui sont respectivement la transformée en cosinus discrète et la transformée en sinus discrète (Shumway et Stoffer (2015)

et Esling et Agón (2012)). Plus récemment, la transformée en ondelettes discrètes (Discrete Wavelet Tranform) s'est imposée comme une excellente alternative à la DFT.

3.2 Détection de séries similaires

Afin de détecter des séries similaires, la notion de distance entre séries doit être définie. Le choix d'une distance est très dépendante du domaine Shumway et Stoffer (2015) scientifique, dont la série dépend. La distance Euclidienne est la distance la plus utilisée Fu (2011), mais n'est pas forcément la plus adaptée à tous les domaines. Par exemple, elle s'adapte mal à la notion abstraite de forme, qui est intuitive chez l'être humain, et ne permet donc pas de détecter des formes d'évolution similaire (notamment la périodicité). Cependant, dans Esling et Agón (2012), les auteurs expliquent tout de même que pour des séries de grandes tailles, la distance Euclidienne reste une méthode de comparaison suffisante.

La détection de séries similaires peut se faire selon quatre types de méthodes, chacune nécessitant la définition d'une distance appropriée : par comparaison de la forme globale des séries ; par comparaison du nombre minimal d'opérations nécessaires pour transformer les séries en une troisième ; par extraction des caractéristiques des séries afin de comparer les caractéristiques entre elles ; par comparaison sur une structure de plus haut niveau. Une des approches est l'affectation d'un modèle à chaque série, puis par comparaison des paramètres des modèles entre eux.

3.3 Résolution numérique d'une équation différentielle

Il existe diverses méthodes de résolution numérique d'équations différentielles Press et al. (2002), qui ont pour but de calculer des valeurs de la fonction solution à partir d'une solution initiale, notée x_0 à l'instant t_0. L'écart temporel entre les valeurs calculées est fixé avant l'application de la méthode et est noté h. Ainsi on aura : $t_n = t_0 + n * h$. Nous nous limiterons ici aux équations différentielles linéaires d'ordre 1, qui peuvent être écrites sous la forme : $\dot{x}(t) = f(t, x(t))$.

3.3.1 Les méthodes d'Euler

Euler explicite La méthode d'Euler explicite repose sur les approximations suivantes :

$$\dot{x}(t) \approx \frac{x(t_{n+1}) - x(t_n)}{t_{n+1} - t_n} \; et \; f(t, x(t)) \approx \; f(t_n, x(t_n)) \tag{1}$$

pour t au voisinage de t_n.
Ainsi, on obtient, à l'aide de l'équation différentielle : $\frac{x(t_{n+1}) - x(t_n)}{t_{n+1} - t_n} \approx f(t_n, x(t_n))$
Ce qui donne la formule de la méthode d'Euler explicite :

$$x_{n+1} = x_n + h * f(t_n, x_n) \tag{2}$$

Avec : $x_n = x(t_n)$ et $h = t_{n+1} - t_n$.

Euler implicite La version implicite de la méthode d'Euler consiste à prendre t dans le voisinage de t_{n+1} conduisant à l'approximation de $f(t, x(t))$ par $f(t_{n+1}, x_{n+1})$. Ce qui donne la formule de la méthode d'Euler implicite :

$$x_{n+1} = x_n + h * f(t_{n+1}, x_{n+1}) \tag{3}$$

La méthode est dite implicite, car le calcul de x_{n+1} dépend de lui-même.

3.3.2 Runge-Kutta d'ordre N

La méthode de Runge-Kutta d'ordre 1 est équivalente à la méthode d'Euler explicite. Les méthodes d'ordre supérieur consistent à utiliser des points intermédiaires pour le calcul de x_{n+1} à partir de x_n. La méthode de Runge-Kutta d'ordre 2 est également appelée la méthode du point milieu, puisque le calcul de x_{n+1} repose sur une évaluation de $x_{n+\frac{1}{2}}$ avec : $x_{n+\frac{1}{2}} = x(t_n + \frac{h}{2})$.

L'utilisation d'étapes supplémentaires permet de réduire l'ordre de l'erreur commise sur les valeurs calculées. La méthode d'Euler explicite (ou Runge-Kutta d'ordre 1) est dite d'ordre 1 car l'erreur commise $e_n = x_n^{th} - x_n$ est un $O(h^2)$, où x_n^{th} est la valeur que prendrait x au point t_n s'il était possible de résoudre l'équation et d'obtenir la formule explicite de x sur son domaine de définition. De manière générale, une méthode est dite d'ordre N, lorsque l'erreur e_n est un $O(h^{N+1})$.

Ainsi, les méthodes de Runge-Kutta d'ordre supérieur à 1 ont une meilleure précision que la méthode d'Euler explicite et sont plus recommandées en pratique. Cependant, elles nécessitent plus de calcul à chaque étape que la méthode d'ordre 1. Par exemple, la méthode de Runge-Kutta d'ordre 4 est l'application de la formule suivante :

$$x_{n+1} = x_n + \frac{1}{6} * [k_1 + 2 * k_2 + 2 * k_3 + k_4] \tag{4}$$

avec

$$\begin{cases} k_1 &= h * f(t_n, x_n) \\ k_2 &= h * f(t_n + \frac{h}{2}, x_n + \frac{k_1}{2}) \\ k_3 &= h * f(t_n + \frac{h}{2}, x_n + \frac{k_2}{2}) \\ k_4 &= h * f(t_n + h, x_n + k_3) \end{cases}$$

Cette méthode nécessite quatre évaluations de f, alors qu'il n'en faut qu'une pour la méthode d'ordre 1. Ainsi, l'ordre de la méthode diminue l'ordre de l'erreur, mais augmente la complexité théorique des calculs.

3.4 Les entrepôts de données

3.4.1 Le processus ETL

La figure 3 reprend de façon simplifiée le schéma donnée par Elliott (2013). Ce schéma est la structure générale d'un entrepôt de données. Le processus ETL joue un rôle essentiel dans un entrepôt de données, puisqu'il a pour but de lire les données des sources et d'en extraire les informations utiles à l'entrepôt. Il effectue ensuite un ensemble d'opérations de transformation qui peuvent varier selon les sources de données à traiter (Elliott (2013) et Vassiliadis et al. (2009)). Le rôle des transformations peut être de corriger des erreurs, résoudre des problèmes

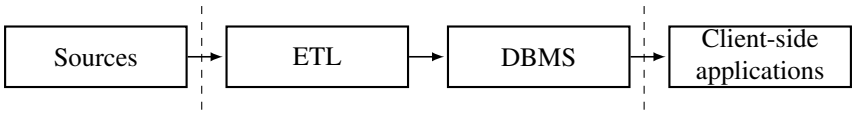

FIG. 3: Schéma simplifié d'un EDD

de conflits, filtrer (ou nettoyer) les données, mettre les données aux formats de l'entrepôt ou supprimer les doublons.

3.4.2 Le modèle dimensionnel

La modélisation dimensionnelle, introduite par Ralph Kimball dans les années 90 Adamson (2006), a été créée pour améliorer les performances des requêtes, notamment en facilitant la navigabilité dans le modèle. Elle permet ainsi de traiter de grandes quantité de données Ballard et al. (2006) et Elliott (2013). Ce modèle permet aussi de décrire le contexte d'une donnée, pour sa représentation soit la plus complète possible. Le schéma en étoile est une instance du modèle dimensionnel permettant le stockage dans une base de données.

4 Notre approche

Nous cherchons donc à étendre la notion d'entrepôt de données à celle d'*entrepôt de modèles*, en prenant les équations différentielles, comme exemple d'étude.

L'utilisation de la modélisation dimensionnelle au travers du schéma en étoile permet de stocker des équations et aussi un certain nombre d'informations qui gravitent autour à l'aide des dimensions (s'il s'agit d'un modèle issu de l'analyse de données expérimentales, la date, le contexte, les paramètres tel que les constantes, les intervalles de temps sur lesquels l'équation est considérée...). De plus, une même équation peut servir à représenter plusieurs séries chronologiques différentes, si celles-ci se différencient par leur pas de temps, ou l'intervalle de temps considéré, ou encore s'il s'agit d'une équation possédant des paramètres. Dans cette représentation, une série chronologique est une équation nécessairement associée à son contexte.

Ainsi une équation différentielle sera a minima défini par sa formule explicite, ainsi que les données nécessaires à son approximation Albert (2015). En effet, pour obtenir la série chronologique correspondante, il est nécessaire d'en recalculer les valeurs à partir de l'équation en utilisant un algorithme de calcul numérique, ceux utilisés pour l'étude dans Albert (2015) étant Euler explicite et Runge-Kutta d'ordre 4.

Citons l'exemple utilisé par Albert (2015) avec une équation assez simple telle que :

$$Ty'(t) + y(t) = Ku(t) \tag{5}$$

Où T, K, $u(t)$ et $y(t)$ sont, respectivement, la constante de temps, le gain, l'entrée et la sortie du système. Cette équation doit être accompagnée d'un système de données comprenant :

une valeur de T ; une valeur de K ; une définition de $u(t)$; une valeur t_0 d'origine temporelle de la série chronologique ; une valeur initiale x_0 ; un pas de temps.

Ainsi, plusieurs séries chronologiques peuvent découler de la même équation, si elles diffèrent uniquement par le jeu de données associé.

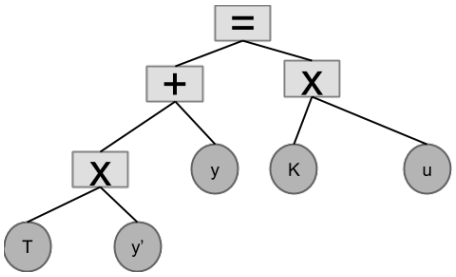

FIG. 4: Arbre de stockage de l'équation 5

L'abstraction d'une série chronologique par son modèle offre une possibilité de réduction du volume de données nécessaire à son stockage potentiellement plus puissante que les autres représentations (voir section 5). Ainsi, il est possible de stocker plus d'information dans un espace de stockage réduit. Ce système souffre d'une perte de précision au niveau des données stockées. En effet, la série régénérée à l'aide de son modèle n'est pas nécessairement rigoureusement égale à la série de base. Cela dit, les autres représentations, qu'elles soient temporelles ou spectrales posent aussi le même genre de problèmes et nécessitent une prise en compte des erreurs commises. Un algorithme de segmentation par exemple (sliding window Keogh et al. (2001)) est capable d'approximer une portion d'une série chronologique par une droite. La portion est initialement de deux points consécutifs, puis augmenter point par point tant que la somme des erreurs est inférieure à une valeur seuil. Dans le cas du stockage d'équations, les écarts entre la série régénérée et la série de départ sont dus, en premier à l'erreur commise lors de l'abstraction de la série (la solution de l'équation n'est pas rigoureusement égale à la série de départ) et à l'erreur commise lors de l'application des algorithmes numériques (les valeurs calculées ne sont pas rigoureusement égales).

Un second problème notable est le coût de calcul pour régénérer une série chronologique. Il est, en effet, bien plus important que pour les autres méthodes Albert (2015). D'ailleurs, la complexité des calculs, de même que leur précision, sera dépendante de l'algorithme numérique utilisé.

En revanche, les entrepôts de données permettent de ne rien supprimer, afin de conserver un historique des données Inmon (2002). Ainsi, le système de stockage pourrait donc aussi être un support d'apprentissage incrémental, ou un support d'aide à la prédiction, et permettre d'indiquer une plage temporelle à laquelle le modèle était considéré valide.

5 Prototype expérimental

5.1 Structure de stockage

Des travaux préliminaires ont permis de définir une structure de stockage pour une équation différentielle. La structure d'arbre binaire a été choisie, car cette structure permet de représenter une équation différentielle et les outils informatiques de stockage et de manipulation des arbres binaires sont nombreux. La figure 4 est un exemple de représentation d'une équation par un arbre binaire, il s'agit de l'arbre représentant l'équation 5. Les feuilles sont les différents termes

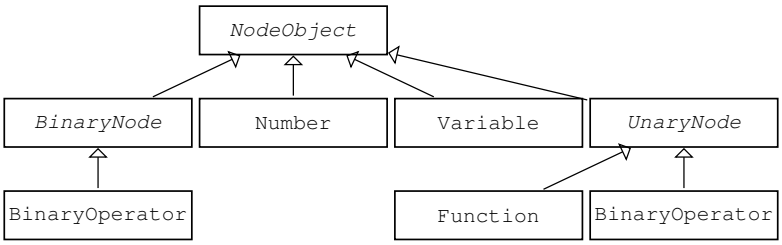

FIG. 5: Diagramme UML simplifié décrivant les nœuds et feuilles d'un arbre

de l'équation (les constantes, les fonctions...), tandis que les nœuds sont les opérateurs. La racine contient toujours l'opérateur `"="`.

La figure 5 contient un diagramme UML simplifié des objets représentant les nœuds et feuilles contenus dans l'arbre.

— La classe `NodeObject` est abstraite et est la représentation générale d'un nœud.

— Les classes `Number` et "Variable" seront des feuilles qui représentent soit un nombre réel invariable dans l'équation (par exemple le nombre 2 dans $2y' = y$), soit une constante (comme T et K dans 5).

— La classe `Function`, représente les fonctions (comme y et u dans 5 ou d'autres fonctions classiques, tel que le cosinus ou l'exponentielle).

— Les classes `BinaryNode` et `UnaryNode` représentent un nœud qui peut avoir deux fils pour la première et un seul fils pour la seconde.

— Les classes `BinaryOperator` et `UnaryOperator` représentent les opérateurs qui peuvent s'appliquer sur deux termes (binaire) ou un seul (unaire).

Une série de tests a été réalisée afin de comparer le stockage des séries chronologiques sous forme d'équations et d'autres représentations classiques (voir sections 5.3, 5.4 et 5.5). Les représentations utilisées pour les tests sont :

— Deux algorithmes de compression, le premier nommé `All-Extrema`, consistant à récupérer les extrema de la série et le second, nommé `Important-Extrema`, étant la version améliorée du premier (voir section 3.1.1) ;

— Deux algorithmes de segmentation, nommés `Bottom-Up` et `Slide-Window`. Ce sont deux algorithmes de segmentation, dont le premier consiste à approximer la série avec le plus de précision possible, puis les segments sont fusionnés, entraînants une perte de précision. Le processus de fusion est contrôlé par un critère d'arrêt. Le second algorithme consiste à utiliser une fenêtre, dans laquelle les points de la série sont approximés par une droite, puis l'on calcul la somme des erreurs, qui doit être inférieur à un certain seuil. Ladite fenêtre est ensuite agrandie jusqu'à atteindre la taille maximale permise par l'erreur seuil ;

— Une représentation directe, dite "brute", la série est stockée tel quel, sans aucune modification.

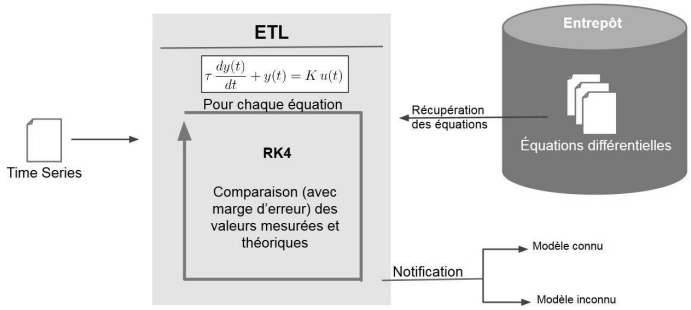

FIG. 6: Prototype d'*ETL "orienté modèles"* Albert (2015)

5.2 Architecture logicielle

Une des utilisations de l'*entrepôt de modèles* requise par les chercheurs en automatique est la comparaison d'une série chronologique avec les modèles préexistants dans la base, afin de repérer des comportements similaires et d'éviter de chercher un modèle qui serait déjà connu. L'activité de comparaison devrait également permettre de vérifier si un modèle existant permet de générer la série en modifiant seulement le système de données associé (voir section 4). Pour réaliser la comparaison, il faut également une activité de génération de données à partir du modèle. Toutes ces considérations nécessitent l'ajout de nouvelles opérations dans l'entrepôt de données classique.

Une opération de génération, qui doit pouvoir récupérer une équation dans la base de données et y appliquer un algorithme d'approximation numérique ; Une opération de comparaison de deux séries chronologiques ou d'une série chronologique avec un ensemble de séries chronologiques.

Ainsi, la figure 6 montre un prototype de processus ETL adapté à un *entrepôt de modèles*. Nous pourrons parler de processus *ETL "orienté modèles"*. Le processus prend une série chronologique en entrée. Dans notre cas, il s'agit de la série brute, mais dans un cas général, il s'agira d'une des représentations temporelles ou spectrales possibles pour une série chronologique. Un processus *ETL "orienté modèles"* devra donc être capable de gérer l'hétérogénéité des représentations en plus des problèmes d'hétérogénéité déjà connus des entrepôts de données. Ensuite, le processus ETL récupère l'ensemble des équations contenues dans l'entrepôt. Ensuite, une première série est générée à l'aide de la méthode de Runge-Kutta d'ordre 4. La série générée est comparée à la série d'entrée (comparaison valeur par valeur, avec une marge d'erreur), si elles sont égales, une notification est envoyée indiquant qu'un modèle correspondant à la série d'entrée a été trouvé, sinon une autre série est générée avec une autre équation. Si aucune égalité n'est détectée, le processus le notifie également.

La figure 7 présente la définition du processus faîtes à l'aide de Talend Open Studio. Un `prejob` est défini afin de récupérer toutes les équations de l'entrepôt. Ensuite, les éléments `timeseries` et `creation_ts` se récupèrent les données des séries, afin d'en créer un objet Java les représentant. Puis les éléments `foreach_DifferentialEquation` et `foreach_System` vont itérer sur les équations et pour une même équation sur chacun des systèmes de données associés. Enfin, les éléments `ModelFound` et `NotFound` sont les pro-

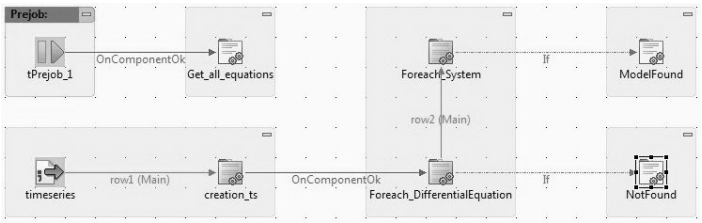

FIG. 7: Définition à l'aide de Talend du processus *ETL "orienté modèles"* Albert (2015)

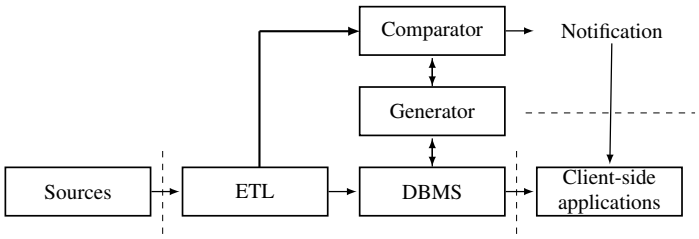

FIG. 8: Schéma d'un *entrepôt de modèles*

cessus de notification.

La notion d'*ETL "orienté modèles"* induit des modifications de la chaîne présentée figure 3. En effet, les opérations ETL classiques telles que le filtrage ou le nettoyage ne sont pas adaptées aux traitements d'une série chronologique et il faut des algorithmes spécifiques de comparaison, par exemple pour la détection de similitude. Par ailleurs, ces algorithmes auront besoin d'une étape préliminaire consistant en la régénération des valeurs des séries à partir des modèles, impliquant l'utilisation d'algorithmes numériques.

Nous proposons donc la chaîne de la figure 8, sur laquelle la chaîne originelle a été légèrement modifiée pour prendre en compte de nouveaux processus. La chaîne ainsi créée n'est plus complètement linéaire avec communication unilatérale entre les différentes couches, mais les communications bilatérales ont été limitées aux processus supplémentaires, qui ont besoin de récupérer des données existantes dans la base pour le générateur et d'effectuer des requêtes de génération pour le comparateur. Il faut également noter que la chaîne ainsi créée permet une communication du processus ETL vers le SGBD sans obligation d'appeler le processus de comparaison. Ainsi, un *entrepôt de modèles* peut-être utilisé comme un entrepôt de données.

5.3 Espace de stockage

Pour les tests, une série de 1 000 éléments fournie par l'équipe de chercheurs en automatique, supposée suivre l'équation 5 a été utilisée. Les séries sont stockées dans des fichiers textes au format csv avec des espaces comme séparateurs, tandis que l'équation est stockée dans un fichier au format XML. Le format XML étant particulièrement adapté au stockage d'arbres binaires.

Le tableau 1 indique le nombre d'éléments contenu dans chaque série selon l'algorithme utilisé. Il n'y a pas de colonne correspondant au stockage par modèle, sachant que dans ce cas

Unité	Crude	All-Extrema	Important-Extrema	Bottom-Up	Sliding-Window
néant	1000	73	36	74	28

TAB. 1: Nombres d'éléments après application des algorithmes

Unité	Crude	All-Extrema	Important-Extrema	Bottom-Up	Sliding-Window	Modèle
octet	88208	6398	3181	6477	2540	2633
%	0	92.75	96.39	92.66	97.12	97.02

TAB. 2: Taille en mémoire

	All-Extrema	Bottom-Up	Slide-Window
u (taille)	72	98	37
u (o)	6318	9182	3260
total (o)	8951	11815	5893
erreur max (%)	0.0	0.0	32.00
erreur moy (%)	0.0	0.0	16.00
compression (%)	89.85	86.61	93.32

TAB. 3: Résumé des performances obtenues pour le stockage de modèles

particulier, aucun élément de la série n'est stocké. En revanche, les éléments sont recalculés à partir de l'équation. Cependant, le tableau 2 indique en octet la taille en mémoire nécessaire au stockage de la série, selon sa représentation, avec le taux de compression associé. La méthode de stockage de modèle permet un taux de compression de $97,02\%$, ce qui est comparable aux taux atteint par les algorithmes *Important-Extrema* et *Sliding-Window*.

Cependant, la valeur indiquée ne prend pas en compte le stockage de l'entrée u. Cette entrée est, en générale, un paramètre connu de l'expérience (ici, il s'agit d'une somme de plusieurs échelons). Le choix le plus simple, qui a été fait dans notre étude, est de stocker cette entrée sous forme de série, bien que d'autres représentations plus optimales puissent être utilisées. Ainsi, le stockage d'une équation différentielle prend plus d'espace que la série elle-même, puisqu'il faut aussi stocker la série d'entrée (u). Cependant, le stockage de modèles présente tout de même les avantages suivants : plusieurs équations peuvent partager une même entrée ; à l'instar de l'exemple utilisé, l'entrée peut être plus simple que la série elle-même, ainsi les algorithmes de compression ou segmentation pourront atteindre de meilleures taux de compression, pour un impact minimal sur la perte de précision des valeurs (voir tableau 5). D'ailleurs, le tableau 3 résume les performances obtenu pour le stockage par modèle, selon l'algorithme utilisé pour u. L'algorithme *All-Extrema* permet d'obtenir un taux de compression de $89,85\%$, sans pertes de précision sur u.

Unité	Brute	All-Extrema	Important-Extrema	Bottom-Up	Slide-Window	Modèle
ms	285	172	168	174	164	3415

TAB. 4: Temps d'exécution pour la récupération de la série Albert (2015)

Brute	All-Extrema	Important-Extrema	Bottom-Up	Slide-Window	Modèles
0	0.035 - 0.577	0.334 - 1.957	0.248 - 0.995	0.458 - 1.337	0.001 - 0.002

TAB. 5: Écarts minimums et maximums Albert (2015)

5.4 Temps d'exécution

Le tableau 4 contient les résultats d'une analyse des performances en temps de calcul nécessaire à la récupération de la série originale ou une version approchée, puisque les algorithmes de compression, segmentation de même que l'abstraction par le modèle induisent des pertes par rapport à l'original. Nous remarquons que les deux algorithmes de compression et de segmentation permettent d'améliorer les performances. En effet, les temps de calculs sont tous plus bas (quasiment divisés par deux pour *Sliding-Window*) que le temps nécessaires à la récupération de la série brute. En revanche, pour le stockage par modèle, le temps de récupération est très largement supérieur. Ceci s'explique par la nécessité d'utiliser des algorithmes de calcul numérique.

Un test a été réalisé avec la base de modèles précédemment décrite, contenant un seul modèle, permettant de calculer le temps d'exécution du processus "orienté modèles" défini ci-dessus. Dans le cas où la série d'entrées correspond au modèle, le temps de calcul est de $3\,485$ ms, dans le cas contraire, le temps a diminué à $1\,639$ ms. En effet, la comparaison étant faite valeur par valeur, lorsqu'une nouvelle valeur est générée, elle est immédiatement comparée à sa valeur correspondante de la série d'entrée. Par conséquent, si la différence entre les deux valeurs est supérieure à une valeur seuil, le processus s'arrête immédiatement. Ainsi, pour détecter l'égalité entre deux séries, il faut en générer tous les éléments, mais en cas d'inégalité, seules les premières valeurs seront générées. A noter que les $3\,485$ ms dans le cas où la série correspond au modèle sont légèrement supérieurs aux $3\,415$ ms nécessaires à la seule régénération de la série et le gain de temps est assez significatif lorsque la génération est avortée. Le plus coûteux en calcul est donc bien l'application des algorithmes de calcul numérique.

5.5 Erreurs

Les modifications effectuées sur une série pour son stockage impliquent une perte de précision sur les valeurs de la série. Ainsi, si l'on compare la série originale avec sa version régénérée à partir des données stockées, celles-ci ne contiennent plus exactement les mêmes valeurs. Le tableau 5 contient les écarts minimums et maximums entre les valeurs de la série originale et des séries recalculées. Ici, l'utilisation de modèles offre une bien meilleure approximation de la série originale que les autres méthodes. Cependant, ce résultat dépend à la fois de la conformité du modèle avec la série originale et de la précision de l'algorithme de calcul numérique

utilisé. Par exemple avec une méthode telle que Runge-Kutta, augmenter l'ordre de la méthode permet d'avoir une meilleure approximation des valeurs de la fonction solution de l'équation. Ainsi, si la série originale est exactement égale à ladite fonction, l'approximation de la série originale en est meilleure.

5.6 Conclusion

Le stockage de modèle est à priori plus lourd que le stockage de la série brute, cependant, des perspectives d'optimisation existent. Aussi, l'utilisation des modèles induit un fort coût de calcul, qu'il sera donc nécessaire d'étudier ultérieurement. Il semble que la perte de précision sur la série originale est bien moindre, à condition que les différentes sources d'approximations (le modèle, le calcul numérique et la série d'entrée) soient maîtrisées.

6 Conclusion

Nous avons proposé une nouvelle vision d'entreposage de données qui consiste à substituer les données par leurs modèles, motivée par l'utilisation massive des séries chronologiques dans plusieurs domaines scientifiques. La notion d'*entrepôt de modèles*, ainsi qu'une démarche de développement de ce dernier a été présentée, avec un exemple d'utilisation par des automaticiens. Une preuve de concept a été apportée et a également permis d'identifier un problème de performances en temps de calcul.

En termes de perspectives, nous nous focalisons sur la formalisation de la notion de processus *ETL "orienté modèles"*. Nous proposons d'ajouter de nouvelles fonctionnalités au processus ETL classique (des outils d'analyse des séries chronologiques, ainsi que des algorithmes de calcul numériques). Afin de proposer une définition formelle du processus *ETL "orienté modèles"*, nous nous sommes d'abord penchés sur la notion de modèle formel d'un processus ETL, abordées dans les articles suivants Skoutas et Simitsis (2007), Vassiliadis et al. (2009), Muñoz et al. (2010) et Vassiliadis et al. (2002).

Le processus *ETL "orienté modèles"* devra également être capable de gérer l'homogénéité des représentations des séries chronologiques. Aussi, la détection des séries dont le comportement peut-être représenté par une même équation différentielle, pourra être réalisée à l'aide d'une opération de détection de séries similaires.

Références

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

Albert, F. (2015). Entrepôt de modèles, application aux données issues de l'automatique. Master's thesis, Université de Poitiers.

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

Elliott, R. (Ed.) (2013). *The Data Warehouse Toolkit* (Third ed.)., pp. 1–35. Wiley Publishing, Inc.

Esling, P. et C. Agón (2012). Time-series data mining. *ACM Comput. Surv. 45*(1), 12 :1–12 :34.

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

Fu, T.-C. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence 24*(1), 164–181.

Hetland, M. L. (2004). A survey of recent methods for efficient retrieval of similar time sequencies. In M. Last, A. Kandel, et H. Bunke (Eds.), *Data Mining In Time Series Databases*, Volume 57, pp. 23–42.

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

Keogh, E., S. Chu, D. Hart, et M. Pazzani (2001). An online algorithm for segmenting time series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp. 289–296.

Muñoz, L., J. Mazón, et J. Trujillo (2010). A family of experiments to validate measures for UML activity diagrams of ETL processes in data warehouses. *Information & Software Technology 52*(11), 1188–1203.

Press, W., S. Teukolsky, W. Vetterling, et B. Flannery (2002). *Numerical Recipes in C : The Art of Scientific Computing*, pp. 707–752. Cambridge University Press.

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

Skoutas, D. et A. Simitsis (2007). Ontology-based conceptual design of ETL processes for both structured and semi-structured data. *Int. J. Semantic Web Inf. Syst. 3*(4), 1–24.

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

Vassiliadis, P., A. Simitsis, et E. Baikousi (2009). A taxonomy of ETL activities. In *Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP*, DOLAP '09, New York, NY, USA, pp. 25–32. ACM.

Vassiliadis, P., A. Simitsis, et S. Skiadopoulos (2002). Conceptual modeling for etl processes. In *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP*, DOLAP '02, New York, NY, USA, pp. 14–21. ACM.

Zeira, G., O. Maimon, M. Last, et L. Rokach (2004). Change detection in classification models induced from time series data. In *Data Mining in Time Series*, Volume 57, pp. 101–125.

Summary

In the scientific fields, more and more data are to be analysed. Those data may take the form of Time Series, which are a temporal data class, containing a record of chronological values. Those values are considered as a whole instead of a list of individual and independent data. Moreover, Time Series generally contain a large number of values and can be stored in a database in quite large quantity. In this article, we propose a storage tool for Time Series, by using their abstract model (differential equation), leading to the new notion of *Models Warehouse* aiming at giving a new representation of Time Series and an alternative storage solution. Unfortunately, this method triggers a significant cost in terms of computation time. Our proposal is implemented and validated using real dataset issued from automatic.