Vers une modélisation agile des entrepôts de données basée sur la technique Anchor Modeling

Thinhinane Hamitouche*, Omar Boussaid**
Fadila Bentayeb**

* LCSI, École nationale supérieure d'informatique, Alger
ft_hamitouche@esi.dz
**ERIC, Université de Lyon, Lyon2
omar.boussaid@univ-lyon2.fr, Fadila.Bentayeb@univ-lyon2.fr

Résumé. Sous l'influence des données massives (Big Data) le système d'information doit s'adapter et évoluer. Cette évolution passe par une transformation des systèmes décisionnels pour la prise en compte de la multiplication des sources de données et l'émergence de nouveaux besoins d'analyse. Même si la question d'évolution de modèle dans les entrepôts n'est pas nouvelle, concevoir des entrepôts évolutifs reste un défi à soulever. Dans cet article nous proposons une approche de modélisation agile des entrepôts basée sur la technique Anchor Modeling qui permet de faire évoluer le modèle d'un entrepôt. Nous définissons des règles de passage d'un modèle multidimensionnel en étoile vers un modèle d'ancrage, puis d'un modèle d'ancrage vers un modèle physique relationnel. Nous validons notre approche par un prototype logiciel et comparons notre modèle multidimensionnel agile avec les modèles d'entrepôts en étoile traditionnels.

1 Introduction

La force des entrepôts de données réside dans leur capacité à intégrer et à agréger un ensemble de sources de données hétérogènes dans un système commun, offrant aux décideurs une vue unifiée des informations générées. Bien que les rafraîchissements des données dans un entrepôt soit une opération bien maîtrisée, sa portée reste limitée à une mise à jour des données sans impact sur la structure initiale de l'entrepôt, ni sur une prise en compte de nouvelles données pouvant faire évoluer les besoins d'analyse initiaux. Un entrepôt de données stocke des données structurées selon un modèle multidimensionnel, généralement représenté par un schéma en étoile. Or, l'environnement d'un entrepôt de données est en perpétuel changement, d'autre part, interfacer un entrepôt de données avec un lac de données, représentant un réel potentiel de données, permettrait à la BI, ou à tout système d'information décisionnel en général, d'exploiter davantage de données pour élargir et enrichir les besoins d'analyse. Transformer les entrepôts de données en structures agiles, pouvant accepter toute évolution, est

un véritable défi. La notion de modèle multidimensionnel agile reste à définir. Elle est certes différente de la notion d'agilité dans un projet. Bien qu'elle se base sur les trois principaux piliers des approches agiles à savoir : l'adaptation, la transparence et l'inspection.

Dans cet article, nous présentons notre approche d'un modèle multidimensionnel agile AM-DW. Pour ce faire, nous recourrons à une technique de modélisation agile qu'est Anchor Modeling (AM). En effet, cette dernière est bien appropriée pour modéliser les entrepôts de données. Le modèle agile est obtenu grâce à des règles de passage d'un modèle multidimensionnel en étoile vers un modèle d'ancrage, puis d'un modèle d'ancrage vers un modèle physique relationnel. Tous les composants multidimensionnels du modèle en étoile sont traduits dans le formalisme AM. Ainsi, AM offre une agilité qui permet d'effectuer des changements aussi bien dans la structure que dans les données d'un entrepôt. Nous validons notre approche par un prototype logiciel et comparons notre modèle agile avec les modèles d'entrepôts en étoile traditionnels. Pour mener à bien nos expérimentations, nous utilisons un extrait de la base de données Northwind de Microsoft portant sur des données d'importation de produits alimentaires. Les résultats que nous avons obtenus montrent que les schémas d'ancrage répondent correctement aux besoins d'analyse et supportent l'évolution de ceux-ci. Toutefois, les performances dépendent de la configuration du schéma choisie.

Le reste de l'article est organisé de la façon suivante. La section 2 présente le formalisme Anchor Modeling. La section 3 présente un état de l'art sur la modélisation multidimensionnelle et la modélisation par ancrage. Dans la section 4, nous décrivons l'approche AM-DW que nous proposons. La section 5 est consacrée aux expérimentations que nous avons menées pour évaluer notre modèle. La section 6 conclut l'article et ouvre de nouvelles perspectives de recherche.

2 Présentation de la technique Anchor Modeling

Anchor Modeling (AM) ou modèle d'ancrage est une technique de modélisation graphique qui permet une évolution non destructive. Tous les changements dans un modèle sont effectués sous forme d'extensions, ce qui rend les différentes versions d'un modèle disponibles en permanence en tant que sous-ensembles du dernier modèle (Rönnbäck et al., 2010). AM est basé sur la 6FN (forme normale), cette dernière a pour objectif d'éviter la duplication des données qui varient indépendemment. L'idée est de séparer tous les attributs de sorte à avoir des tables non décomposables.

Selon les auteurs d'Anchor Modeling (Rönnbäck et al., 2010), les notions de base d'AM sont : *Anchors*, *Knots*, *Attributes* et *Ties*. Nous les explicitons dans ce qui suit, et nous les illustrons dans la figure 1.

Anchor représente une entité, il détient son identifiant. Techniquement parlant, l'identifiant est une clé artificielle (*Surrogate Key*) au lieu d'une clé naturelle (*Business Key*). Par exemple, l'*Anchor* PE_Person avec une instance identifiée par <#42>.

Knot est utilisé pour représenter un ensemble fixe d'entités invariantes dans le temps. Alors que les *Anchors* représentent des entités arbitraires, les *Knots* servent à gérer des propriétés (descripteurs ou rubriques) communes partagées entre plusieurs instances de certains *Anchors*. Par exemple, le *Knot* GEN_Gender contient deux va-

leurs 'Male' et 'Female' ({<#1, 'Male'>,<#2, 'Female'>}), cette propriété est partagée par plusieurs instances de PE_Person. Ainsi, l'utilisation d'un *Knot* minimise la redondance.

Attributes représentent les propriétés des *Anchors*. Nous distinguons quatre types d'attributs : *Static*, *Historized*, *Knotted Static* et *Knotted Historized*. *Static Attribute* sert à représenter les propriétés des entités (*Anchors*) dont les valeurs ne changent pas dans le temps comme la date de naissance. *Historized Attribute* est utilisé pour des attributs qui évoluent temporellement comme le poids. *Knotted Static Attribute* modélise la relation entre *Anchor* et *Knot*, i e., il lie l'entité à une propriété qui prend une valeur fixe parmi l'ensemble *Knot*. *Knotted Historized Attribute* est utilisé lorsque la relation avec *Knot* n'est pas stable et est amenée à changer . Un exemple de *Historized Attribute* est PE_WE_Person_Weight avec l'instance <#42, 65, 2020-01-01> liée à l'instance <#42> de l'*Anchor* PE_Person. Un autre exemple de *Static knotted Attribute* est PE_GEN_Person_Gender avec les instances {<#42,#1>,<#43,#2>}, les instances <#42> et <#43> de l'*Anchor* PE_Person sont liées aux instances <#1,'Male'> et <#2,'Female'> du *Knot* GEN_Gender respectivement.

Tie modélise la relation entre les entités (*Anchors*). Il détient les identifiants de ces dernières. Comme pour les attributs, nous distinguons quatre types : *Static*, *Historized*, *Knotted Static* et *Knotted Historized*. Un exemple est *Tie* PE_Child_PE_Parent qui modélise une relation récursive entre deux instances de l'entité PE_Person. Un autre exemple de *Historized Tie* est PE_Person_CI_City qui modélise une relation, entre l'entité PE_Person et CI_City, qui est amenée à changer dans le temps.

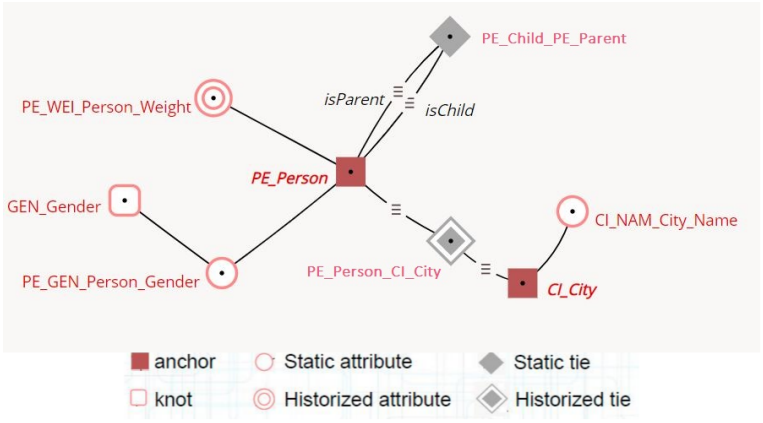

FIG. 1 – *Représentation graphique d'un modèle d'ancrage.*

3 Travaux connexes

Plusieurs travaux de recherche ont été conduits sur la conception des modèles d'entrepôts de données, nous nous intéressons principalement à la modélisation multidimensionnelle (MD) et la modélisation d'ancrage (AM) utilisées dans notre travail.

Toutefois, il existe très peu de travaux récents traitant de l'agilité des modèles d'entrepôts de données basés sur Anchor Modeling. En évaluant les différents travaux étudiés, nous avons remarqué que ces derniers présentent des pistes de recherche différentes. Dans (Malinowski et Zimányi, 2008), les auteurs introduisent un modèle conceptuel *MultiDim* proche d'E/R. Celui-ci couvre tous les concepts MD avec une solution pour gérer l'évolution, c'est le modèle MD conceptuel le plus riche que nous avons recensé. L'adaptation aux changements, qui est un des piliers de l'agilité, dans un modèle MD est notre principal focus. Or, la modélisation d'ancrage s'avère comme une technique de modélisation agile effective.

Les auteurs dans (Rönnbäck et al., 2010) et (Simanjuntak et al., 2016) évoquent cette technique AM dans un contexte général de bases de données relationnelles sans montrer comment l'appliquer sur des modèles d'entrepôts de données. Quand à (Němec, 2012) et (Němec et Zapletal, 2014), ils traitent la modélisation d'ancrage uniquement pour le schéma en étoile, dans une perspective de comparaison de performances. Ils se sont uniquement concentrés sur l'analyse des performances de requêtes sur le schéma d'ancrage et le schéma en étoile. Un autre inconvénient avec ces travaux est qu'ils ciblent les principes relationnels typiques du schéma en étoile (i.e. tables de faits et tables de dimensions), ne couvrant ainsi qu'une seule topologie du modèle multidimensionnel.

Dans la continuité de ces travaux, notre proposition s'en démarque par le fait que nous gérons les hiérarchies en plus des patterns de base, tels que les faits et les dimensions. Notre objectif est d'adapter les principes AM pour apporter une agilité à une large variété de modèles multidimensionnels.

4 L'approche AM-DW

L'approche AM-DW (Agile Modeling for Data Warehouses) que nous proposons utilise le formalisme Anchor Modeling pour créer des modèles multidimensionnels évolutifs. En effet, en se basant sur la modélisation par ancrage, il est possible de faire évoluer le schéma de l'entrepôt. Tous les concepts du modèle multidimensionnel peuvent être décrits avec le formalisme AM, et grâce à sa gestion de la temporalité, il permet de procéder à la mise à jour du schéma du modèle. C'est ainsi que la propriété d'adaptation est satisfaite. Par ailleurs, le formalisme AM est graphique et visuel. Cette propriété facilite la communication des changements au niveau du modèle et la vérification de leur validité. Cela rajoute au modèle multidimensionnel deux autres caractéristiques qui sont la transparence et l'inspection en plus de l'adaptation. Le modèle multidimensionnel évolutif que nous proposons est qualifié d'agile car il respecte les trois caractéristiques de l'agilité. Le processus entier de notre approche est décrit à l'aide de règles de transformations. Pour des fins d'automatisation, nous avons développé un prototype qui restitue automatiquement le modèle physique d'un entrepôt sous forme d'un modèle d'ancrage. Ce dernier représente alors le modèle logique de l'entrepôt qui sera la version où les modifications pourront être introduites.

Dans notre approche AM-DW, tous les niveaux d'une dimension, qu'elle soit à un seul niveau (*niveau 0*) ou à plusieurs (*niveau n*), sont décomposés en objets *Anchor* A^{DIM}, et un ensemble de m objets *Attribute* $Attr^{DIM}$ reliés à chaque niveau, où m={1,..,r-1} avec r le nombre d'attributs du niveau en question. Étant donné que la

technique AM utilise une clé primaire artificielle (*Surrogate Primary Key*), on n'a pas besoin de représenter l'identifiant de l'objet A^{DIM} comme attribut. Par conséquent, on obtient r-1 attributs. Ces derniers peuvent être historisés (*Historized Attributes*) s'ils évoluent dans le temps. Ils peuvent également appartenir à une catégorie (*Knotted Attributes*), i.e., prendre leurs valeurs dans un ensemble fixe. Ces choix nécessitent la connaissance du métier (*Business Process*).

Nous pouvons classer les hiérarchies en deux grandes familles : hiérarchies strictes où toutes les relations entre les niveaux de dimensions sont de type « *one-to-one* » ou « *one-to-many* » et des hiérarchies non strictes où il existe au moins une relation de type « *many-to-many* ». Dans la technique AM, les liens entre les entités sont modélisés par un objet *Tie*. Ce dernier est de type « *many-to-many* » qui inclut toutes les relations. De ce fait, quel que soit le type de la hiérarchie, les niveaux peuvent être reliés deux à deux par des objets $Tie^{DIM,DIM}$. La relation $Tie^{DIM,DIM}$ =(K_1^*,K_2^*) a une clé primaire composée des clés étrangères pointant les deux niveaux reliés.

Nous proposons trois configurations (Config.1, Config.2 et Config.3) pour modéliser les faits, les dimensions et leurs éventuels niveaux d'hiérarchies ainsi que les relations entre eux. Les deux premières sont inspirées des travaux de (Nëmec et Zapletal, 2014), que nous avons étendues en prenant en compte les hiérarchies dans une dimension d'une part, et auxquelles nous avons rajouté une formalisation, d'autre part. Quand à la troisième, nous la proposons dans le cadre de notre travail comme une amélioration des précédentes (Hamitouche, 2020). La force de notre approche consiste à montrer les alternatives possibles pour construire un modèle d'ancrage correspondant au modèle multidimensionnel initial. L'idée est de maximiser l'agilité et de trouver ensuite un compromis pour maintenir les performances. Nous détaillons dans ce qui suit les trois configurations proposées.

1. La première configuration (Config.1) consiste à modéliser les faits par des objets *Anchor* A^{FACT} et un ensemble de m objets *Attribute* $Attr^{FACT}$ reliés à chaque fait, où m={1,..,r} avec r l'ensemble des mesures du fait en question (figure 3). Étant donné que la technique AM utilise une clé primaire artificielle (*Surrogate Primary Key*), on n'a pas besoin de représenter l'identifiant de l'objet A^{FACT} (ensemble des clés pointant les niveaux reliés) comme attribut.
   Le fait est relié à une dimension par un seul objet $Tie^{FACT,DIM}$.
   La relation $Tie^{FACT,DIM}$=(K^*, K_1^*,..,K_n^*) a une clé composée de (n+1) clés étrangères pointant la clé primaire du fait et celles de n niveaux (Němec, 2012).
2. La deuxième introduit une nouvelle variante du concept *Tie* auquel sont ajoutées les mesures, appelé *TieFact* (Nëmec et Zapletal, 2014). En effet, un seul objet $TieFact^{FACT,DIM}$ est créé pour relier le fait aux niveaux de dimensions. La relation $TieFact^{FACT,DIM}$=(K_1^*,..,K_n^*,M_1,..,M_m) a une clé primaire composée de n clés étrangères pointant les clés primaires de n niveaux de dimensions. M étant l'ensemble de m mesures. La notation graphiue de l'objet *TieFact* proposé dans ce travail apparaît dans la figure 4.
3. La troisième configuration (Config.3), que nous proposons dans ce travail, est similaire à la première (Config.1) quant à la représentation des faits et des mesures. Néanmoins, un objet de type $Tie^{FACT,DIM}$ est utilisé pour relier le fait à chaque dimension. Par exemple, si un fait est relié à n dimensions, il est

nécessaire de créer n objets $Tie^{FACT,DIM}$. La relation $Tie^{FACT,DIM}$=(K^*,K_1^*) a une clé primaire composée de deux clés étrangères, la première pointe la clé du fait et la deuxième celle du niveau de dimension (figure 5).

Ces différentes configurations sont illustrées dans la section 4.1.1. Celles-ci permettent de couvrir les différents patterns MD et de gérer leurs évolutions. Ainsi , nous obtenons des modèles agiles qui supportent des changements au niveau des sources de données et des besoins d'analyse. Cependant, nous considérons que chaque configuration a son degré de flexibilité. Nous avons défini une métrique, appelée score d'agilité, pour classer les configurations de la moins à la plus agile comme suit : Config.2, Config.1 et enfin Config.3 (Hamitouche, 2020).

La traduction du modèle multidimensionnel vers un modèle d'ancrage se fait automatiquement en appliquant un ensemble de règles de passage listées dans ce qui suit.

4.1 Règles de passage d'un modèle multidimensionnel vers un modèle d'ancrage

- **Règle 1 :** tout niveau de dimension est traduit en *Anchor* et ses attributs en *Attributes* et/ou *Historized Attributes* et/ou *Knotted Attributes* et/ou *Knotted Historized Attributes* sauf l'identifiant.
- **Règle 2 :** la relation entre deux niveaux d'une dimension est traduite en *Tie* ou *Historized Tie* .
- **Règle 3 :** tout fait est traduit en *Anchor* et ses mesures en *Attributes* ou *Historized Attributes* (sauf dans Règle 4b).
- **Règle 4 :** la relation entre le fait et les dimensions de niveau 0 peut être traduite de trois manières différentes :
  - **Règle 4a :** un seul objet *Tie* reliant le fait à toutes les dimensions de niveaux 0 (Config.1).
  - **Règle 4b :** un seul objet *TieFact* reliant le fait à toutes les dimensions de niveaux 0, cet objet contient également les mesures (Config.2).
  - **Règle 4c :** un objet *Tie* reliant le fait à chaque dimension de niveau 0 (Config.3).

4.1.1 Exemple

Pour illustrer notre approche, nous avons utilisé un extrait de l'entrepôt de données *Northwind*[1] avec le fait Sales et les dimensions Customer, Product et sa hiérarchie Category. Par soucis de clarté, nous avons réduit le nombre d'attributs et des mesures des dimensions et du fait respectivement (figure 2).

Les figures 3, 4 et 5 illustrent l'application de l'approche AM-DW pour obtenir la 1ère configuration (Config.1), la 2ème configuration (Config.2) et la 3ème configuration (Config.3) respectivement.

1. http ://northwinddatabase.codeplex.com/

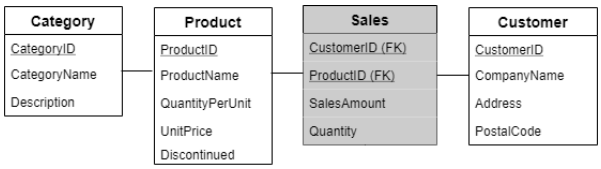

FIG. 2 – *Schéma d'un extrait de l'entrepôt de données Northwind.*

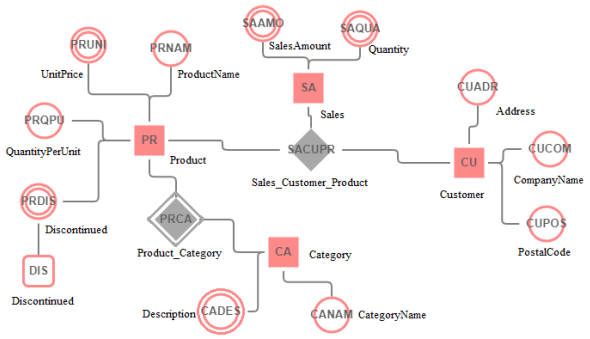

FIG. 3 – *Modèle d'ancrage : Config.1.*

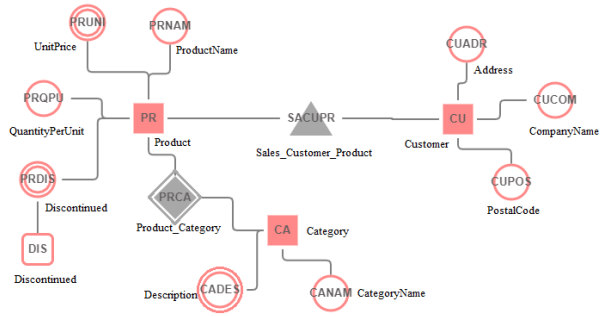

FIG. 4 – *Modèle d'ancrage : Config.2.*

4.2 Règles de passage d'un modèle d'ancrage vers un modèle relationnel physique

Afin de pouvoir comparer les différentes configurations proposées lors des expérimentations, nous avons traduit le modèle logique AM en un modèle physique relationnel. Cette phase se déroule selon les règles suivantes.

— **Règle 5 :** tout niveau de dimension est traduit en une table T_L qui contient uniquement une clé primaire artificielle.

— **Règle 6 :** tout attribut de niveau est traduit en une table T_A qui contient une clé primaire, référençant la table T_L, et la valeur de l'attribut.

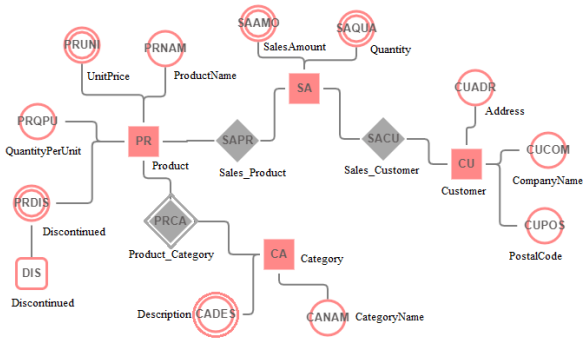

FIG. 5 – *Modèle d'ancrage : Config.3.*

- **Règle 6a :** si l'attribut est attaché à un *Knot*, sa valeur contiendra dans ce cas une clé étrangère référençant la table T_K (Règle 11).
- **Règle 6b :** si l'attribut est historisé, une colonne est ajoutée pour stocker la date à partir de laquelle l'attribut devient valide. La date de validité constitue également la clé primaire.

— **Règle 7 :** la relation entre deux niveaux de dimension L_1 et L_2 est traduite en une table T_{L_L} dont la clé primaire est constituée des deux clés étrangères référençant T_{L1} et T_{L2} .

- **Règle 7a :** si la relation est historisée, une colonne est ajoutée pour stocker la date à partir de laquelle la relation entre un membre de L_1 et de L_2 devient valide. La date de validité constitue également la clé primaire.

— **Règle 8 :** tout fait est traduit en une table T_F qui contient uniquement une clé primaire artificielle (sauf dans Règle 10b).

Règle 9 : toute mesure est traduite en une table T_M qui contient une clé primaire référençant la table T_F ainsi que la valeur de la mesure (sauf dans Règle 10b).

- **Règle 9a :** si la mesure est historisée, une colonne est ajoutée pour stocker la date à partir de laquelle la mesure devient valide. La date de validité constitue également la clé primaire.

— **Règle 10 :** la relation entre le fait et les niveaux de dimension peut être traduite de trois manières différentes :

- **Règle 10a :** si un seul objet *Tie* relie le fait à tous les niveaux L_1,..,L_n, cet objet sera traduit en une table T_{F_L} dont la clé primaire est constituée de clés étrangères référençant les tables T_F et T_{L1},..,T_{Ln}.
- **Règle 10b :** si un objet *TieFact* relie le fait à tous les niveaux L_1,..,L_n, cet objet sera traduit en une table T_{F_L} dont la clé primaire est constituée des clés étrangères référençant les tables T_F et T_{L1},..,T_{Ln}. Autant de colonnes que de mesures seront ajoutées pour contenir les valeurs des mesures.
- **Règle 10c :** si un objet *Tie* relie le fait à chaque niveau L, cet objet sera traduit en une table T_{F_L} dont la clé primaire est constituée des clés étrangères

référençant les tables T_F et T_L.

— **Règle 11 :** tout objet *Knot* est traduit en une table T_K qui contient une clé primaire artificielle et la valeur du *Knot*.

Nous reprenons l'exemple précédent (extrait de l'ED *Northwind*) pour montrer la passage du modèle logique d'ancrage (précisément Config.3) vers un modèle relationnel physique de l'entrepôt de données. Par soucis de clarté, nous n'adoptons pas la notation des colonnes pour les tables relationnelles. Le symbole (*) signifie que l'attribut est une clé primaire de la table, l'équivalent de (PK, *Primary Key*).
Le tableau 1 décrit le modèle relationnel physique de la Config.3 (figure 5).

Table Anchor	
SA_Sales	(SA_ID^*)
CU_Customer	(CU_ID^*)
PR_Product	(PR_ID^*)
CA_Category	(CA_ID^*)
Table Knot	
DIS_Discontinued	(DIS_ID^*, Discontinued)
Table Attribute	
SAAMO_SalesAmount	(SA_ID^*, SalesAmount, $SAAMO\_ValidFrom^*$)
SAQUA_Quatity	(SA_ID^*, Quatity, $SAQUA\_ValidFrom^*$)
CUADR_Address	(CU_ID^*, Address)
CUCOM_CompanyName	(CU_ID^*, CompanyName)
CUPOS_PostalCode	(CU_ID^*, PostalCode)
PRNAM_ProductName	(PR_ID^*, ProductName)
PRUNI_UnitPrice	(PR_ID^*, UnitPrice, $PRUNI\_ValidFrom^*$)
PRQPU_QuantityPerUnit	(PR_ID^*, QuantityPerUnit)
PRDIS_Discontinued	(PR_ID^*, Discontinued, $PRDIS\_ValidFrom^*$)
CADES_Description	(CA_ID^*, Description, $CADES\_ValidFrom^*$)
CANAM_CategoryName	(CA_ID^*, CategoryName)
Table Tie	
SACU_Sales_Customer	(SA_ID^*, CU_ID^*)
SAPR_Sales_Product	(SA_ID^*, PR_ID^*)
PRCA_Product_Category	(PR_ID^*, CA_ID^*, $PRCA\_ValidFrom^*$)

TAB. 1 – *Modèle relationnel physique de l'extrait de l'ED Northwind : Config.3.*

4.3 Évolution du modèle

Dans ce qui suit, nous montrons comment notre approche supporte l'évolution du modèle de manière plus efficace que la modélisation multidimensionnelle classique, et ce en faisant évoluer l'exemple de la figure 2. Supposons qu'un nouveau besoin d'analyse apparaît, celui d'analyser les ventes (SALES) de la société Norhthwind selon l'axe des fournisseurs, de nouvelles données doivent également alimenter cet entrepôts. Ceci engendre la création d'une nouvelle dimension (SUPPLIER) mais aussi la modification du fait SALES qui devrait être rattaché à cette nouvelle dimension (voir figure 6). L'un des problèmes qui peuvent surgir est celui de l'intégrité de l'entrepôt de données. En effet, l'ajout de la clé étrangère, référençant la dimension SUPPLIER, à la clé primaire du fait SALES déclenche un problème d'intégrité de type valeurs NULL puisque les anciennes instances de SALES étaient uniquement rattachées aux instances de CUSTOMER et PRODUCT. Encore faut-il penser des mécanismes à chaque évolution pour maintenir la qualité de l'entrepôt de données.

Notre approche AM-DW tire avantage de la technique AM qui permet une évolution non destructive. Ainsi, tous les changements sont effectués sous forme d'extensions

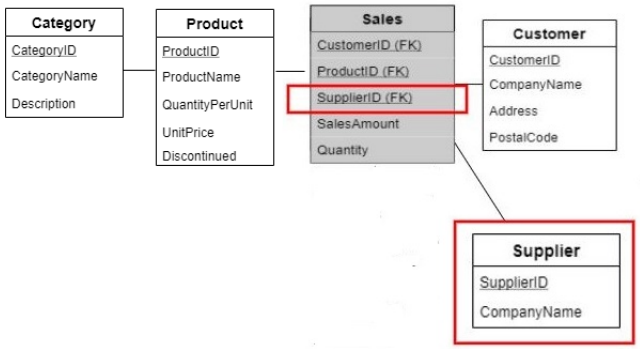

FIG. 6 – *Schéma de l'extrait de l'entrepôt de données Northwind après évolution.*

sans modifier l'existant. Pour montrer l'évolution du schéma de l'entrepôt de données Northwind (figure 2), nous prenons le cas de la troisième configuration (Config.3), en appliquant la **Règle 1** et **Règle 4c**. Il suffit de rajouter de nouveaux éléments AM sans détruire le schéma existant (figure 7).

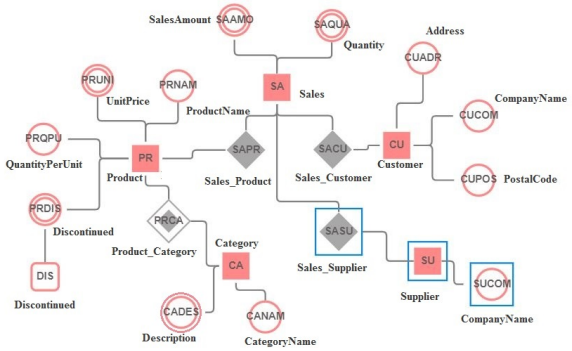

FIG. 7 – *Modèle d'ancrage après évolution : Config.3.*

Au niveau du modèle relationnel physique, ceci se traduit par l'ajout des tables suivantes (voir tableau 2) à celle déjà mentionnées dans le tableau 1.

Table Anchor	
SU_Supplier	(SU_ID^*)
Table Attribute	
SUCOM_CompanyName	(SU_ID^*, CompanyName)
Table Tie	
SASU_Sales_Supplier	(SA_ID^*, SU_ID^*)

TAB. 2 – *Tables rajoutées après évolution : Config.3.*

4.4 Implémentation

L'objet de ce travail étant de proposer une approche formalisée et de l'implémenter afin de démontrer sa faisabilité ; nous avons développé un prototype appelé *Modeller* (c.f. Hamitouche (2020)). La phase d'automatisation du passage d'un modèle multidimensionnel vers un modèle d'ancrage n'est pas encore intégrée. Nous considérons donc que *Modeller* dispose du modèle AM comme entrée. Ainsi, il gère ses versions évolutives et trace les changements. *Modeller* implémente également les règles de passage proposées pour générer le schéma physique de toute configuration du modèle d'ancrage. Nous utilisons *Modeller* pour générer les modèles d'ancrage physiques et mener des expérimentations.

5 Expérimentation et résultats

Pour évaluer l'approche proposée, nous avons mené un ensemble d'expérimentations sur l'ED *Northwind*. Nos objectifs au travers de l'évaluation expérimentale consistent d'une part, à valider sémantiquement le modèle obtenu en appliquant l'approche présentée, et ce en montrant qu'il répond aux besoins de la même manière que le modèle initial (**Objectif 1**). D'autre part, évaluer et comparer les performances du modèle AM selon les trois configurations proposées (**Objectif 2**).

5.1 Protocole d'expérimentation

5.1.1 Spécifications du test

La raison de la sélection de l'entrepôt de données *Northwind* pour l'expérimentation réside dans le fait qu'il soit un exemple typique qui engendre pratiquement tous les concepts multidimensionnels. De ce fait, nous avons pu appliquer toutes les règles de passage de notre solution. En plus, la taille de cet entrepôt et le nombre de structures présentes sont convenables pour construire des schémas d'ancrage et les alimenter avec les données de l'ED de manière assez facile et rapide. En effet, cet ED contient un fait et 12 dimensions avec une taille total de 2 480 Ko.

Dimensions	**Attributs**
Category	3
Product	6
Customer	10
Supplier	9
Shipper	3
Employee	14
Time	4
City	4
Region	3
Country	2
Fait	**Attributs**
Sales	13

Tables	**Config.1**	**Config.2**	**Config.3**
Anchor	11	10	11
Knot	1	1	1
Attribute	48	42	48
Tie	6	5	11
Tie Fact	0	1	0
Vues	11	10	11

TAB. 3 – *Synthèse des caractéristiques de l'ED Northwind et des 3 modèles d'ancrage.*

5.1.2 Environnement d'expérimentation

- **Serveur physique de base de données** : Intel(R) Core (TM) i7-6700HQ CPU @ 2.60GHz, 2601 MH, 16 GB RAM
- **Système de gestion de base de données** : PostgreSQL 12.2
- **Système de test de performance :** Apache JMeter 5.2

5.1.3 Échantillon de requêtes de test SQL/OLAP

Pour effectuer notre évaluation, nous avons sélectionné un échantillon de 9 requêtes de l'ED *Northwind* qui couvrent les principales opérations OLAP (Roll up, Cube, Grouping set et Window Functions). L'idée est de vérifier que les schémas d'ancrage répondent correctement aux besoins d'analyse dans un ED. Bien évidement, la syntaxe diffère pour les schémas d'ancrage. Nous avons reformulé manuellement les 9 requêtes afin d'obtenir l'équivalent pour chaque schéma AM. Notons que nous avons fait appel aux vues matérialisées dans le but d'optimiser ces requêtes.

5.1.4 Métriques d'évaluation

Chaque requête a été exécutée 100 fois sur chacun des 4 schémas (le schéma MD et les trois schémas AM) afin d'éviter une distorsion significative des résultats par des valeurs aberrantes éventuelles. les métriques employées sont les suivantes :

Différence des résultats : Il s'agit de vérifier pour toute requête la similitude des résultats renvoyés lorsqu'elle est exécutée sur le schéma MD et sur les 3 schémas AM (**Objectif 1**). Pour ce faire, nous appliquons la différence des relations selon la formule suivante :

$$R_{md} - R_{am} \cup R_{am} - R_{md} = \varnothing \tag{1}$$

Nous considérons R_{md} la relation qui résulte de l'exécution de la requête Q_{md} sur le schéma MD et R_{am} celle qui résulte de l'exécution de la requête Q_{am} sur l'un des 3 schémas AM. Ainsi, la *Métrique.1* correspond à la relation (1).

Différence de temps d'exécution : Il s'agit de calculer pour chaque requête le temps moyen de 100 exécutions et ce sur le schéma MD et les 3 schémas AM. Ensuite, nous mesurons la différence pour évaluer les performances (**Objectif 2**) selon la formule suivante :

$$Métrique.2 = \frac{1}{100}(\sum_{i=1}^{100} T_{ami} - \sum_{i=1}^{100} T_{mdi}) \tag{2}$$

5.2 Résultats et discussion

Lors des tests, tous les résultats ont montré 0% d'erreurs. Ainsi, les tests peuvent être considérés comme réussis. Les requêtes ont également été validées en terme de similitude des résultats. Les résultats de toutes les requêtes correspondent à leurs équivalents dans les 4 schémas de test (*Métrique.1=vide*). En terme de performance, comme

Requête	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**
Lignes en sorties	1763	1852	1763	1852	2155	2155	2155	2155	2155
Colonnes en sorties	3	3	3	3	4	4	4	4	4
***Métrique.1* : différence des résultats**									
AM Config.1	∅	∅	∅	∅	∅	∅	∅	∅	∅
AM Config.2	∅	∅	∅	∅	∅	∅	∅	∅	∅
AM Config.3	∅	∅	∅	∅	∅	∅	∅	∅	∅
***Métrique.2* : différence de temps d'exécution (en ms)**									
AM Config.1	14	57	64	46	13	9	11	5	6
AM Config.2	0	0	0	0	0	0	0	2	3
AM Config.3	75	65	76	56	18	30	32	7	8

TAB. 4 – *Synthèse des résultats des expérimentations.*

nous pouvons le remarquer sur le tableau 4, les schémas AM présentent des temps d'exécution plus élevés que le schéma initial (MD). Ceci est étroitement lié aux nombreuses jointures du modèle physique même si ces dernières ont été nettement réduites par les vues matérialisées. Nous pouvons classer les schémas AM du moins performant au plus performant comme suit : (AM Config.3, AM Config.1 et enfin AM Config.2). Dans le cas des requêtes Q1...Q7, le 2ème schéma AM présente les mêmes temps d'exécution moyens vu qu'elles interrogent une seule table (table de fait). Les temps d'exécution moyens des requêtes Q8 et Q9 sont également très proches puisqu'elles nécessitent une seule jointure supplémentaire.

Nous ne pouvons pas affirmer quelle configuration AM parmi les trois est la meilleure. En effet, auparavant, nous avons classé les configurations selon l'efficacité du support de l'évolution du modèle par ordre décroissant comme suit : (AM Config.3, AM Config.1 et enfin AM Config.2.). Par conséquent, nous déduisons que la meilleure en terme d'évolution du modèle étant celle que nous avons proposé dans notre approche (AM Config.3). La plus performante en terme d'exécution de requêtes étant AM Config.2. Quant à AM Config.1, elle peut être considérée comme une solution-compromis parmi les trois.

Discussion. En général, nous pouvons dire que les schémas d'ancrage relationnels sont moins performants que les schémas multidimensionnels relationnels en terme de temps d'exécution des requêtes. Le but de l'utilisation de la modélisation par ancrage et de permettre de mieux gérer l'évolution du modèle MD afin de permettre à un entrepôt de nouvelles possibilités d'analyse autres que celles pour lesquelles il a été construit à l'origine. Afin de maintenir le niveau de performance des ED multidimensionnels, il est intéressant d'étudier la perspective de traduire le modèle logique d'ancrage en un modèle physique en NoSQL. Parmi les avantages d'une telle solution c'est de prendre en compte une variété des données plus larges, ainsi que l'intérêt d'interfacer un entrepôt avec un lac de données. Cependant, l'impact de l'inconvénient des performances d'un modèle d'ancrage relationnel doit être mis en balance avec d'autres avantages que présente notre approche de modélisation agile des entrepôts de données.

6 Conclusion

Poussées par la croissance continue des données, les approches des entrepôts de données doivent être adaptées. Généralement, les modèles en étoile sont inadéquats

lorsqu'il s'agit de données massives qui ont besoin de systèmes évolutifs et flexibles. C'est dans ce contexte que nous avons proposé le modèle multidimensionnel agile basé sur le formalisme Anchor Modeling. Ce dernier permet l'évolution de modèle dans les entrepôts pour tenir compte des nouveaux besoins d'analyse des utilisateurs et de nouvelles sources de données. Ce travail ouvre de nombreuses perspectives. La reformulation des requêtes pour les variantes de modèles évolutifs obtenus ainsi que leur alimentation par les données constituent une voie intéressante à explorer et à automatiser au final. Par ailleurs, nous proposons de mener une réflexion sur le passage d'un modèle logique d'ancrage vers un modèle physique NoSQL, chose qui n'a jamais été étudiée dans la littérature à ce jour. Une amélioration fonctionnelle de notre prototype serait envisagée pour générer automatiquement les modèles physiques NoSQL.

Références

Hamitouche, T. (2020). Vers une modélisation agile des entrepôts de données. *Mémoire de projet de fin d'études d'ingénieur en informatique*.

Malinowski, E. et E. Zimányi (2008). A conceptual model for temporal data warehouses and its transformation to the er and the object-relational models. *Data & Knowledge Engineering 64*(1), 101–133.

Němec, R. (2012). The comparison of anchor and star schema from a query performance perspective. *World Academy of Science, Engineering and Technology 71*, 1718–1722.

Němec, R. et F. Zapletal (2014). The design of multidimensional data model using principles of the anchor data modeling : An assessment of experimental approach based on query execution performance. *WSEAS Transactions on Computers 13*, 177–194.

Rönnbäck, L., O. Regardt, M. Bergholtz, P. Johannesson, et P. Wohed (2010). Anchor modeling—agile information modeling in evolving data environments. *Data & Knowledge Engineering 69*(12), 1229–1253.

Simanjuntak, H., M. Tambunan, K. Manullang, et H. Panjaitan (2016). An automatic tool for anchor model data warehouse development. *INTERNETWORKING INDONESIA 8*(2), 33–38.

Summary

Under the influence of Big Data, the information system must adapt and evolve. This development requires a transformation of decision-making systems to take into account the multiplication of data sources and the emergence of new analysis needs. While the issue of model evolution in Data Warehouses (DW) is not new, designing scalable DWs remains a challenge. In this article we propose an agile DW model based on the Anchor Modeling technique that allows its evolution. We define rules for moving from a multidimensional star model to an anchor model, then from an anchor model to a relational physical model. We validate our approach with a software prototype and compare our agile multidimensional model with traditional multidimensional star models.