Fonctions d'oubli dans les entrepôts de données

Aliou Boly*, **, Georges Hébrail*, Marie-Luce Picard**

*Ecole Nationale Supérieure des Télécommunications de Paris
46, Rue Barrault 75634 Paris cedex 13 France
boly@enst.fr, hebrail@enst.fr
**Electricité de France Recherche et Développement
1, Av. du Général de Gaulle 92141 Clamart Cedex
marie-luce.picard@edf.fr

Résumé. Les entrepôts de données stockent des quantités de données de plus en plus massives, en particulier du fait de la constitution d'historiques. Nous proposons ici une solution pour éviter la saturation des entrepôts de données. Nous définissons un langage de spécifications de fonctions d'oubli des données les plus anciennes, permettant de déterminer ce qui doit être présent dans l'entrepôt de données à chaque instant. Ces spécifications de fonctions d'oubli se traduisent par des opérations de résumé par agrégation, et par des opérations de suppression des données anciennes réalisées de façon mécanique à chaque pas de mise à jour. La communication présente tout d'abord une description syntaxique du langage de spécifications des fonctions d'oubli. Les contraintes à vérifier pour assurer la cohérence du langage sont ensuite décrites. Enfin, nous proposons des structures de données adaptées au stockage des données nécessaires à la gestion des fonctions d'oubli.

1 Introduction

L'objectif de cette communication est d'apporter une réponse au problème de saturation des entrepôts de données, en définissant un langage de spécifications de fonctions d'oubli des données. Ces spécifications conduisent à supprimer de façon mécanique les données à 'oublier', tout en conservant un résumé de celles-ci par agrégation. Pour définir ces stratégies d'oubli des données, le langage considère comme critère d'oubli une dimension temporelle : l'ancienneté de la donnée. L'ancienneté d'une donnée, définie au niveau du n-uplet d'une table, peut être soit la date de dernière mise à jour du n-uplet (fournie par le SGBD et appelée timestamp), soit de façon explicite par une colonne de type Date. Les spécifications d'oubli sont définies sur chaque table, prises indépendamment les unes des autres dans ce travail.

Ce travail est à relier aux travaux sur les bases de données temporelles [Dumas *et al.,* 2001] où l'on cherche à dater les informations et à gérer leur historique, et aux travaux sur la gestion des versions dans les bases de données [Cellary *et al,.* 2001], où l'on cherche à conserver et manipuler les évolutions de données. Il peut également être relié aux travaux de [Chaudhuri *et al.,* 1998] et [Chaudhuri *et al.,* 2001], qui cherchent à estimer les résultats de requêtes sur une table volumineuse en ne considérant qu'un échantillon du contenu de la table. Cependant, notre problématique est différente car dans ces travaux, l'ensemble des données reste toujours en ligne, alors que nous proposons ici des stratégies pour archiver définitivement les données détaillées anciennes, en n'en conservant qu'une version agrégée.

La section 2 décrit la grammaire du langage : la syntaxe du langage est traitée dans la section 2.1, la section 2.2 présente la BNF du langage. La section 3 étudie les contraintes à vérifier pour assurer la cohérence du langage. La section 4 propose des structures de données adaptées au stockage des données nécessaires à la gestion des fonctions d'oubli.

Définition de la valeur d'ancienneté d'une donnée : La valeur d'ancienneté d'une donnée d notée $v_{ancien}(d)$ est égale à la différence entre l'instant où se fait la mise à jour de l'entrepôt ($t_{updatestep}$) et le timestamp de la donnée ($t_{timestamp}(d)$).

$$v_{ancien}(d) = t_{updatestep} - t_{timestamp}(d)$$

Prenons comme exemple la table de clients dont les attributs sont : Id_cli (l'identifiant du client), le nom, la ville, le département (Dept), la région, le sexe, l'âge et le revenu du client. La structure de la table est la suivante :

CLIENT	Id_cli	Nom	Ville	Dept	Region	Sexe	Age	Revenu

Une spécification de fonction d'oubli sur cette table peut être la suivante :

```
SUMMARY TABLE Clients {
  TIMESTAMP = SYSTEM ;
  DISCRETISE (Age) = Discretise_Age: (]0, 25[ 'Jeune'; [25, 60[ 'Adulte'; [60, 90[ 'Agé';)
  HIERARCHY (Géographie) : Ville→ Dept→ Region ;
  TO – 30 DAY : DETAIL ;
  TO – 3 MONTH : SUM (Revenu) BY Ville, Sexe, Discretise_Age, DAY ;
  TO – 1 YEAR : SUM (Revenu) BY Dept, Discretise_Age, MONTH ;
  TO – 10 YEAR : SUM (Revenu) BY Région, YEAR ;
  TO – 15 YEAR : SUM (Revenu) BY YEAR ;
}
END SUMMARY ;
```

Cette spécification indique d'abord que la fonction d'oubli s'applique à la table Clients et que l'ancienneté considérée est le timestamp fourni par le système. L'option DISCRETISE permet de transformer la colonne AGE en un type non numérique, en donnant un nom à chaque intervalle de valeurs. L'option HIERARCHY permet de définir une hiérarchie basée sur les attributs Ville, Dept et Region. Enfin, les ordres 'TO' permettent de spécifier le contenu de l'entrepôt après application de la fonction d'oubli. L'interprétation des spécifications de la fonction d'oubli est la suivante :

Pour les données de moins de 30 jours ($v_{ancien}(d) \leq 30$ jours), on garde le détail (c'est-à-dire les n-uplets de la table Clients). Pour les données de valeur d'ancienneté comprise entre 30 jours et 3 mois (30 jours $\leq v_{ancien}(d) \leq 3$ mois), on supprime les n-uplets correspondants dans la table Clients, et on en conserve une version agrégée par ville, sexe, age et jour. Il est important de souligner que lorsque l'on agrège par jour, c'est l'attribut timestamp qui est considéré (on veut dire que l'on obtient ici le jour à partir du timestamp de la donnée et il en est de même quand on agrège par mois ou année). A noter aussi que l'agrégation par Age est rendue possible grâce à la discrétisation de l'attribut Age. Les attributs Ville, Sexe, Discretise_Age, DAY sont appelés *dimensions*. Une dimension est

considérée ici, comme un composant d'une clause BY, qui peut être défini par l'une des trois façons suivantes : (1) un attribut non numérique de la table, (2) un attribut numérique discrétisé de la table, (3) une période associée au timestamp ou à l'attribut de type Date définissant l'ancienneté du n-uplet (par exemple l'année de facture dans le cas d'une colonne Date de facture).

Ensuite, les données de valeur d'ancienneté comprise entre 3 mois et 1 an sont agrégées par département, age et mois. Pour les données de valeur d'ancienneté comprise entre 1 an et 10 ans, l'agrégation se fait par région et année. Le passage d'une agrégation par département à une agrégation par région est possible car la hiérarchie (Ville→ Dept→ Région) a été spécifiée. La dernière spécification signifie que les données de valeur d'ancienneté comprise entre 10 ans et 15 ans sont agrégées à l'année, et que toute donnée de valeur d'ancienneté supérieure à 15 ans est totalement supprimée de l'entrepôt de données.

2 Grammaire du langage

2.1 Description syntaxique du langage

Dans la description des fragments de grammaire du langage, les mots-clefs sont en majuscules, les constructions entourées de '[' et de ']' sont facultatives, celles suivies de '*' peuvent apparaître 0 ou plusieurs fois, celles suivies de '+' doivent apparaître au moins une fois et le signe 'I' signifie autre forme possible. Les éléments dont le nom commence par *id* représentent une séquence alphanumérique quelconque. Le fragment de grammaire suivant définit la structure d'une fonction d'oubli :

```
< fonction_oubli > ::= SUMMARY TABLE idTable {
        TIMESTAMP = SYSTEM| <nom_colonne> ;
        (< discrétisation > ;)*
        (< hiérarchie > ;)*
        (< spécification > ;)+
}
END SUMMARY ;
```

Dans ce qui suit, on explique brièvement chacune des formes introduites dans la structure ci-dessus. Une description complète de chaque fragment figure dans la section 2.2.

- Le *timestamp* : sa valeur peut être donnée soit par le système, soit par une colonne de type Date de la table.
- Une *discrétisation* : permet de discrétiser un attribut de type numérique pour constituer une dimension.
- Une *hiérarchie* : permet de définir les relations père-fils entre les niveaux d'une dimension pour déterminer la manière dont les données dépendantes peuvent être agrégées.
- Une *spécification* : sa structure syntaxique est décrite comme suit :

```
< Spécification > ::= TO - < valeur > : (DETAIL | < Spécif_agrégat >)
```

Quelques points importants à noter dans cette description :

(a) Les spécifications peuvent être ordonnées de façon croissante suivant <valeur> : si $Spécif_1$ et $Spécif_2$ sont 2 spécifications ayant respectivement pour valeurs val_1 et val_2, alors on dit que $Spécif_1$ précède $Spécif_2$ si et seulement si : $val_1 < val_2$ (le '<' ici concerne des durées, par exemple : 30 DAY < 1 MONTH < 1 YEAR).

(b) La clause TO - <Valeur> permet de désigner, pour la *première* spécification, toutes les données qui ont une valeur d'ancienneté inférieure à la valeur se trouvant à droite du TO :

Si val représente la valeur se trouvant à droite du TO d'une spécification Spécif et si $v_{ancien}(d)$ est la valeur d'ancienneté d'une donnée quelconque d ,

Alors $d \in$ Spécif si et seulement si : $v_{ancien}(d) \leq val$

Pour tout autre spécification (différente de la première), la clause TO - <Valeur> désigne toutes les données de valeur d'ancienneté comprise entre la valeur indiquée dans la spécification et celle indiquée dans la spécification précédente :

Si val représente la valeur se trouvant à droite du TO d'une spécification Spécif et val_p la valeur se trouvant à droite du TO de la spécification qui lui précède et si $v_{ancien}(d)$ est la valeur d'ancienneté d'une donnée quelconque d ,

Alors $d \in$ Spécif si et seulement si : $val_p \leq v_{ancien}(d) \leq val$

(c) Enfin, toute donnée de valeur d'ancienneté supérieure à la valeur mentionnée dans la dernière spécification, sera supprimée.

De telles spécifications permettent de définir les mises à jour à exécuter à chaque instant où l'on souhaite appliquer la fonction d'oubli. La fonction d'oubli peut être appliquée de façon périodique, par exemple le 1er de chaque mois.

2.2 BNF du langage

< fonction_oubli > ::= SUMMARY TABLE *id*Table {
TIMESTAMP = SYSTEM| <nom_colonne> ;
(< discrétisation > ;)*
(< hiérarchie > ;)*
(< spécification > ;)+
}
END SUMMARY ;
<nom_colonne> ::= *id*Att
<discrétisation> ::= DISCRETISE '('<nom_Attribut>')'= <nom_discretise>:'('(
<intervalle> <nom_intervalle> ;)+ ')'
<nom_Attribut> ::= *id*Att
<nom_discretise> ::= *id*Att
<intervalle> ::= ([|])<val_numeric>, <val_numeric>([|])
<val_numeric> ::= ([+] | -)<Nombre>[.<Nombre>]
<Nombre> ::= <chiffre><chiffre>*
<chiffre> ::= 0| 1| 2| 3| 4| 5| 6| 7| 8| 9
<hierarchie> ::= HIERARCHY '('<nom_dimension>')' : '(' *id*Att→*id*Att [→ *id*Att]* ')'
<nom_dimension> ::= *id*Att
<spécification> ::= TO - <valeur> : (DETAIL | <Spécif_agrégat>)
<Valeur> ::= <Nombre> <Unité>
<Unité> ::= SECONDE| MINUTE| HOUR| DAY| WEEK| MONTH| YEAR
<Spécif_agrégat> ::= <measure> (,<measure>)* BY <membre_dimension>
(,<membre_dimension>)*

<measure> ::= <aggregate> ‘(’<nom_colonne>‘)’
<aggregate> ::= SUM| COUNT| AVG| MIN| MAX
<membre_dimension> ::= <nom_colonne>| <nom_discretise>| <Attribut_timestamp>|
<Attribut_temps>
<Attribut_timestamp> ::= DAY| MONTH| SEMESTER| QUARTER| YEAR
<Attribut_temps> ::= DAY‘(’<nom_colonne>‘)’| MONTH‘(’<nom_colonne>‘)’
| SEMESTER‘(’<nom_colonne>‘)’| QUARTER‘(’<nom_colonne>‘)’
| YEAR ‘(’ <nom_colonne>‘)’

3 Description des contraintes à vérifier

La cohérence des spécifications de fonctions d’oubli nécessite de vérifier des contraintes supplémentaires à la syntaxe du langage. Dans cette partie, pour des raisons d’espace, nous ne présentons que quelques contraintes (l’ensemble des contraintes devant être vérifiées n’est pas décrit).

Contrainte sur la dimension temporelle :

L’idée est de refuser d’agréger des données à un niveau d’ancienneté qu’elles n’ont pas encore atteint. Par exemple, les données de moins d’1 mois ne peuvent pas être agrégées au mois. La spécification suivante doit être rejetée :

TO – 1 MONTH : SUM (Revenu) BY MONTH ;

Contrainte sur les dimensions :

Pour agréger sur une dimension (ou sur un niveau d’une dimension), il faut l’inclure dès la première spécification. Par exemple, les spécifications suivantes sont incohérentes :

TO – 3 MONTH : AVG (Revenu) BY Sexe, DAY
TO – 1 YEAR : AVG (Revenu) BY Ville, MONTH ;

En effet, il est impossible dans la deuxième spécification de calculer l’agrégat sur Ville par la seule connaissance des agrégats sur Sexe et Jour.

Contrainte sur les agrégats :

Les agrégats des différentes spécifications doivent être compatibles entre eux. Par exemple, si on ne garde que le minimum d’une colonne, on ne peut plus déterminer la somme dans une spécification ultérieure. Les spécifications suivantes sont incohérentes :

TO – 3 MONTH : MIN(Revenu) BY Dept, DAY ;
TO – 1 YEAR : SUM(Revenu) BY Dept, MONTH ;

Les agrégats au niveau de la deuxième spécification devraient être inclus dans l’ensemble des agrégats de la première spécification. A noter que la moyenne peut se calculer à partir des agrégats SUM et COUNT.

Contraintes sur les hiérarchies :

Il est nécessaire de définir des contraintes pour assurer la cohérence de la définition d’une hiérarchie. Pour la hiérarchie (Ville-> Dept-> Région), deux villes de même nom ne doivent pas appartenir à des départements différents. Cette contrainte standard porte sur le contenu de la table Clients.

4 Stockage des données

Les spécifications de fonctions d’oubli permettent de déterminer les données qui doivent être présentes dans l’entrepôt de données à chaque application de la fonction d’oubli. Pour certaines données le détail est conservé. D’autres données sont agrégées alors que le détail

est supprimé, en fonction de l'ancienneté. Une solution pour gérer la fonction d'oubli est d'adopter la structure de données suivante :

- la table initiale pour stocker les données détaillées à conserver
- un cube de données pour les données agrégées

Ce cube est inégalement rempli (son contenu est une sorte de gruyère) car il stocke des données agrégées à des niveaux d'abstraction différents. Pour interroger les données d'un tel cube, on peut s'appuyer sur les travaux de Dyreson [Dyreson, 1996].

5 Perspectives

Les perspectives associées à ce travail en cours de développement sont nombreuses. Elles concernent principalement :

- L'extension du langage à la définition de fonctions d'oubli portant sur plusieurs tables liées entre elles par des contraintes de référence (clés étrangères) ;
- La définition d'autres critères d'oubli, comme l'importance ou l'utilité des données ;
- L'étude d'un stockage optimisé des données agrégées ;
- La définition d'un langage de requêtes pour exploiter les données historiques agrégées.

Références

[Cellary et al., 2001] W. Cellary, W. Gançarski, G. Jomier, M. Manouvrier, Les versions, Chapitre du livre « Bases de Données et Internet » A. Doucet et G. Jomier éditeurs, Hermes 2001.

[Chaudhuri et al., 2001] S. Chaudhuri, G. Das, V. Narasayya, A robust, optimisation-based approach for approximate answering of aggregate queries, Proceedings of SIGMOD Conference 2001.

[Chaudhuri et al., 1998] S. Chaudhuri, R. Motwani, V. Narasayya, Random sampling for histogram construction: how much is enough? In Proc. of ACM SIGMOD, 1998.

[Dumas et al., 2001] M. Dumas, C. Fauvet, P. Scholl, Modèles et langages pour données temporelles, Chapitre du livre « Bases de Données et Internet » A. Doucet et G. Jomier éditeurs, Hermes 2001.

[Dyreson, 1996] C. Dyreson, Information Retrieval from an Incomplete Data Cube, Proceedings of the 22nd VLDB conference Mumbai (Bombay), India, 1996, pp 532-543.

Summary

The amount of data stored in data warehouses grows very quickly so that they get saturated. We propose a solution to overcome this saturation problem. A language for specification of forgetting functions of older data is introduced. This language defines what data should be present in the data warehouse at each step of time. These specifications are based on the construction of aggregated summaries of older data, old detailed data being deleted at every update step.

The communication first describes the syntax of the specification language of forgetting functions. A special attention is given to the definition of the semantic constraints necessary to keep specifications coherent. Then, data structures are proposed for storing data necessary for the practical management of the forgetting functions.