Vers l'échantillonnage d'un entrepôt de données

Raphaël Féraud, Fabrice Clérot

France Télécom R&D, avenue Pierre Marzin, 22307 Lannion
Contact : raphael.feraud@orange-ftgroup.com

Résumé. L'afflux de données sur les usages des produits et services nécessite des traitements lourds pour les transformer en information. Or la capacité à traiter les données ne peut pas suivre l'augmentation exponentielle des volumes stockés. Avec les technologies actuelles, un difficile compromis doit être trouvé entre le coût de mise en œuvre et la qualité de l'information produite. Nous proposons une approche basée sur l'échantillonnage d'un entrepôt de données pour déployer à moindre coût un système d'information décisionnel utilisant tout notre potentiel d'information. La brique technologique essentielle pour construire ce système repose sur un opérateur d'échantillonnage des jointures.

1 Introduction

Une tendance lourde depuis la fin du siècle dernier est l'augmentation exponentielle du volume des données stockées. Les progrès de ces capacités de stockage ne se traduisent pas nécessairement par une meilleure compréhension de l'environnement. En effet, les données doivent être analysées afin de les transformer en connaissance or la capacité à traiter ces données n'a pas suivi cette augmentation exponentielle.

La première raison est simple : le processus d'analyse des données requiert une intervention humaine. Chaque augmentation de la couverture des domaines de données stockées induit une augmentation des équipes qui analysent ces données pour les transformer en information exploitable.

La seconde raison est liée à la capacité de traitement des données. Même si la puissance de calcul des ordinateurs suivait une pente équivalente à celle de la capacité de stockage (la loi de Moore), il existerait toujours des barrières algorithmiques bien plus infranchissables. Par exemple, pour un algorithme dont la complexité est en $O(n^2)$, le doublement d'une puissance de calcul ne permet d'augmenter que de $\sqrt{2}$ les possibilités de traitements.

Dès lors calculer des indicateurs pour alimenter un datamart devient problématique et demande une architecture technique coûteuse. De plus les requêtes ad hoc sont susceptibles de bloquer l'entrepôt de données. Elles sont en général prohibées ou fortement administrées de manière à prévenir l'engorgement du système d'information. Les techniques OLAP sont efficaces pour traiter des requêtes d'agrégation sur des cubes de données prédéfinis. Ces techniques manquent de souplesse et d'expressivité pour répondre à l'ensemble des requêtes ad hoc.

Avec nos technologies, il est difficile d'exploiter tout le potentiel de l'information contenu dans l'entrepôt.

2 Relever le défi de la volumétrie

Pour effectuer une analyse exploratoire sur des données très volumineuses, une approche consiste à approximer le résultat d'une requête (Hellerstein J. M. et al 1997, Haas P. J. et al 1999, Hellerstein J. M. et al 2000). La requête est lancée et elle s'arrête lorsque le degré de confiance est suffisant. Cette technique est souple et permet d'approximer toutes les requêtes. Elle présente néanmoins un défaut applicatif important : les requêtes sont lancées sur l'entrepôt de données. Lorsque quelques individus hors normes pèsent lourds dans le résultat d'une requête, comme une moyenne, il peut être nécessaire de parcourir toute la base pour obtenir un degré d'approximation satisfaisant. Cette limitation peut se révéler problématique lorsque l'entrepôt de données est bloqué par des requêtes exploratoires.

Une approche plus générale pour analyser des données très volumineuses est de produire un synopsis des requêtes. Le synopsis consiste en une structure de données permettant d'accélérer les traitements d'une requête. Le synopsis peut ensuite être interrogé de manière autonome pour des requêtes exploratoires. Le bon fonctionnement du système d'information décisionnel est garanti, tout en donnant la souplesse nécessaire à l'exploration des données.

L'histogramme des valeurs des attributs est le synopsis le plus courant et le plus simple à utiliser. Il permet d'accélérer le résultat d'une requête au prix d'un résultat approché (Ioannidis Y. et al 1995, Ioannidis Y. et al 1999, Aboulnaga A. et al 1999, Poosla V. et al 1996, Poosla V. 1997). Les ondelettes peuvent être utilisées pour décomposer les attributs de manière hiérarchique pour mieux contrôler le compromis entre la précision et la rapidité (Vitter J. S. et al, 1999). Les limites de cette approche sont atteintes lorsque la requête porte sur plusieurs attributs. De plus, son implémentation nécessite des changements considérables sur les moteurs de base de données pour créer et maintenir les histogrammes ou les décompositions en ondelettes.

Le projet AQUA développé par Bell propose un moteur d'approximation de requêtes basé sur des synopsis (Gibbons P. B. et al 1998, Gibbons P. B. et al 1999, Acharyal S. et al 1999-1, Acharyal S. et al 1999-2). Le synopsis utilisé est une jointure échantillonnée. L'ensemble des synopsis correspond à un sous-ensemble des jointures de la base de données. La principale limitation de cette approche est liée à la méthode d'échantillonnage : chaque synopsis doit nécessairement correspondre à une jointure sur clé étrangère, c'est-à-dire une jointure où la clé de jointure est une des clés primaires d'une des tables considérées. De plus, le graphe de connexion de la base de données ne doit pas avoir de cycles.

Notre approche pour relever le défi de la volumétrie est d'utiliser un synopsis se basant sur un échantillon de l'entrepôt. Cet échantillon est alimenté au même rythme que l'entrepôt. Il n'a pas besoin d'être recalculé pour être mis à jour. Pour la plupart des requêtes, une réponse approchée pourra être obtenue à partir d'opérations sur cet échantillon de l'entrepôt. L'échantillon de l'entrepôt sera alors interrogé de manière autonome. Pour les requêtes, plus complexes, plus rares, l'échantillon devra être complété en temps réel par une requête sur l'entrepôt. Nous garantissons ainsi le maximum de souplesse et de rapidité, tout en utilisant avec parcimonie les ressources de l'entrepôt. La brique technologique essentielle pour construire ce système repose sur un opérateur d'échantillonnage des jointures. Cet opérateur d'échantillonnage doit permettre d'échantillonner une jointure sans faire la moindre hypothèse sur la nature de la clé de jointure, afin de pouvoir échantillonner n'importe quel entrepôt de données. Dans la partie suivante, nous allons présenter un opérateur d'échantillonnage de jointures.

3 L'échantillonnage de jointures

3.1 Echantillonnage d'une équijointure sur clé primaire

Le cas le plus simple de l'échantillonnage d'une équijointure correspond au cas où la clé de jointure est la clé primaire d'une des deux tables. Dans ce cas, il existe une méthode permettant d'échantillonner la jointure sans la calculer complètement (Olken F.1993) . Soit R_1 et R_2 les deux relations à échantillonner. La clé primaire de R2 est utilisée comme clé de jointure. Pour $\lambda \in [0,1]$ on notera $S_\lambda(R)$ l'échantillon réalisé en retenant les instances de R qui satisfont au test $ALEA < \lambda, \lambda \in [0,1]$ où ALEA est un nombre aléatoire tiré pour chaque instance, uniformément dans [0,1] .La probabilité d'inclusion d'une instance dans l'échantillon est λ. En notant ECHλ(J) l'échantillon recherché de la jointure, on a :

$$ECH_\lambda(J) = S_\lambda(R_1 \rhd\lhd R_2) = R_1 \rhd\lhd S_\lambda(R_2)$$

Dans la suite du document nous nous concentrerons sur le cas général, où la clé de jointure n'est pas la clé primaire d'une des deux relations.

3.2 Echantillonnage d'une équijointure sur clé étrangère

3.2.1 Difficultés

L'approche triviale pour échantillonner une équijointure J consiste à calculer la jointure, puis à appliquer une fonction de tirage aléatoire :

$$ECH_\lambda(J) = S_\lambda(R_1 \rhd\lhd R_2)$$

L'échantillon obtenu est le meilleur possible. Le temps de traitement est important puisqu'il correspond au calcul de l'équijointure.

L'approche naïve pour échantillonner une équijointure J consiste à échantillonner chaque relation puis à les joindre :

$$ECH_\lambda(J) = S_\lambda(R_1) \rhd\lhd S_\lambda(R_2)$$

Le temps de traitement est ici réduit puisque la jointure est faite sur des tables préalablement échantillonnées. Qu'en est-il de la qualité de l'échantillon ?

Afin de mettre en relief la difficulté l'échantillonnage d'un ensemble de relations, nous avons repris et développé un exemple donné dans (Chaudhuri S. et al, 1999-1). Soit deux relations R_1 et R_2 :

$$R1 = \{(a_0,b_0);(a_1,b_1);...;(a_1,b_k)\}$$
$$R2 = \{(a_1,c_0);(a_0,c_1);...;(a_0,c_k)\}$$

Soit J l'équijointure de R_1 et R_2 avec la clé a. On a :

$$J = R1 \rhd\lhd R2 = \{(a_0,b_0,c_1),...,(a_0,b_0,c_k),(a_1,b_1,c_0),...,(a_1,b_k,c_0)\}$$

$S_\lambda(R_1 \rhd\lhd R_2)$ contiendra en moyenne $2\lambda k$ éléments, avec une faible probabilité d'être égal à l'ensemble vide :

$$P\{S_\lambda(R_1 \rhd\lhd R_2) = \emptyset\} = (1-\lambda)^{2k} \xrightarrow[k\to\infty]{} 0$$

Il est en revanche beaucoup plus probable que $S(R1,\lambda) \rhd\lhd S(R2,\lambda)$ corresponde à l'ensemble vide :

$$P\{S_\lambda(R1) \rhd\lhd S_\lambda(R2) = \emptyset\} \geq P\{a_0 \notin S_\lambda(R1)\}.P\{a_1 \notin S_\lambda(R2)\}$$
$$\Leftrightarrow P\{S_\lambda(R1) \rhd\lhd S_\lambda(R2) = \emptyset\} \geq (1-\lambda)^2$$

On a donc :

$$P\{S_\lambda(R1) \rhd\lhd S_\lambda(R2) = \emptyset\} \underset{k\to\infty}{\not\to} 0$$

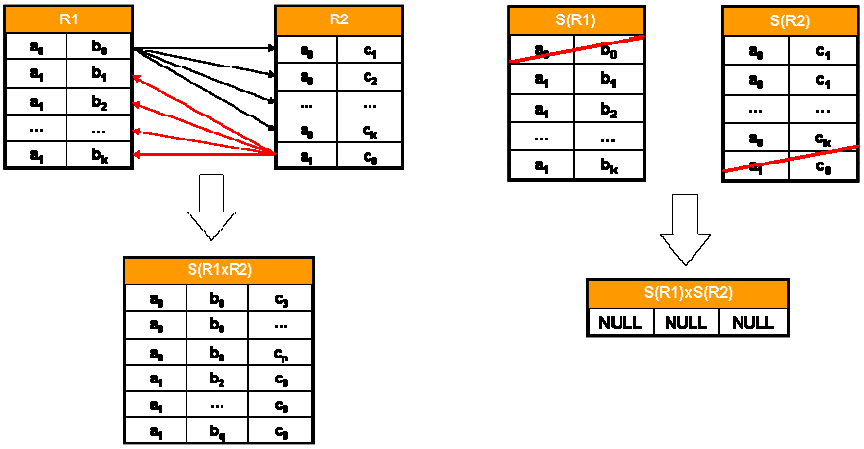

Figure 1 : l'approche triviale donne le meilleur résultat possible mais nécessite le calcul préalable de la jointure. L'approche naïve est rapide, mais peut conduire à une mauvaise estimation de la jointure.

L'approche naïve, si elle est rapide, ne conduit pas nécessairement au même résultat que l'approche triviale. S n'est pas distributif sur l'équijointure. C'est ce qui rend difficile le problème de l'échantillonnage d'une équijointure de deux relations.

3.2.2 Principales approches

Olken et Rotem (1986,1993) ont proposé un algorithme permettant d'échantillonner une jointure sans la calculer préalablement dans le cas où la clé de jointure n'est pas la clé primaire d'une des deux relations :

$$S_k(R1,R2) \leftarrow WR\left[k, ACCEPT\left(\frac{|R2.x_i|}{|R2.x|_{\max}}, \psi_1(\psi_1(R1) \rhd\lhd R2)\right)\right]$$

Ici, S_k(R1, R2) est un échantillon de la jointure comportant k instances. WR(k,<expression>) signifie que <expression> est évaluée jusqu'à ce qu'un ensemble de k éléments soit obtenu, avec remise des éléments sélectionnés. ACCEPT(α,<expression>) indique que l'élément évalué dans <expression> est accepté avec la probabilité α.

ψ_1(R) correspond au tirage d'une instance de R, x_i est la valeur de la clé de jointure correspondant à l'instance tirée, $|R.x_i|$ est la cardinalité de la valeur x_i dans la relation R et $|R.x|_{\max} = \max_i |R.x_i|$.

Les auteurs ont montré que cette méthode d'échantillonnage est équivalente à calculer la jointure puis à l'échantillonner; on peut remarquer que seule la taille (k instances) de l'échantillon est connue: pour connaître le taux d'échantillonnage, il faut au préalable connaître la taille de la jointure. Cette méthode d'échantillonnage nécessite de posséder des index sur R_1 et sur R_2, ainsi que les fréquences de la clé de jointure pour la relation R_2. En outre, la méthode de Olken et Rotem (1986,1993), basée sur un algorithme de rejet, est relativement coûteuse. Le nombre de tirages et de tests pour chaque n-uplet de l'échantillon est de l'ordre de (Chaudhuri S. et al, 1999-2) :

$$\frac{\left|R_2.x\right|_{\max}.\left|R_1\right|}{\sum_i \left|R_1.x_i\right|.\left|R_2.x_i\right|}$$

Une approche pour tirer efficacement l'échantillon est d'effectuer un tirage biaisé sur le flux de données (Chaudhuri S. et al, 1999-2). L'algorithme se divise en deux phases :

- La première phase utilise un réservoir avec des poids $\left|R_2.x(t)\right|$ pour produire un échantillon S_1 de R_1 biaisé sur la fréquence de la clé de jointure de chaque n-uplet t dans R_2.
- La deuxième phase consiste à joindre chacun des n-uplets t1 de l'échantillon R_1 avec un n-uplet t_2 tiré aléatoirement tel que $t_2.x=t_1.x$.

Chaque n-uplet de l'échantillon est tiré en une seule passe. L'index de la clé de jointure dans R_2, et la connaissance des fréquences de la clé de jointure dans R_2 sont nécessaires au fonctionnement de l'algorithme. Les auteurs ont proposé des améliorations de cet algorithme pour tenir compte des individus hors norme (Chaudhuri S. et al, 1999-2).

Les méthodes d'échantillonnage d'une équijointure peuvent être classées en trois catégories (Chaudhuri S. et al, 1999-2) :

- Cas A : aucune information n'est utilisée sur R_1 ou R_2.
- Cas B : un index et des statistiques sur la clé de jointures dans R_2 sont utilisés.
- Cas C : un index et des statistiques sur la clé de jointures dans R_1 et R_2 sont utilisés.

Les méthodes appartenant à la catégorie B ou C utilisent la fréquence de la clé de jointure sur une table pour biaiser le tirage sur l'autre table. Ces méthodes sont difficilement généralisables à n équijointures. Considérons par exemple le cas d'un échantillon de la jointure de trois tables $R_1 \rhd\lhd R_2 \rhd\lhd R_3$. Utiliser une méthode de la catégorie B ou C nécessite de disposer de statistiques et d'index sur $R_2 \rhd\lhd R_3$ ou de calculer au préalable la jointure $R_1 \rhd\lhd R_2$ pour se ramener au cas à deux jointures. Ces approches sont difficilement applicables pour calculer des échantillons d'un nombre variable de jointures. Nous nous trouvons devant un paradoxe : pour échantillonner une équijointure, il est nécessaire de biaiser le tirage en fonction de la fréquence de la clé de jointure, mais pour échantillonner n équijointures, il n'est pas envisageable de disposer d'index et des fréquences de la clé de jointures sur toutes les équijointures partielles.

Nous proposons une approche permettant de dépasser ce paradoxe.

3.3 Echantillonnage d'équijointures par hachage

3.3.1 Présentation de l'algorithme

Soit X l'ensemble des valeurs de la clé de jointure x dans R_1 et dans R_2, et h(x) une fonction de hachage bornée par N. Soit $T_1^k = \{t_1 \in R_1 \text{tels que } h(t_1.x) = k\}$, l'ensemble des n-uplets de R_1 dont le hachage suivant x a pour valeur k, et $T_2^k = \{t_2 \in R_2 \text{tels que } h(t_2.x) = k\}$, l'ensemble des n-uplets de R_2 dont le hachage suivant x a pour valeur k.

Si la fonction de hachage choisie produit un hachage parfait de X, il n'y a pas de collisions de clés différentes dans la table de hachage. En conséquence, quelque soit x_h, il existe $k = h(x_h)$ tel que : $T_1^k = R_1.x_h$ et $T_2^k = R_2.x_h$

$$\Rightarrow \frac{|T_1^k|.|T_2^k|}{\sum_j^N |T_1^j|.|T_2^j|} = \frac{|R_1.x_h|.|R_2.x_h|}{\sum_i^{|X|} |R_1.x_i|.|R_2.x_i|} = P(x_h)$$

où $P(x_h)$ est la probabilité d'occurrence de la valeur x_h dans X l'ensemble des valeurs possibles de la clé de jointure. Lorsqu'il n'y a pas de collisions, la construction des tables de hachage T_1^k, T_2^k permet :

- de calculer la probabilité de tirage suivant la fréquence des valeurs de la clé de jointure dans $J = R_1 \rhd\lhd R_2$.
- de fournir un index de R_1, R_2 permettant d'accéder par la clé de jointure aux n-uplets.

Ces deux propriétés sont à la base de l'algorithme d'échantillonnage d'équijointure par hachage (Figure 2).

$S = 0$
Pour k de 1 à N
  Pour i de 1 à m
    $T_i^k = 0$
Pour i de 1 à n
  Pour j de 1 à $|R_i|$
    $k = h(t_i^j.x)$
    $T_i^k = T_i^k + \{t_i^j\}$
    $|T_i^k| = |T_i^k| + 1$

Pour k de 1 à N, Calcul de $P(k) = \dfrac{\prod_i^m |T_i^k|}{\sum_j^N \prod_i^m |T_i^j|}$

Tant que $|S| < \lambda |J|$
  On tire k suivant $P(k)$
  Pour i de 1 à m
    On tire $t_i \in T_i^k$ uniformément
  Si $t_1.x = t_2.x = ... = t_m.x$ alors $S = S + \{t_1 t_2 ... t_m\}$

Figure 2 : Algorithme d'échantillonnage de n équijointures par hachage

Si le hachage est parfait, l'échantillon ainsi obtenu contient un échantillon de $\lambda.|J|$ n-uplets tirés suivant la fréquence de la clé de jointure dans $J = R_1 \triangleright\triangleleft R_2$. On peut noter que pour obtenir un échantillon correspondant à un taux d'échantillonnage fixé à l'avance, il faut connaître la taille de la jointure $R_1 \triangleright\triangleleft R_2$; il existe des techniques rapides permettant d'estimer cette taille (Alon N. et al, 1999). Si on ne connaît pas la taille de la jointure, on doit se contenter de fixer à l'avance la taille de l'échantillon. Si des collisions se produisent, les n-uplets sont tirés suivant une approximation de la fréquence de la clé de jointure. Dans ce cas, l'échantillon pourrait contenir des n-uplets ne faisant pas partie de la jointure. C'est pourquoi, un test a été rajouté avant l'inclusion du n-uplet dans l'échantillon (Figure 2).

On remarquera que cet algorithme ne nécessite aucune connaissance sur les relations R_1 et R_2. Si les relations R_1 et R_2 sont indexées par la clé de jointure, la fonction de hachage est remplacée par l'index afin d'obtenir un hachage parfait. Comme les statistiques sur la clé de jointure et l'index de la clé de jointure sont évalués par le hachage des relations, la méthode d'échantillonnage par hachage est naturellement généralisable à n équijointures (Figure 2)

3.3.2 Complexité algorithmique

La complexité de cet algorithme d'échantillonnage de m équijointures est de l'ordre de :

$$C = 2.A.O(n.m) + B.O(N) + D.O(\frac{\lambda.m.|J|}{1-P(c)})$$

Le premier terme représente le nombre d'appels à la fonction de hachage. A représente le coût associé à la fonction de hachage. n représente le cardinal de la plus grande des relations de la jointure. m est le nombre d'équijointures à réaliser, on a : $m << n$

Le second terme correspond au calcul de la fonction de probabilité. B représente le coût du calcul de P(k). N est le nombre de cases de la fonction de hachage. Un choix judicieux de N correspond à un nombre premier de l'ordre du nombre de valeurs de la clé de jointure. On a donc $N < n$.

Le dernier terme est le nombre de tirages dans les tables de hachage T_i^k . D est le coût de tirage d'un n-uplet suivant P(k). P(c) représente la probabilité de collisions dans les tables de hachage. Dans l'implémentation proposée, plus le nombre de collisions est important, moins l'algorithme est rapide. Une autre approche moins couteuse en temps de calcul consisterait à parcourir a posteriori l'échantillon pour supprimer les n-uplets ne faisant pas partie de la jointure. Dans ce cas la taille de l'échantillon généré et donc la qualité de l'estimation serait moindre.

3.3.3 La fonction de hachage

Nous considérerons que les clés de nos données sont des nombres entiers. Si tel n'est pas le cas, une transformation préalable en nombre entier est nécessaire. Nous proposons d'utiliser comme fonction de hachage, le reste de la division entière par un entier N :

$$h(x) = x \bmod N$$

Cette fonction de hachage a l'avantage d'être très rapide à calculer. Pour minimiser le risque de collisions, nous choisirons N comme étant un nombre premier le plus proche possible de $|X|$, le nombre de clés différentes. Il existe des techniques rapides permettant d'estimer ce

nombre avec très peu de mémoire et en une passe sur les tables (Cormode et al, 2002). Si ce nombre ou plutôt son ordre de grandeur n'est pas connu, nous fixerons N à un très grand nombre premier.

4 Comparaison des méthodes

4.1 Introduction

Nous avons choisi comme points de référence la méthode triviale et la méthode naïve. La méthode naïve consiste à échantillonner les deux tables puis de faire leur jointure afin de produire un échantillon de la jointure. Cette approche est assurément la plus rapide, mais comme nous l'avons vu précédemment sa précision peut être faible. La méthode triviale consiste à calculer la jointure puis tirer un échantillon de la jointure. C'est une méthode lente, lorsque les tables ne tiennent pas en mémoire, mais précise. Le positionnement de la méthode par hachage par rapport à ces points de références permettra d'en évaluer le potentiel.

4.2 Le coût de traitement

Afin de ne pas comparer les implémentations des méthodes et les conditions expérimentales, pour comparer le coût de traitement, nous nous appuierons sur la complexité algorithmique. La complexité algorithmique de la méthode triviale correspond au coût de calcul d'un tri :

$$C_1 = 2.A_1.O(n\log(n)) + B_1O(\lambda.|J|)$$

A_1 mesure le coût d'une opération de comparaison et B_1 le coût d'une sélection d'une instance dans la jointure. Généralement le premier terme est grand devant le second terme de l'équation.

La complexité algorithmique de la méthode naïve s'obtient de la même manière :

$$C_2 = 2.A_1.O(\lambda.n\log(\lambda.n)) + B_1O(\lambda.|J|)$$

On a $C_1 > C_2$.

La complexité algorithmique de la méthode d'échantillonnage par hachage est :

$$C_3 = 2.A_2.O(n) + B_2.O(N) + D_2.O(\frac{\lambda.|J|}{1-P(c)})$$

Cette méthode d'échantillonnage a un coût qui dépend du paramètre N. Plus N est grand, plus le second terme de l'équation est important et plus la probabilité de collision tend vers 0. Nous avons tracé sur un exemple le temps de traitement nécessaire à l'échantillonnage d'une jointure en fonction du choix de N (Figure 3). On remarquera que le temps de traitement diminue à mesure que N augmente. Il n'y a pas de différence significative de temps de traitement pour N=39 989 ou N=99 991. En effet, le nombre de collisions tend vers 0, alors que N est encore très faible devant n= 6 044 279. On peut donc en conclure que pour un choix judicieux de N, par exemple le premier nombre premier supérieur au nombre de valeurs distinctes de la clé de jointure, on a :

$$C_3 = 2.A_2.O(n) + D_2.O(\lambda|J|) + K$$

K est le coût associé au calcul de la fonction de probabilité de tirage des valeurs de la clé de jointure. K est petit devant les deux premiers termes de l'équation.

Le premier terme de C_3 est toujours inférieur au premier terme de C_1. Lorsque les clés primaires des tables tiennent en mémoire, on peut considérer que A_1, qui représente le coût d'une opération de comparaison, et A_2 qui représente l'appel à la fonction de hachage sont similaires. En revanche dans le cas de la méthode par hachage le nombre d'opérations est en O(n) alors que pour la méthode triviale le nombre d'opération est en O(n log (n)).

Si les clés primaires des tables ne tiennent pas en mémoire, le premier terme de C_1 devient très grand devant C_3 puisqu'il faut rajouter à A_1 le coût d'un accès aléatoire aux données stockées sur un disque alors que pour A_2 l'accès est séquentiel.

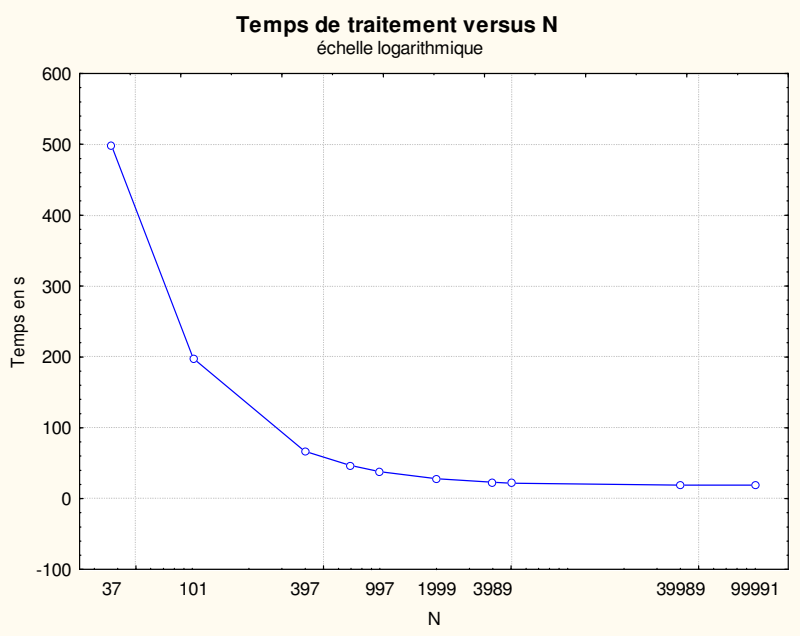

Figure 3 : Influence de N sur le temps de traitement

La comparaison du second terme des équations C_1 et C_3 dépend du nombre de clés de jointure. En effet, dans le cas de la méthode par hachage, il est indispensable de faire un accès aléatoire aux tableaux contenant les clés de jointure. Pour la méthode triviale un accès séquentiel est suffisant. Si les tableaux tiennent en mémoire les temps de traitements associés au second terme sont comparables. Si ces tableaux ne tiennent pas en mémoire, le temps de traitement associé au second terme sera beaucoup plus important pour la méthode de hachage. On remarquera néanmoins que si les clés de jointures ne tiennent pas en mémoire, les clés primaires ne peuvent tenir en mémoire. Dans ce cas, le premier terme de C_1 devient grand devant C_3.

En conclusion, la méthode par hachage est moins couteuse en temps de traitement que la méthode triviale. Elle peut néanmoins demander un effort d'implémentation supérieur.

4.3 Expérimentations

4.3.1 La base de données

Pour tester les différentes méthodes d'échantillonnage, nous avons utilisé un extrait d'une base de données issue du domaine des télécommunications.

La table des tiers représente la table des clients. Ces tiers sont liés à 1 à n utilisateurs d'un ou plusieurs services de télécommunication (email, téléphone fixe, mobile…). La table des éléments de parc contient la liste des services souscrits. Chaque client possède 0 à n services. La table des comptes-rendus d'usage contient les traces laissées par l'usage des services. Chaque utilisateur a généré 0 à n comptes-rendus d'usage. Chaque service ou élément de parc a généré 0 à n comptes-rendus d'usage.

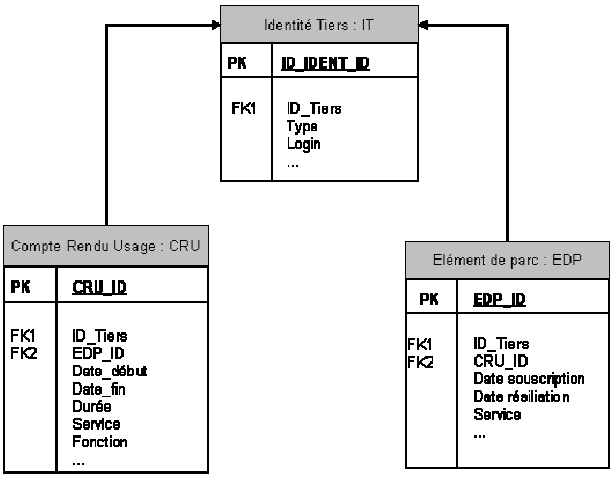

Figure 4 : Le modèle de données représente l'usage de services par des utilisateurs

La table des comptes-rendus est la plus volumineuse. Elle regroupe de l'ordre de 6 millions d'enregistrements. La table des éléments de parc compte approximativement 80000 enregistrements. Les utilisateurs sont un peu plus de 15000. Dans le Tableau 1 nous avons caractérisé chacune des jointures produites sur ces tables sur la clé de jointure du client ID_Tiers. Cette clé n'est pas la clé primaire des tables Identité Tiers, Compte Rendu d'Usage et Elément de parc.

R1><R2	|R1|	|R2|	| R1><R2|	F_In_Moy	F_Out_Moy
EDPxCRU	81579	6044279	39192792	486,2328	6,486334
IT x CRU	14819	6044279	6043231	413,2689	1
IT x EDP	14819	81579	81573	5,506480	1,002482

Tableau 1 : Description des jointures. F_In indique le "fan-in", nombre d'enregistrements de R1 joints à un enregistrement de R2; F_Out indique le "fan-out", nombre d'enregistrements de R2 joints à un enregistrement de R1.

La jointure EDP x CRU est la plus volumineuse. C'est aussi la plus éloignée d'une jointure sur clé primaire : chaque enregistrement de la table EDP correspond en moyenne à 486 enregistrements de la table CRU, et inversement, à un n-uplet de la table CRU correspond en moyenne à 6.5 n-uplets dans la table EDP. La jointure IT x CRU est une jointure équivalente à une jointure sur la clé primaire de IT. IT x EDP est une jointure très proche de la clé primaire. Dans la suite nous exposerons les résultats des différentes méthodes sur la jointure EDP x CRU.

4.3.2 Estimation des fréquences

Nous avons choisi d'estimer les fréquences des valeurs de la clé de jointure obtenues à partir des différentes méthodes d'échantillonnage pour les comparer. La précision d'un estimateur se décompose en deux grandeurs, respectivement la variance et le biais :

$$E(T-\theta)^2 = Var(T) + (E(T)-\theta)^2$$

Nous avons construit dix échantillons de 5% de la jointure pour chacune des méthodes d'échantillonnage. Nous avons calculé la valeur moyenne de la précision, du biais et la variance des estimateurs des fréquences des valeurs de la clé de jointure (Tableau 2).

	Précision moyenne	Biais moyen	Variance moyenne
Hachage	4.86	1.07	4.74
Naïve	8.84	5.4	8.73
Triviale	4.87	1.11	4.76

Tableau 2 : Précision, biais et variance moyenne des estimateurs des fréquences de la clé de jointure. Afin de faciliter la lecture des résultats, nous avons appliqué la transformation suivante aux valeurs : $x' = \log(10^9 x)$.

La différence de qualité d'estimation est très nette entre la méthode naïve et les méthodes triviales et par hachage. Comme attendu, la méthode naïve est peu précise et peu robuste. La précision et la variance des estimateurs obtenus par la méthode par hachage ou triviale sont similaires. Les estimateurs proposés sont tous sans biais. Néanmoins, les biais de ces estimateurs basés sur des méthodes d'échantillonnage différentes ne tendent pas à la même vitesse vers 0. La méthode naïve présente un biais supérieur et une vitesse de convergence inférieure aux méthodes d'échantillonnage par hachage et triviale (Figure 5).

Nous avons construit dix échantillons de 5%, 0.5 % et 0.05 % de la jointure afin de comparer les vitesses de convergence des estimateurs des fréquences en fonction de la taille des échantillons. La précision des estimateurs converge plus rapidement pour les méthodes triviales et par hachage que pour la méthode naïve (Figure 6).

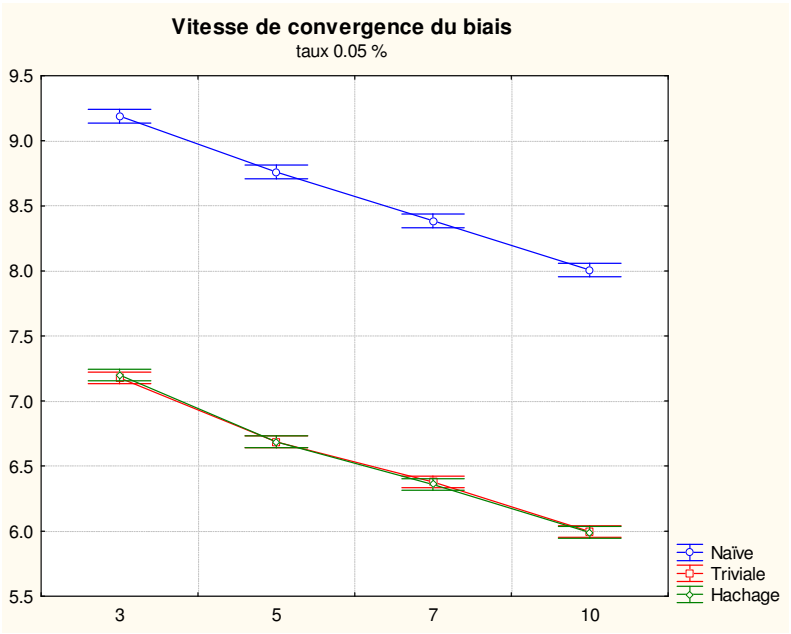

Figure 5 : Vitesse de convergence du biais en fonction du nombre d'échantillons

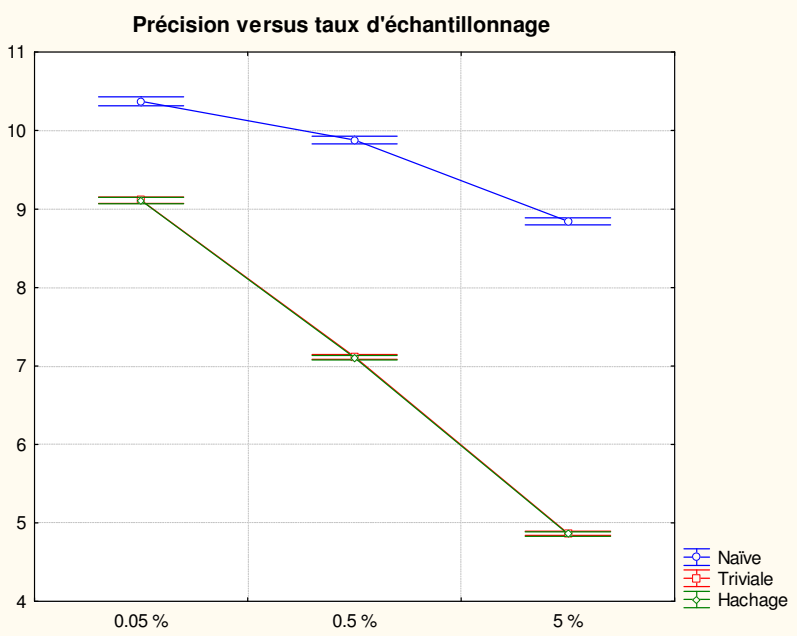

Figure 6 : Vitesse de convergence de la précision en fonction du taux d'échantillonnage

5 Conclusion

Nous avons proposé une méthode d'échantillonnage d'une jointure par hachage. Contrairement aux méthodes proposées dans l'état de l'art, cette méthode peut être étendue à n jointures. De plus, cette méthode ne nécessite aucune connaissance particulière sur les relations à échantillonner. Elle est utilisable comme opérateur élémentaire pour générer l'échantillon d'un entrepôt de données quelconque.

Nos résultats expérimentaux montrent qu'elle permet de construire des estimateurs d'une précision comparable ceux obtenus en échantillonnant une jointure préalablement calculée. Le coût de traitement théorique de cette méthode d'échantillonnage est inférieur au calcul de la jointure.

La méthode par hachage nécessite cependant un accès aléatoire aux clés de jointures des relations à échantillonner. Si ces clés ne tiennent pas en mémoire, dans une implémentation naïve, l'algorithme peut être considérablement ralenti. Une solution pour prendre en compte cette contrainte d'implémentation consiste à diviser par hachage de la clé de jointure le schéma d'échantillonnage en sous schémas traités indépendamment.

Une autre voie intéressante serait d'adapter cette méthode à l'échantillonnage d'un flux de données avec une table en calculant en ligne la probabilité de tirage d'une clé.

Références

Aboulnaga A. and Chaudhuri S. (1999), *Self-Tuning Histograms: Building Histograms Without Looking at Data*. ACM SIGMOD.

Acharya S., Gibbons P. B., Poosala V., and Ramaswamy S. (1999-1), *The Aqua approximate query answering system*. ACM SIGMOD International Conf. on Management of Data (SIGMOD '99).

Acharya S., Gibbons P. B., Poosala V., and Ramaswamy S. (1999-2), *Join Synopses for Approximate Query Answering*. ACM SIGMOD International Conf. on Management of Data (SIGMOD '99), pp. 275-286.

Alon N., Gibbons P., Matias Y., and Szegedy M. (1999). *Tracking Join and Self-Join Sizes in Limited Storage*. 18'th ACM Principles of Database Systems conference, ACM Press, New York, pages 10-20.

Cormode G., Datar M., Indyk P., and Muthukrishnan S. (2002). *Comparing data streams using hamming norms* (how to zero in). 28th International Conf. on Very Large Data Bases (VLDB), pages 335-345.

Chaudhuri S., Motwani R. (1999-1). *On Sampling and Relational Operators*. IEEE on Data Engineering, 1999

Chaudhuri S., Motwani E., and Narasayya V. (1999-2). *On Random Sampling over Joins*. ACM SIGMOD.

Gibbons P. B. and Matias Y. (1999). *Synopsis data structures for massive data sets*. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization.

Gibbons P. B., Poosala V., Acharya S., Bartal Y., Matias Y., Muthukrishnan S., Ramaswamy S., and Suel T. (1998). *AQUA: System and techniques for approximate query answering. tech. rep*. Bell Laboratories, Murray Hill, New Jersey, .http://citeseer.ist.psu.edu/gibbons98aqua.html.

Haas P. J. and Hellerstein J. M. (1999). *Ripple Joins for Online Aggregation*. ACM SIGMOD.

Hellerstein J. M., Avnur R., and Raman V. (2000*). Informix under CONTROL: Online Query Processing*. Data Mining and Knowledge Discovery Journal.

Hellerstein J. M., Haas P. J., Wang H. J. (1997). *Online Aggregation*. In Proc Of ACM SIGMOD.

Ioannidis Y. and Poosala V. (1995). *Balancing Histogram Optimality and Practicality for Query Result Size Estimation*. ACM SIGMOD.

Ioannidis Y. and Poosala V. (1999). *Histogram-Based Approximation of Set-Valued Query Answers*. VLDB.

Olken F. (1993). *Random Sampling from Databases*. PhD thesis, U.C. Berkeley.

Olken F., Rotem D. (1986). *Simple random sampling from relational databases*. 12th VLDB, pages 160-169.

Poosala V., Loannidis Y.,Haas P., and Shekita E. (1996). *Improved Histograms for Selectivity Estimation of Range Predicates*. ACM SIGMOD.

Poosala V. (1997). *Histogram-Based Estimation Techniques in Database Systems*. PhD Thesis, Univ. of Wisconsin.

Vitter J. S., and Wang M. (1999). Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. ACM SIGMOD.

Summary

Turning the wealth of customer usage data into valuable information requires very demanding treatments and the increase in processing power cannot cope with the dramatically increasing volumes of data. Even with state-of-the-art technologies, a difficult balance has to be found between the processing cost and the quality of the information. We propose an approach based on the sampling of the data-warehouse as a means to deploy a low-cost information system while using all our potential information. This approach leverages a joint sampling technique which is accurate while avoiding to build the joint itself.