Réduction du nombre des prédicats pour les approches de répartition des entrepôts de données

Mourad Ghorbel[1], **Karima Tekaya**[2], **Abdelaziz Abdellatif**[3]

1. URAPOP, Université de Tunis El Manar
Faculté des Sciences de Tunis, El Manar 2092, Tunisie
ghorbel.fst@gmail.com

2. Université de Tunis
École supérieure des sciences économiques et commerciales de Tunis
Montfleury 1089, Tunisie
Karima.tekaya@gmail.com

3. LIPAH, Université de Tunis El Manar
Faculté des Sciences de Tunis, El Manar 2092, Tunisie
abdelaziz.abdellatif@fst.rnu.tn

RÉSUMÉ. *Dans le domaine des entrepôts de données, la plupart des approches de répartition se basent essentiellement sur les techniques de fragmentation et d'allocation des tables. Ces approches exploitent communément en entrée les prédicats extraits des requêtes OLAP les plus utilisées dans le processus de partitionnement. Étant donné que le nombre de prédicats est en augmentation continue, et vu l'impact négatif qu'engendre cette augmentation sur le nombre de partitions générées, il devient intéressant de le réduire avant de procéder au processus de fragmentation. Dans cet article, nous proposons une solution basée sur un algorithme de classification permettant de diminuer le nombre des prédicats pour les approches de répartition des entrepôts de données. La solution proposée englobe quatre phases : la sélection préliminaire des prédicats, la codification des prédicats sous forme de matrices binaires, la classification de ces prédicats par l'algorithme k-means et une phase finale pour la réduction du nombre de prédicats. Nous avons validé notre solution sur un entrepôt de données réel issu du benchmark APB-1 et TPC-H.*

ABSTRACT. *In the domain of data warehousing, most approaches of distribution are essentially based on the techniques of fragmentation and allocation tables. These approaches exploit in input extracts predicates of OLAP queries most used in the partitioning process. Since continues increase of the number of predicates, and her negative impact, it becomes more and more interesting to reduce this increase before the fragmentation process. In this paper, we propose a solution based on a classification algorithm to reduce the number of predicates in the data warehouses allocation approaches. The proposed solution encompasses for phases: a preliminary phase for predicates selection, a predicates coding phase as binary matrices, a classification phase of these predicates using the k-means algorithm and a final phase to*

reduce the number of predicates. We validate our solution on a real data warehouse basing on the APB1 and TPC-H benchmarks.

MOTS-CLÉS : entrepôt de données, classification, répartition.

KEYWORDS: data warehouse, classification, distribution.

DOI:10.3166/ISI.21.1.81-102 © 2015 Lavoisier

1. Introduction

Aujourd'hui, les systèmes d'informations décisionnels (SID) font face à de nouvelles contraintes notamment l'augmentation du volume et la variété des données, la décentralisation des structures de prise de décision et des bases de données (BD) opérationnelles, conjuguées aux récentes avancées technologiques en matière de télécommunication. Ces nouvelles contraintes appellent à l'élaboration du SID selon une toute nouvelle perspective : les solutions d'entrepôts de données (ED) selon une approche répartie. Ainsi, des techniques qui étaient jusque-là réservées aux bases de production sont de plus en plus envisagées pour les ED. Il s'agit de la répartition et de la fragmentation des données.

Dans ce contexte, l'adaptation de l'organisation de l'ED à l'organisation décentralisée de l'entreprise commence tout d'abord, par la fragmentation des tables de l'ED puis, leur répartition sur les différents sites de l'entreprise selon les besoins des décideurs.

La fragmentation peut être horizontale, verticale ou mixte (Darmont, 2006). Elle est basée sur une liste de prédicats extraite à partir des requêtes OLAP (*On Line Analytical Processing*) locales aux utilisateurs ou distantes. Ces prédicats proviennent du jeu des requêtes posées par les utilisateurs (clause WHERE). Plusieurs techniques de fragmentation ont été proposées dans l'état de l'art (Boukhalfa, 2009 ; Barr, 2010 ; Ziyati, 2010) et (Bellatreche *et al.*, 2011), mais leur efficacité reste à prouver lorsque le nombre de prédicats augmente.

En effet, l'augmentation du nombre de prédicats est une conséquence évidente suite à l'évolution des besoins des SID. Cette augmentation implique l'accroissement du nombre de partitions générées indépendamment de la démarche de fragmentation utilisée, le processus de contrôle et de gestion des partitions devient de plus en plus complexe ce qui risque de provoquer un problème d'efficacité de l'ED.

Malgré l'intérêt accordé aux techniques de fragmentation des données et la diversité des solutions proposées, nous avons constaté que ce problème n'a pas bénéficié de l'attention qu'il mérite en dépit de son importance.

Dans notre travail, nous nous sommes concentrés sur le problème de réduction du nombre de prédicats dans le cas de la fragmentation horizontale (FH) des tables. Nous proposons une approche de réduction des prédicats qui peut constituer une

base de travail pour les techniques de fragmentation les plus utilisées (Bellatreche, 2008 ; Tekaya, 2011 ; Derrar *et al.,* 2013). Cette approche est testée sur un ED réel issu du benchmark APB1 (Spofford, 1998). En partant d'un ED centralisé, nous utilisons notre solution de fragmentation horizontale que nous avons proposé dans (Ghorbel *et al.,* 2014) pour le partitionnement de l'ED. Ensuite, nous comparons deux scénarios : un ED partitionné sans réduction du nombre de prédicats et un ED partitionné et réparti avec réduction du nombre de prédicats. Nous considérons les résultats obtenus encourageant.

Le présent article est construit ainsi : dans la première section, nous présentons un tour d'horizon sur l'état de l'art comportant une variété de travaux reliés à la fragmentation des ED. Dans la deuxième section, nous décrivons les différentes phases de notre approche de réduction des prédicats. Dans la troisième section, notre approche est concrétisée à travers son application sur un ED réel issu du benchmark APB1 (Spofford, 1998).

2. État de l'art

Les vertus de la fragmentation dans les ED ont été prouvées durant la dernière décennie. En favorisant un accès plus rapide aux données pertinentes et un maximum de parallélisme intra-requête, elle permet de réduire le temps d'exécution des requêtes OALP. Combinée avec d'autres techniques d'optimisation (index, vues matérialisées, etc.), son utilisation est devenue une solution souvent adoptée dans le domaine des SID. Plusieurs solutions ont été proposées pour la fragmentation des ED relationnels. Ces techniques exploitent communément la liste des prédicats comme base de travail. Dans ce qui suit, nous présentons un tour d'horizon sur les approches de fragmentation les plus connues dans l'état de l'art et nous montrons que la plupart des solutions s'orientent vers la technique de fragmentation sans tenir compte de l'impact de l'évolution du nombre de prédicats sur l'efficacité de la solution.

Bellatreche *et al.* (Bellatreche *et al.,* 2004) ont proposé une solution pour la fragmentation horizontale des tables multidimensionnelles de l'ED. Cette solution est une adaptation de l'approche basée sur les affinités du travail de (Zhang et Orlowska, 1995) développée initialement pour la fragmentation verticale des tables relationnelles. La solution utilise une liste de prédicats initiale extraite des requêtes OLAP. Mais, aucune optimisation préliminaire n'a été effectuée.

Derrar *et al.* (Derrar *et al.,* 2008) proposent une approche de répartition des données de l'entrepôt. Elle est basée sur une méthode issue des approches biomimétiques pour remédier la variation des requêtes décisionnelles dans le temps et leurs effets sur les schémas de fragmentation. L'approche proposée exploite une liste de prédicats initiale non optimisée.

Bellatreche *et al.* (Bellatreche *et al.,* 2011) ont proposé un algorithme de sélection des meilleurs schémas de fragmentation en combinant un modèle de coût

mathématique de (Bellatreche, 2008) avec l'algorithme du recuit simulé. Dans le même contexte, (Boukhalfa, 2009) a proposé un ensemble d'approches permettant d'optimiser les ED. Ses approches d'optimisation reposent sur l'utilisation de trois techniques d'optimisation : la FH primaire, dérivée et les index de jointure binaires (IJB). Les résultats obtenus par (Bellatreche *et al.*, 2011 et Boukhalfa, 2009) sont très motivant mais les auteurs n'ont pas étudié l'impact de l'évolution du nombre des prédicats sur l'efficacité des approches proposées.

Ziyati (Ziyati, 2010) a formalisé le problème de sélection du meilleur schéma de fragmentation verticale comme un problème d'optimisation avec contrainte. Il a adapté l'algorithme génétique pour la fragmentation des ED. Sa solution commence par fragmenter le schéma relationnel d'un ED horizontalement, ensuite, verticalement afin de réduire le coût d'exécution des requêtes. Barr (Barr, 2010) a utilisé un algorithme basé sur les colonies de fourmis pour le partitionnement des tables de l'ED. Derrar *et al.* (Derrar *et al.*, 2012) ont exploité leur travail (Derrar *et al.*, 2008) pour la réalisation des schémas de partitionnement optimaux et ont proposé une approche basée sur l'exploitation de l'accès aux données statiques pour la fragmentation de données dynamiques dans l'entrepôt de données (Derrar *et al.*, 2013). Les travaux présentés par (Ziyati, 2010) et (Derrar *et al.*, 2013) montrent l'importance de la fragmentation dans le domaine des ED. Cependant, les travaux réalisés se basent communément sur une liste statique et non évolutive des prédicats.

Par ailleurs, nous considérons la méthode de classification par l'algorithme k-means comme étant une méthode originale pour la FH. Elle permet de contrôler à l'avance le nombre de fragments et d'intégrer les caractéristiques de la base et de l'utilisation des données sous un format quantitatif dans des matrices simples à utiliser. Nous présentons dans ce qui suit, deux travaux utilisant la classification comme technique de fragmentation.

Mahboubi (Mahboubi, 2008) a proposé une solution pour la FH des ED XML afin de les répartir sur une grille. Il a exploité l'algorithme k-means. Sa solution englobe trois étapes : 1) codage des prédicats de sélection d'un ensemble de requêtes dans une matrice binaire qui représente le contexte de classification ; 2) classification des prédicats par application de la technique des k-means qui permet de partitionner l'ensemble des prédicats en k classes disjointes et enfin, 3) construction des fragments. La solution proposée permet de contrôler le nombre de partitions en fixant la valeur de k, mais ne tient pas compte de l'évolution du nombre de prédicats et de son impact sur l'efficacité de l'ED.

Dans le même contexte, Tekaya *et al.* (Tekaya *et al.*, 2010) ont défini deux nouvelles notions : la corrélation sémantique et la corrélation géographique. La corrélation sémantique permet de fusionner par conjonction de deux prédicats inclus dans la même clause WHERE. Une corrélation géographique est la fusion de deux prédicats n'appartenant pas à la même requête mais utilisés par la même localisation géographique. Une localisation géographique désigne le (ou les) site(s) sur lesquels sont alloués les magasins de données (MD). La solution proposée intègre l'aspect

géographique dans le processus de fragmentation. Elle englobe quatre phases : 1) détermination d'une liste de prédicats simples, 2) création de la matrice corrélation, 3) application de l'algorithme k-means pour la classification des prédicats et 4) génération des fragments horizontaux. Le travail proposé par (Tekaya *et al.*, 2010) tient compte de la réduction du nombre de prédicats par une simple application de l'algorithme COMM_MIN (Ozsu et Valduriez, 1999). Cependant, aucune technique spécifique n'a été proposée.

La diversité des solutions proposées pour la fragmentation horizontale des ED montre son importance. Cependant, l'efficacité de ces solutions risque de diminuer faisant face au problème d'augmentation du nombre de prédicats. A nos connaissances, ce problème n'a pas encore été considéré dans l'état de l'art.

3. Solution proposée

Dans notre travail, nous nous sommes concentrés sur la liste initiale des prédicats. Cette liste constitue un point d'entrée commun aux solutions de répartition les plus connues dans l'état de l'art. Nous considérons son augmentation comme étant un problème important qu'il faut traiter avant de procéder à la fragmentation des données. En effet, les prédicats sont extraits à partir des requêtes les plus fréquentes dans le SID. Pour minimiser leur nombre, il faut procéder par calcul de pertinence. Dans ce qui suit, nous proposons une solution pour la réduction du nombre de prédicats pour les approches de fragmentation des ED. Son but est de minimiser le nombre de prédicats pour répartir les données.

La solution proposée se déroule en quatre phases :

– phase de sélection préliminaire des prédicats ;
– phase de codification ;
– phase de classification ;
– phase de réduction du nombre de prédicats.

3.1. Phase de sélection préliminaire des prédicats

Le choix des prédicats les plus pertinents est une phase que nous considérons primordiale dans tout processus de répartition. Pour identifier les prédicats, nous utilisons, tout d'abord, l'algorithme COMM_MIN de (Ozsu et Valduriez, 1999).

Cet algorithme permet, à partir d'une liste de prédicats associés à une table de dimension, de générer une liste minimale et complète de prédicats pour cette table en respectant la règle (règle R) suivante :

Règle R – *Une relation ou bien un fragment est partitionné en au moins deux parties qui sont accédées différemment par au moins une application.*

Algorithme 1. Algorithme COMM_MIN

```
1:  generateurRepresentationEtatsActions (Entrée : P : liste des prédicats, d : une table de
2:  dimension      Sortie : P' : Liste de prédicats complète et minimale, i=entier, k=entier)
3:  {
4:      Trouver p_i ∈ P qui partitionne d selon la règle R
5:      P'= P_i ; P← (P – p_i) ; F← f_i
6:      Répéter
7:          Trouver p_j ∈ P de telle sorte que p_j partitionne certains f_k selon la règle R
8:              P'← P' ∪ p_i ; P← (P – p_i) ; F← F ∪ f_i
9:              Si ∃p_k ∈ P' qui n'est pas pertinent Alors
10:                 P'← P' – p_k
11 :                F← F – f_k
11:             Fin si
12:     Jusqu'à ce que P' soit complet
13 :  Retourner P'
14:  }
```

L'algorithme COMM_MIN est considéré aujourd'hui comme étant une référence à toute technique de fragmentation dans les bases de données relationnelles. En effet, il garantit les règles de reconstruction, complétude et disjonction des fragments de tables comme suit : la reconstruction des tables initiales peut se faire par l'opérateur algébrique d'union (∪). Une telle liste de prédicats assure une fragmentation sans perte d'information (d'où la complétude des données). Chaque prédicat figure dans un seul fragment ce qui assure la disjonction des fragments.

3.2. Phase de codification

Dans cette deuxième phase, nous nous sommes inspirés du travail réalisé par (Mahboubi, 2008) pour la conversion des prédicats en codes binaires. (Mahboubi, 2008) a utilisé une matrice d'utilisation des prédicats (MUP) qui englobe les utilisations des prédicats par les requêtes. Dans la figure 1, les colonnes de la MUP englobent les prédicats les plus utilisés. Les lignes désignent les requêtes OLAP les plus fréquentes sur le système. Une cellule de la MUP englobe la valeur 1 si une requête donnée utilise un prédicat donné et la valeur 0, sinon.

Dans notre travail, nous adaptons cette matrice comme suit. Pour un nombre de prédicats allant de 1 à u et un nombre de sites allant de 1 à k. Les lignes de la MUP contiennent les prédicats utilisés. Les colonnes de la MUP englobent les sites géographiques de l'entreprise. Une cellule de la MUP englobe la valeur 1 si un prédicat donné est utilisé par un site donné et la valeur 0 sinon. Par rapport au travail de (Mahboubi, 2008), nous remplaçons les requêtes par les prédicats et les prédicats par les sites de l'entreprise. Nous gardons par contre son principe de codification. Dans la figure 2, nous présentons un exemple de MUP.

	P_1	P_2	…	P_m
R_1	1	1	0	0
R_2	1	0	0	1
..				
R_u	0	1	1	0

Figure 1. MUP (Mahboubi, 2008)

	S_1	S_2	...	S_k
P_1	1	1	0	0
P_2	1	0	0	1
.				
P_u	0	1	1	0

Figure 2. MUP (Ghorbel et al.*, 2014)*

La phase de codification permet de produire une représentation binaire des prédicats selon leurs utilisations dans les k sites de l'entreprise. La MUP constitue notre point d'entrée à la phase de classification des prédicats.

3.3. Phase de classification

Dans cette troisième phase, nous utilisons la MUP comme entrée de la phase de classification (dans la partie pratique). Elle englobe une description binaire de tous les prédicats utilisés dans leurs sites.

Pour la classification des prédicats des tables de dimension, nous utilisons l'algorithme des k-means (Tekaya, 2011). Le principe de base de cet algorithme assigne chaque objet à un sous-ensemble dont le centre est le plus proche de l'objet en question. L'algorithme 2 procède comme suit :

Algorithme 2. Algorithme des k_means

```
1:  generateurRepresentationEtatsActions (Entrée : MUP, k :entier, C = C1,…,Ck :Une
2:  première partition          Sortie : C' = C1,…,Ck )
3:  {
4:      Répéter
5:      Affectation : générer une nouvelle partition C' en assignant chaque objet au groupe
6:      dont le centre de gravité est le plus proche.
7:      Représentation : calculer les centres de gravité associés à la partition.
8 :     Jusqu'à convergence de l'algorithme vers une partition stable.
9 :   Retourner C'
10:  }
```

Nous commençons par choisir le nombre des sous-ensembles k à obtenir. Ensuite, k groupes sont choisis de manière aléatoire et les centres sont déterminés. Nous pouvons aussi prendre aléatoirement k objets comme étant les centres initiaux. Après, chaque objet est assigné au groupe dont le centre est le plus proche et les centres de chaque groupe sont recalculés. On passe à la phase suivante lorsque la composition des groupes devient stable. Le calcul se fait en fonction de la distance inter-classes.

Nous utilisons l'algorithme k-means (Tekaya, 2011) pour la classification des prédicats des tables de dimensions contenus dans la MUP. C'est un algorithme de partitionnement de données relevant des statistiques et de l'apprentissage automatique. K-means est une méthode dont le but est de diviser des observations en k partitions dans lesquelles chaque observation appartient à la partition avec la moyenne la plus proche. Nous avons choisi d'utiliser cet algorithme car il est utilisé dans des contextes similaires par des travaux récents (Mahboubi, 2008 ; Tekaya, 2011) d'autant plus qu'il permet de superviser à l'avance (par l'initialisation de k) le nombre de classes générées et en conséquence le nombre de fragments générés.

Dans la figure 3, nous présentons un exemple de classification en utilisant la technique des k-means. Nous utilisons comme formalisme de présentation des résultats obtenus une matrice de classification **(MC)**. Chaque colonne correspond à une classe (ou dans notre cas, un site *S* donné). Pour un nombre de classes allant de *1* à *k* et pour un nombre prédicats allant de *1* à *u.* Les prédicats seront placés par site selon les distances inter-classes de l'algorithme des k-means.

$$\begin{pmatrix} C_1 & C_2 & \ldots & C_k \\ & & & \\ P_1 & P_3 & P_4 & P_5 \\ P_2 & S_2 & S_3 & P_6 \\ S_1 & & & S_4 \end{pmatrix}$$

Figure 3. Exemple de MC

La MC constitue une base de travail pour le processus de fragmentation et d'allocation des MD dans les différents sites de l'entreprise. Dans la section qui suit, nous présentons une approche pour le contrôle et la réduction du nombre de prédicats.

3.4. Phase de réduction du nombre de prédicats

Pour la réduction du nombre de prédicats, nous procédons en trois étapes (Ghorbel *et al.*, 2014) :

- application de l'algorithme d'optimisation du nombre de prédicats ;
- réduction des prédicats par corrélation sémantique ;

– réduction des prédicats par corrélation géographique ;

3.4.1. Réduction par l'algorithme d'optimisation du nombre des prédicats

Nous appliquons l'algorithme de réduction du nombre des prédicats. L'algorithme 3 ci-dessous, détaille cette quatrième phase. Le but de cet algorithme est d'éliminer les prédicats appartenant à tous les sites de l'entreprise. Il est inutile de fragmenter l'ED par un prédicat utilisé par tous les sites.

Algorithme 3. Algorithme d'optimisation du nombre de prédicats

```
1:   generateurRepresentationEtatsActions (Entrée : P :liste des prédicats,
2:    Sortie : P' : Liste de prédicats plus minimale, nb=entier, i=entier, j=entier)
3:   {
4:      P' ← P
5:      F ← chaque prédicat (Pi) correspond à 1fragment ( Fpi)
6:      Pour tout ( i=0 ; i < nb ; i ++ ) Faire
7:         Si (Pi ∈ P) s'illustre dans tous les sites crées Alors
8:            P← (P – Pi)
9:            F← (F – Fpi)
10:           P'← (P – Pi)
11:        Fin si
12:     Fin pour
13 :  Retourner P'
14:  }
```

En effet, la variable P d'entrée en ligne 1 de l'algorithme 3 reçoit une liste de prédicats, la variable F en ligne 5 initialisée par l'ensemble des fragments des prédicats P et au fur et à mesure reçoit ou élimine les fragments des prédicats ajoutés ou supprimés et la variable P' de sortie située en ligne 2 reçoit la liste finale des prédicats optimisée.

En premier lieu, nous éliminons les prédicats appartenant à tous les sites. L'algorithme 3 procède par traitement itératif en ligne 6 qui nous permet de parcourir tous les prédicats en entrée et l'instruction de condition *Si* en ligne 7 vérifie si le prédicat appartient à tous les sites ou non. S'il appartient à tous les sites, on l'élimine. Sinon, il reste dans la liste. Cela permet de minimiser le nombre de prédicats sans toucher les données et en conséquence, nous minimisons le nombre de fragments générés.

Nous avons implémenté cet algorithme de complexité O(n) qui nous permet d'éliminer les prédicats appartenant à tous les sites donnés. L'accès à ces données peut se faire de n'importe quel site et ces prédicats n'entrent pas dans le processus de fragmentation de l'ED. Le fragment répond au besoin des utilisateurs du site donné et il reste spécifique à ces utilisateurs.

Prenons l'exemple d'un ED pour la *gestion des ventes* et les deux requêtes (R_1) et (R_2) suivantes:

(R_1)	(R_2)
SELECT SUM(Montant)	SELECT *
FROM Vente	FROM Vente
WHERE id_ville = 'Tunis'	WHERE id_ville = 'Tunis'

(R_1) et (R_2) sont exécutées sur deux sites différents. Il est inutile de fragmenter l'ED selon *id_ville = 'Tunis'* car ce prédicat est utilisé par tous les sites. Ainsi, nous minimisons le nombre de prédicats et par conséquent le nombre de fragments. Cette réduction du prédicat n'a pas d'impact sur les besoin des utilisateurs puisque les données restent dans les deux sites.

3.4.2. Réduction des prédicats selon les corrélations sémantiques

Nous avons utilisé la notion de corrélation sémantique proposée par (Tekaya, 2011) pour réduire le nombre de prédicats généré par l'algorithme d'optimisation du nombre des prédicats. De même, nous proposons de regrouper par conjonction les prédicats ayant une interdépendance sémantique, c'est à dire utilisés dans la même clause WHERE.

Prenons l'exemple d'un ED pour la *gestion des ventes* et les deux requêtes (R_1) et (R_2) suivantes:

(R_1)	(R_2)
SELECT SUM(Montant)	SELECT SUM(Montant)
FROM Vente	FROM Vente
WHERE id_produit = '100'	WHERE id_produit = '200'
AND id_temps = 'Mai'	AND id_temps = 'Juin'

(R_1) et (R_2) sont utilisées par une *même localisation géographique* S_1 sur une table de faits *Vente* ayant le schéma relationnel suivant :

Vente (<u>id_produit, id_client, id_temps, id_ville</u>, Montant)

Ci-dessous, nous présentons la liste des prédicats extraits à partir de (R_1) et (R_2):

P_1 : id_produit = '100'
P_2 : id_temps = 'Mai'
P_3 : id_produit = '200'
P_4 : id_temps = 'Juin'

Les prédicats P_1 *et* P_2 ont une corrélation sémantique parce qu'ils sont utilisés par la même requête R_1. De même, les prédicats P_3 et P_4 ont une corrélation sémantique parce qu'ils sont utilisés par la même requête R_2. P_1 et P_2 seront

fusionnés en P_1' (P_1^P_2) par conjonction. P_3 et P_4 seront fusionnés en P_2' (P_3^P_4) par conjonction de P_3 et P_4.

3.4.3. Réduction des prédicats selon les corrélations géographiques

Nous avons utilisé la notion de corrélation géographique proposée par (Tekaya, 2011) pour réduire le nombre de prédicats généré par l'algorithme d'optimisation du nombre des prédicats. Nous proposons de regrouper par conjonction les prédicats qui ont une interdépendance géographique dans un même prédicat. Ceci permet de regrouper les prédicats qui sont utilisés par les mêmes sites.

Reprenons l'exemple de l'ED pour la *gestion des ventes* précédent. Les prédicats P_1' et P_2' ont une corrélation géographique parce qu'ils sont utilisés par deux requêtes appartenant à une même localisation géographique. P_1' et P_2' seront fusionnés par conjonction en P_1'' (P_1'^P_2').

La solution proposée nous permet d'éliminer les prédicats appartenant à tous les sites de l'entreprise. Dans chaque classe, nous procédons à la réduction du nombre de prédicats selon les corrélations sémantiques et/ou géographiques à fin de réduire le nombre de fragments générés. Dans la section qui suit, nous présentons un exemple détaillé de notre solution, ainsi que son application sur un ED réel issu du banc d'essai APB1.

4. Validation expérimentale

Dans cette section nous présentons une concrétisation de notre solution. D'abord, nous présentons l'environnement expérimental du travail. Ensuite, nous proposons un exemple d'application détaillé des différentes étapes de l'approche proposée. Enfin, nous exposons les résultats obtenus après son application sur un ED.

4.1. Présentation de l'environnement expérimental

Pour valider notre travail, nous avons utilisé un ED réel issu du *benchmark* APB1 (Spofford, 1998). Ce *benchmark* demeure le plus utilisé par les travaux abordant le problème de la fragmentation des ED que nous considérons similaires à notre contexte de travail. Nous citons les laboratoires de recherches comme ENSMA (Boukhalfa, 2009), ERIC (Darmont, 2006). Le choix d'APB1 nous permettra de comparer nos résultats à ces travaux aussi bien au niveau technique que pratique de la solution.

Sur cet ED, nous avons exécuté un ensemble de 19 requêtes réparties sur deux sites géographiquement distants. De point de vue matériel, nous avons utilisé deux machines ayant les caractéristiques suivantes:

– machine 1 : Intel(R), Core(TM) 2 Duo, CPU 2GHz, 1,99 Go de RAM,

– machine 2 : réduction Intel(R), Core(TM) i3, CPU 2GHz, 4 Go de RAM,

Pour la répartition de l'ED sur ces 2 machines, nous avons utilisé un réseau privé virtuel (VPN) (Pham, 2002). Un VPN repose sur un protocole, appelé « protocole de tunnélisation » qui permet aux données passant d'une extrémité à l'autre du VPN d'être sécurisées par des algorithmes de cryptographie. Pour la génération des tables du banc d'essai APB1, nous avons installé Oracle 10g sur la machine 1 et Oracle 11g sur la machine 2 que nous avons trouvé les plus adaptées aux capacités des machines.

L'ED est constitué de 5 tables : 1 table de faits Actvars et 4 tables de dimensions : Prodlevel, Custlevel, Timelevel et Chanlevel (Spofford, 1998).

Le tableau 1 résume les caractéristiques de chaque table.

Tableau 1. Caractéristiques des tables de l'entrepôt de données

Table	Nombre d'enregistrement	Taille d'un enregistrement
Actvars	24 786 000	74
Chanlevel	9	24
Custlevel	900	24
Prodlevel	9000	72
Timelevel	24	36

Dans cette expérimentation, nous avons initialisé k à la valeur 3. Le choix de k peut se faire en fonction du nombre de sites de l'entreprise. Dans notre cas, par exemple k = 2 ou bien en fonction du nombre de fragments (ou MD) que l'entreprise souhaite créer et répartir. Dans cet exemple, nous avons choisi de générer trois fragments (k = 3) que nous allons répartir sur deux sites distants.

4.2. Exemple d'application sur le benchmark APB1

A partir de l'ensemble des requêtes présenté en annexe, nous avons collecté 18 prédicats de sélection sur les tables de dimension. Sur ces prédicats, nous avons appliqué les 4 phases de notre solution.

Ci-après, la liste des prédicats choisie selon le principe du COMM_MIN (phase 1).

P_1 : id_mois = ‘Novembre’ P_2 : id_mois = ‘Décembre’
P_3 : id_mois = ‘Janvier’ P_4 : id_ville = ‘Tunis’
P_5 : id_ville = ‘Sfax’ P_6 : id_ville = ‘Bizerte’
P_7 : id_client = ‘1’ P_8 : id_client = ‘2’
P_9 : id_client = ‘3’ P_{10} : id_produit = ‘Chaussure’
P_{11} : id_mois = ‘Février’ P_{12} : id_mois = ‘Mars’
P_{13} : id_mois = ‘Avril’ P_{14} : id_mois = ‘Mai’
P_{15} : id_mois = ‘Juin’ P_{16} : id_mois = ‘Juillet’
P_{17} : id_mois = ‘Août’ P_{18} : id_produit = ‘Veste’

Après la phase de sélection préliminaire des prédicats, nous abordons la phase de codification des prédicats.

Dans cette phase, nous commençons par la conversion en code binaire des différentes utilisations des prédicats. Les colonnes de la MUP englobent les sites géographiques de l’entreprise. Les lignes de la MUP contiennent les prédicats utilisés. Les colonnes de la MUP englobent les sites géographiques de l’entreprise. Une cellule de la MUP contient la valeur 1 si un prédicat donné est utilisé par un site donné et la valeur 0 sinon. Nous proposons dans la figure 4, comme exemple, pour les 3 sites, la MUP suivante:

Après la phase de codification des prédicats, nous passons à la classification des prédicats.

Dans cette phase, nous exécutons l’algorithme des k-means *via* l’infrastructure expérimentale TANAGRA (Rakotomalala, 2004) sur la MUP que nous avons proposé comme exemple dans la phase 2. Chaque colonne correspond à une classe. Chaque classe allant de 1 à k correspond à un fragment donné qui contient les besoins des utilisateurs. Les prédicats seront placés par site selon l’algorithme des k-means. Dans la figure 5, nous présentons le résultat obtenu pour k = 3 :

Les classes générées sont :

$C_1 : P_1 \wedge P_2 \wedge P_3 \wedge P_4 \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8 \wedge P_9 \wedge P_{10} \wedge P_{17} \wedge P_{18} \wedge S_1$

$C_2 : P_{11} \wedge P_{12} \wedge P_{13} \wedge P_{10} \wedge P_{17} \wedge P_{18} \wedge S_2$

$C_3 : P_{14} \wedge P_{15} \wedge P_{16} \wedge P_{10} \wedge P_{17} \wedge P_{18} \wedge S_3$

Nous terminons par la dernière phase de réduction du nombre de prédicats.

Dans cette étape, nous appliquons tout d’abord, l’algorithme de réduction du nombre de prédicats. Ensuite, nous procédons à la réduction des prédicats par la fusion selon les interdépendances sémantiques et/ou géographiques.

La liste des prédicats devient :

P'_1 : id_mois in (‘Novembre’,’Décembre’,’Janvier’)

P'_2 : id_ville in (‘Tunis’,’Sfax’,’Bizerte’)

P'_3 : id_client in (‘1’,’2’,’3’)

P'_4 : id_mois in (‘Février’,’Mars’,’Avril’)

P'_5 : id_mois in (‘Mai’,’Juin’,’Juillet’)

	S_1	S_2	S_3
P_1	1	0	0
P_2	1	0	0
P_3	1	0	0
P_4	1	0	0
P_5	1	0	0
P_6	1	0	0
P_7	1	0	0
P_8	1	0	0
P_9	1	0	0
P_{10}	1	1	1
P_{11}	0	1	0
P_{12}	0	1	0
P_{13}	0	1	0
P_{14}	0	0	1
P_{15}	0	0	1
P_{16}	0	0	1
P_{17}	1	1	1
P_{18}	1	1	1

Figure 4. MUP

C_1	C_2	C_3
P_1	P_{11}	P_{14}
P_2	P_{12}	P_{15}
P_3	P_{13}	P_{16}
P_4	P_{10}	P_{10}
P_5	P_{17}	P_{17}
P_6	P_{18}	P_{18}
P_7	S_2	S_3
P_8		
P_9		
P_{10}		
P_{17}		
P_{18}		
S_1		

Figure 5. MC

En effet, les prédicats P_{10}, P_{17} et P_{18} ont été éliminés par l’algorithme d’optimisation du nombre de prédicats car ils appartiennent à tous les sites de l’entreprise. De plus, le prédicat P'_1 a remplacé les prédicats P_1, P_2 et P_3, le prédicat P'_2 a remplacé les prédicats P_4, P_5 et P_6, le prédicat P'_3 a remplacé les prédicats P_7, P_8 et P_9, le prédicat P'_4 a remplacé les prédicats P_{11}, P_{12} et P_{13} et le prédicat P'_5 a remplacé les prédicats P_{14}, P_{15} et P_{16} par l’utilisation des interdépendances géographiques. Les classes générées après réduction des prédicats sont :

$C_1 : P'_1 \wedge P'_2 \wedge P'_3 \wedge S_1$

$C_2 : P'_4 \wedge S_2$

$C_3 : P'_5 \wedge S_3$

En appliquant notre solution de réduction des prédicats, nous avons pu diminuer la liste de 18 prédicats à 5 prédicats pour k=3 fragments.

4.3. Interprétation des résultats obtenus

Dans notre exemple, la dernière phase de classification a engendré 3 classes de prédicats. Nous avons utilisé la conjonction de ces prédicats par classe pour le partitionnement de la table de faits sur trois fragments selon la solution de (Tekaya *et al*., 2010). Les fragments engendrés ont été par la suite alloués aléatoirement sur les deux machines distantes. Pour la validation de notre solution, nous avons commencé par mesurer les temps d'exécution des requêtes dans le cas d'un ED centralisé, ensuite, puis partitionné sans réduction des prédicats, et enfin réparti en appliquant notre solution de réduction des prédicats. Les résultats obtenus sont récapitulés dans le tableau 2.

Tableau 2. Temps d'exécution des requêtes utilisées en (mn : s : ms)

Numéro de requête	Temps d'exécution centralisé	Temps d'exécution partitionné	Temps d'exécution réparti
1	1 : 58 : 34	0 : 48 : 12	0 : 22 : 10
2	1 : 58 : 01	0 : 54 : 54	0 : 31 : 43
3	5 : 31 : 88	2 : 13 : 18	5 : 04 : 16
4	5 : 31 : 08	4 : 31 : 24	8 : 34 : 45
5	7 : 10 : 25	3 : 42 : 85	1 : 58 : 70
6	9 : 14 : 33	7 : 20 : 11	4 : 53 : 62
7	10 : 13 : 14	9 : 49 : 62	7 : 18 : 78
8	14 : 05 : 13	13 : 03 : 91	10 : 22 : 83
9	15 : 01 : 03	10 : 48 : 65	8 : 59 : 17
10	16 : 14 : 78	13 : 22 : 14	10 : 33 : 45
11	7 : 19 : 11	5 : 32 : 92	4 : 44 : 89
12	7 : 28 : 17	4 : 55 : 69	3 : 52 : 51
13	8 : 17 : 21	6 : 24 : 28	5 : 36 : 77
14	9 : 45 : 02	8 : 57 : 95	7 : 51 : 09
15	6 : 17 : 20	5 : 39 : 78	4 : 34 : 30
16	7 : 45 : 34	5 : 52 : 74	4 : 01 : 09
17	9 : 11 : 97	8 : 26 : 85	7 : 11 : 51
18	8 : 48 : 68	7 : 10 : 35	5 : 46 : 03
19	10 : 53 : 93	8 : 14 : 47	7 : 27 : 42

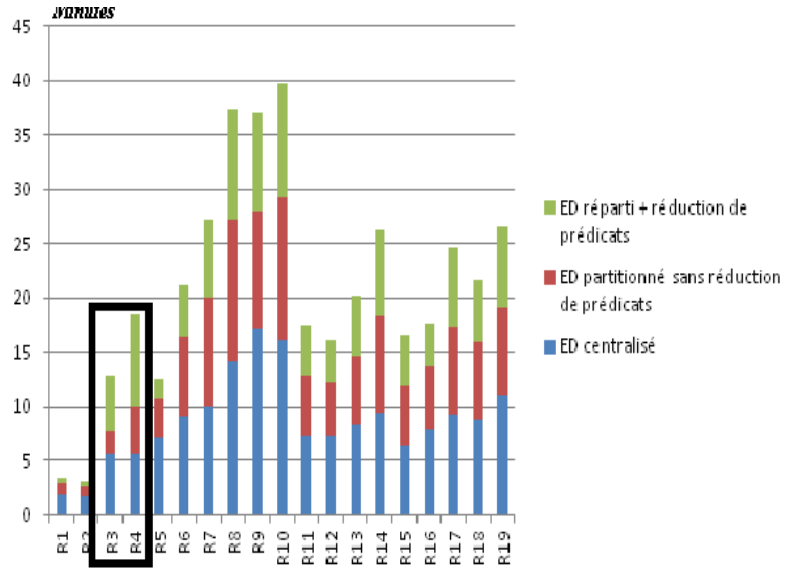

Figure 6. Temps d'exécution des requêtes.

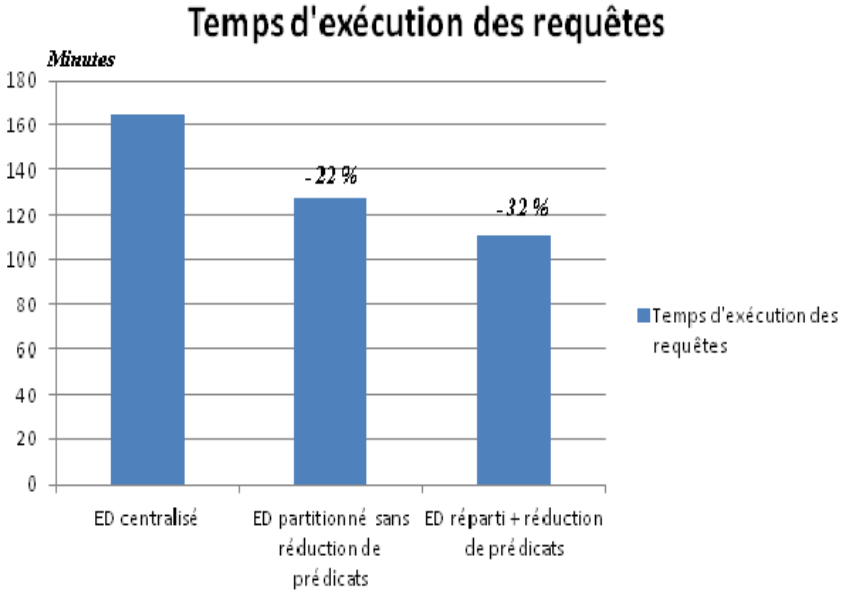

Figure 7. Comparaison des temps d'exécution globaux

Pour les requêtes numéro 1, 2, 5 jusqu'à 19, les temps d'exécution ont diminué dans la figure 6, ce qui constitue pour nous un gain non négligeable. Le temps d'exécution global des requêtes dans un contexte réparti avec réduction des prédicats a diminué de 32 % par rapport au contexte centralisé et de 10 % par rapport au contexte partitionné sans réduction du nombre des prédicats dans la figure 7. Par contre, pour les requêtes 3 et 4, les temps d'exécution ont augmenté dans la figure 6. Ce sont les requêtes les moins bénéficiaires du partitionnement. Elles n'ont pas été exploitées par le partitionnement des tables. Ceci explique clairement l'augmentation remarquable des mesures obtenues. Dans ce contexte, quelques solutions sont envisageables pour la réécriture des requêtes OLAP distantes pour l'optimisation du temps d'exécution notamment les travaux de Liang et Yu (2000) et/ou de Kalnis et Papadias (2001) où ils rajoutent une heuristique d'optimisation des requêtes.

5. Conclusion

Dans cet article, nous nous sommes intéressés à l'optimisation des requêtes décisionnelles exécutées sur un ED modélisé en étoile. Nous avons proposé une solution pour la réduction du nombre de prédicats pour les approches de répartition des ED. Cette solution repose sur quatre phases : une phase de sélection préliminaire des prédicats, une phase de codification, une phase de classification et une phase de réduction des prédicats.

La première phase se base sur l'algorithme COMM-MIN pour générer une liste minimale et complète de prédicats pour chaque table de dimension. La phase de codification consiste à produire une représentation binaire des prédicats selon leur utilisation dans les différents sites. La troisième phase utilise l'algorithme k-means pour obtenir une matrice de classification des prédicats par site. La dernière phase (réduction du nombre de prédicats) se base une corrélation sémantique et une corrélation géographique.

Pour l'évaluation de la solution proposée, nous l'avons appliquée sur un ED réel issu du banc d'essai APB1 que nous avons réparti selon notre démarche de fragmentation en utilisant les différentes requêtes proposées comme exemple d'application. Les résultats obtenus sont motivants et garantissent une utilisation plus adéquate et plus souple des données au sein de l'entreprise.

A l'issue de ce travail, nous estimons que quelques axes de recherches restent à étudier et à approfondir. Le premier est relatif à l'allocation (ou répartition) des données de l'ED en un ensemble de magasins de données (MD). Le problème de l'allocation des données dans un contexte d'ED doit tenir compte des contraintes de répartition notamment la contrainte d'accès à un MD à partir des sites distants, le stockage des données, le délai de réponse et la fréquence d'utilisation de chaque MD par les différents sites de l'entreprise. Il faut aussi tenir compte de la contrainte de chargement d'un MD dans tous les sites.

Dans la plupart des cas, la répartition d'un ED est fondée sur des critères de fragmentation (attributs, prédicats de sélection, affinité, etc.) et/ou des critères de réplication (fréquence d'utilisation, coûts d'accès, coûts de stockage, etc.). Ces critères évoluent selon les besoins des utilisateurs. Pour faire face aux changements, une mise à jour périodique du schéma de répartition est nécessaire. Une automatisation de cette mise permettrait probablement d'améliorer l'utilisation de l'ED.

Bibliographie

Barr M. (2010). *Approche dirigée par les fourmis pour la fragmentation horizontale des entrepôts de données relationnels*. Mémoire de mastère, École nationale Supérieure d'Informatique, Algérie.

Bellatreche L. (2008). Bitmap join indexes and data partitioning. *Encyclopedia of Data Warehousing and Mining 2nd Edition*, 5, p. 37-38.

Bellatreche L., Boukhalfa K., Richard P. (2011). Primary and referential horizontal partitioning selection problems. *Concepts, Algorithms and Advisor Tool.*

Bellatreche L., Karlapalem K. et Li Q. (2004). Derived horizontal class partitioning in oodb: Design strategies, analytical model and evaluation. *Conceptual Modeling ER'98*, p. 465-479.

Boukhalfa K. (2009). *De la conception physique aux outils d'administration et de tuning des entrepôts de données*. Thèse de doctorat, Université de Poitiers.

Darmont J. (2006). *Optimisation et évaluation de performance pour l'aide à la conception et à l'administration des entrepôts de données complexes*. Thèse de doctorat, Université Lumière Lyon 2.

Derrar H., Ahmed-Nacer M., Boussaid O. (2008). Une approche de répartition des données d'un entrepôt basée sur l'Optimisation par Essaim Particulaire. *In EDA 2008*, p. 141-150.

Derrar H., Ahmed-Nacer M., Boussaid O. (2012). Particle swarm optimisation for data warehouse logical design. *International Journal of Bio-Inspired Computation.* vol. 4, n° 4, July, p. 249-257.

Derrar H., Ahmed-Nacer M., Boussaid O. (2013). Exploiting data access for dynamic fragmentation in data warehouse. *International Journal of Intelligent Information and Database Systems* 01/2013, vol.7, n° 1, p. 34-52. DOI: 10.1504/IJIIDS.2013.051736.

Ghorbel M., Tekaya K., Abdellatif A. (2014). Réduction du nombre des prédicats pour les approches de répartition des entrepôts de données. *ASD 2014*, 29-31 mai 2014, Hammamet, Tunisie.

Kalnis P., Papadias D. (2001). Optimization algorithms for simultaneous multidimensional queries in olap environments. *Data Warehousing and Knowledge Discovery*, p. 264-273.

Liang W., Orlowska M., Yu J. (2000). Optimizing multiple dimensional queries simultaneously in multidimensional databases. *The VLDB Journal The International Journal on Very Large Data Bases* 8, p. 319-338.

Mahboubi H. (2008). *Optimisation de la performance des entrepôts de données XML par fragmentation et répartition*. Thèse de doctorat, Université Lumière Lyon 2.

Ozsu T. et Valduriez P. (1999). Principes of distributed database systems. *Prentice Hall*, 19-22.

Pham C. (2002). *VPN et solutions pour l'entreprise*. SaaS, Université de Pau et des Pays de l'Adour.

Rakotomalala R. (2004). *Tanagra : une plate-forme d'expérimentation pour la fouille de données. Open Access Journal*, Université Lumière Lyon 2.

Spofford G. (1998). *OLAP conseil APB-1 benchmark. Guide d'installation.*

Tekaya K., Abdellatif A., Ounalli H. (2010). Data mining based fragmentation technique for distributed data warehouses environment using predicate construction technique. *In Sixth International Conference on Networked Computing and Advanced Information Management (NCM)*, p. 63-68.

Tekaya K. (2011). *Fragmentation et allocation dynamiques des entrepôts de données*. Thèse de doctorat, Faculté des sciences de Tunis.

Zhang Y., Orlowska M. (1995). Fragmentation approaches for distributed database design. *Information Sciences-Applications*, p. 117-132.

Ziyati E. (2010). *Optimisation de requêtes OLAP en Entrepôts de Données : Approche basée sur la fragmentation génétique*. Thèse de doctorat, Faculté des Sciences Rabat.

Annexe

1. Création des tables de l'entrepôt de données APB1 centralisé

```
Create table CHANLEVEL
(BASE_LEVEL CHAR (12) not null, ALL_LEVEL CHAR (12)) ;
Alter table CHANLEVEL Add primary key (BASE_LEVEL) ;

Create table CUSTLEVEL
(STORE_LEVEL CHAR (12) not null, RETAILER_LEVEL CHAR (12) not null) ;

Alter table CUSTLEVEL Add primary key (STORE_LEVEL) ;

Create table PRODLEVEL
(CODE_LEVEL CHAR (12) not null, CLASS_LEVEL CHAR (12) not null,
GROUP_LEVEL CHAR (12) not null, FAMILY_LEVEL CHAR (12) not null,
LINE_LEVEL CHAR (12) not null, DIVISION_LEVEL CHAR (12) not null) ;

Alter table PRODLEVEL Add primary key (CODE_LEVEL) ;

Create table TIMELEVEL
(TID VARCHAR2 (12) not null, YEAR_LEVEL NUMBER (4) not null,
QUARTER_LEVEL VARCHAR2 (6) not null, MONTH_LEVEL NUMBER (2),
WEEK_LEVEL NUMBER (2), DAY_LEVEL NUMBER (2)) ;

Alter table TIMELEVEL Add primary key (TID) ;
```

```
Create table ACTVARS
(CUSTOMER_LEVEL CHAR (12) not null, PRODUCT_LEVEL CHAR (12) not
null, CHANNEL_LEVEL CHAR (12) not null, TIME_LEVEL VARCHAR2 (12)
not null, UNITSSOLD FLOAT not null, DOLLARSALES FLOAT not null,
DOLLARCOST FLOAT);

Alter table ACTVARS Add foreign key (CUSTOMER_LEVEL)
References CUSTLEVEL (STORE_LEVEL) ;

Alter table ACTVARS Add foreign key (PRODUCT_LEVEL)
References PRODLEVEL (CODE_LEVEL) ;

Alter table ACTVARS Add foreign key (CHANNEL_LEVEL)
References CHANLEVEL (BASE_LEVEL) ;

Alter table ACTVARS Add foreign key (TIME_LEVEL)
References TIMELEVEL (TID) ;
```

2. Quelques requêtes utilisées

```
/*R1 : Répartie, analyse des ventes par produit et par année */
select timelevel_site2.year_level,
prodlevel.code_level,
sum (a.unitssold) AS Units,
sum (a.dollarsales) AS Dollars,
sum (a.dollarsales) / sum (a.unitssold) AS Price
from actvars_site2@orcl_link11, prodlevel@orcl_link11,
timelevel_site2@orcl_link11,
custlevel_site2@orcl_link11, chanlevel_site2@orcl_link11
where a.product_level = prodlevel.code_level
and a.time_level = timelevel_site2.tid
and a.customer_level = custlevel_site2.store_level
and a.channel_level = chanlevel_site2.base_level
group by timelevel_site2.year_level,
prodlevel.code_level ;

/*R2 : Répartie, analyse des ventes par produit et par année */
select timelevel_site2.year_level,
prodlevel.code_level,
sum (a.unitssold) AS Units,
sum (a.dollarsales) AS Dollars,
sum (a.dollarsales) / sum (a.unitssold) AS Price
from actvars_site5@orcl_link11 , prodlevel@orcl_link11,
timelevel_site2@orcl_link11,
custlevel_site2@orcl_link11, chanlevel_site2@orcl_link11
where a.product_level = prodlevel.code_level
and a.time_level = timelevel_site2.tid
and a.customer_level = custlevel_site2.store_level
```

```
and a.channel_level = chanlevel_site2.base_level
group by timelevel_site2.year_level,
prodlevel.code_level ;

/*R3 : Répartie, analyse des ventes par customer et par produit */
select custlevel.store_level,prodlevel.code_level,
sum (a.unitssold) AS Units, sum (a.dollarsales) AS Dollars
from actvars_site3@orcl_link11 a, prodlevel@orcl_link11, timelevel@orcl_link11,
custlevel@orcl_link11, chanlevel_site2@orcl_link11
where a.product_level = prodlevel.code_level
and a.time_level = timelevel.tid
and a.customer_level = custlevel.store_level
and a.channel_level = chanlevel_site2.base_level
group by custlevel.store_level, prodlevel.code_level ;

/*R4 : Répartie, analyse des ventes par customer et par produit */
select custlevel.store_level,prodlevel.code_level,
sum (a.unitssold) AS Units, sum (a.dollarsales) AS Dollars
from actvars_site4@orcl_link11 a, prodlevel@orcl_link11, timelevel@orcl_link11,
custlevel@orcl_link11, chanlevel_site1@orcl_link11
where a.product_level = prodlevel.code_level
and a.time_level = timelevel.tid
and a.customer_level = custlevel.store_level
and a.channel_level = chanlevel_site1.base_level
group by custlevel.store_level, prodlevel.code_level ;

/*R5 : Répartie, analyse des ventes par mois, par client et par famille de produit
classées
par ordre croissant du chi_re d'a_aire */
select timelevel_site1.tid, prodlevel.family_level, custlevel_site1.store_level,
sum (a.unitssold) AS Units,sum (a.dollarsales) AS Dollars
from actvars_site1@orcl_link11 , prodlevel@orcl_link11,
timelevel_site1@orcl_link11,
custlevel_site1@orcl_link11, chanlevel_site1@orcl_link11
where a.product_level = prodlevel.code_level
and a.time_level = timelevel_site1.tid
and a.customer_level = custlevel_site1.store_level
and a.channel_level = chanlevel_site1.base_level
group by timelevel_site1.tid, prodlevel.family_level, custlevel_site1.store_level
ORDER BY dollars ;

/*R6 : Centralisée, analyse des ventes par produit par année */
select timelevel.year_level,prodlevel.code_level,
sum (a.unitssold) AS Units,
sum (a.dollarsales) AS Dollars,
sum (a.dollarsales) / sum (a.unitssold) AS Price
```

```
from actvars a, prodlevel, timelevel, custlevel, chanlevel
where a.product_level = prodlevel.code_level
and a.time_level = timelevel.tid
and a.customer_level = custlevel.store_level
and a.channel_level = chanlevel.base_level
and a.channel_level in (‘C701AZI9218J’)
group by timelevel.year_level,prodlevel.code_level ;

/*R7 : Centralisée, analyse des ventes par produit par année */
select timelevel.year_level,prodlevel.code_level,
sum (a.unitssold) AS Units,
sum (a.dollarsales) AS Dollars,
sum (a.dollarsales) / sum (a.unitssold) AS Price
from actvars a, prodlevel, timelevel, custlevel, chanlevel
where a.product_level = prodlevel.code_level
and a.time_level = timelevel.tid
and a.customer_level = custlevel.store_level
and a.channel_level = chanlevel.base_level
and a.channel_level in (‘KYABQYL6OSR8’)
group by timelevel.year_level,prodlevel.code_level ;

/*R8 : Centralisée, analyse des ventes par produit par année */
select timelevel.year_level,prodlevel.code_level,
sum (a.unitssold) AS Units,
sum (a.dollarsales) AS Dollars,
sum (a.dollarsales) / sum (a.unitssold) AS Price
from actvars a, prodlevel, timelevel, custlevel, chanlevel
where a.product_level = prodlevel.code_level
and a.time_level = timelevel.tid
and a.customer_level = custlevel.store_level
and a.channel_level = chanlevel.base_level
and a.channel_level in (‘TUZV6L7ILDXK’)
group by timelevel.year_level,prodlevel.code_level ;

/*R9 : Centralisée, analyse des ventes par produit par année */
select timelevel.year_level,prodlevel.code_level,
sum (a.unitssold) AS Units,
sum (a.dollarsales) AS Dollars,
sum (a.dollarsales) / sum (a.unitssold) AS Price
from actvars a, prodlevel, timelevel, custlevel, chanlevel
where a.product_level = prodlevel.code_level
and a.time_level = timelevel.tid
and a.customer_level = custlevel.store_level
and a.channel_level = chanlevel.base_level
and a.time_level in (‘199508’)
group by timelevel.year_level,prodlevel.code_level ;
```