Optimisation de requêtes par intervalles utilisant une indexation à deux niveaux par des UB-arbres

Samer Housseno, Ana Simonet, Michel Simonet
AGIM laboratory
Université de Grenoble
38700 La Tronche - France
{Samer.Housseno, Ana.Simonet, Michel.Simonet}@imag.fr

Résumé— Cet article traite le problème de l'optimisation des requêtes par intervalles, qui jouent un rôle important dans le domaine des bases de données multidimensionnelles (entrepôts de données, Systèmes d'information Géographique). Les techniques classiques teltes que les Grid Files, les R-trees, les R*-trees, … reposent sur une partition de l'espace de données selon plusieurs axes, ce qui permet le regroupement des données dans des clusters. Les courbes de remplissage de l'espace (Space Filling Curves) permettent de réaliser la transformation d'un n-uplet de données dans l'espace multidimensionel en une clé unique dans un espace unidimensionel. Une structure d'indexation monoattribut peut dès lors être utilisée pour indexer ces données. La structure d'indexation BUB-arbre est une structure d'indexation multidimensionnelle basée sur les B-arbres et sur la courbe Z de remplissage de l'espace.
Notre proposition consiste à indexer à la fois les clusters et les données de chaque clusters par des BUB-arbres. L'indexation des clusters permet de sélectionner ceux qui sont concernés par une requête, ce qui réduit le transfert de données entre le support de stockage et la mémoire centrale. Cette approche a été appliquée pour indexer les objets dans le système Osiris, qui est un système de représentation et de gestion des données et des connaissances qui offre de manière native une partition sémantique de données.

Mots-clés- Indexation multidimensionnelle, Courbe de remplissage de l'espace, UB-arbre, Optimisation de requêtes par intervalles.

I. INTRODUCTION

Le partitionnement de l'espace des données est une organisation physique qui améliore de façon significative les performances des requêtes par intervalles (range queries), requêtes qui jouent un rôle important dans les requêtes portant sur des données multidimensionnelles (Entrepôts de données, Systèmes d'Information Géographiques). Dans les systèmes actuels, les critères de partitionnement peuvent être fournis par le concepteur de la base de données, définis par des méthodes d'analyse basées sur la similarité des données, ou encore déduits des contraintes que les données doivent satisfaire. La première approche, connue sous le nom de «Cluster», est proposée par les SGBD comme Oracle [2] et PostgreSQL [3]. La deuxième approche est adoptée dans le système Osiris, un SGBD-BC de la famille des Logiques de Description [4] qui offre de manière native l'optimisation sémantique des requêtes et en particulier des requêtes par intervalles [5].

L'optimisation de ce type de requêtes nécessite l'indexation des données selon plusieurs attributs (ou dimensions). Ce besoin a contribué à l'émergence d'un certain nombre d'index multidimensionnels tels que les grid file, les R-tree, les R*-tree, dont les performances dépendent fortement du nombre d'attributs d'indexation considérés. Ainsi, pour un type d'index particulier, lorsque le nombre de dimensions dépasse un seuil, sa performance se dégrade. Par exemple, pour un R*-arbre, le temps de recherche double lorsque l'on passe de 6 à 12 dimensions [7]. C'est la "Malédiction de la dimensionnalité" (*Curse of dimensionality*). Une solution permettant de limiter l'impact du nombre d'attributs sur la performance d'un index consiste à transformer l'index multidimensionnel en un index unidimensionnel [8]. Les courbes de remplissage de l'espace (SFC : Space filling Curves) [9] sont des méthodes qui transforment une représentation multidimensionnelle des données en une représentation unidimensionnelle, ce qui permet l'utilisation d'index mono-attribut tels que les B-arbres, qui sont des méthodes simples et efficaces, dans le cadre de données multidimensionnelles.

La méthode des UB-arbres a retenu notre attention car elle a été reconnue comme particulièrement performante pour l'indexation des données multidimensionnelles [17]. Elle utilise la courbe Z de remplissage de l'espace, et les B-arbres pour l'indexation des données unidimensionnelles résultant de la transformation des données multidimensionnelles par la courbe Z [11].

Dans le cadre de notre travail sur le système Osiris [16], nous avons été amenés à implémenter une approche d'indexation à deux niveaux qui repose sur l'existence d'une partition du domaine des attributs qui se prolonge en une partition de l'espace des objets, appelée l'Espace de Classement. Les éléments de cette partition, appelés Eq-classes en Osiris, correspondent aux clusters dans les systèmes classiques. Afin de limiter le traitement des données en mémoire centrale, un prétraitement est réalisé sur les Eq-classes afin de ne charger en mémoire que les objets de celles « concernées » par une requête. Pour optimiser ce prétraitement, l'espace des Eq-classes, qui est également un espace multidimensionnel, est indexé par la méthode des UB-arbres. Les données de chaque Eq-classe sont elles-mêmes indexées par un UB-arbre, conduisant à autant d'UB-arbres qu'il y a d'Eq-classes actives (i.e., contenant au moins un objet). C'est la présentation de ce double niveau d'indexation de données multidimensionnelles par UB-arbres qui fait l'objet de cet article. La méthode présentée peut s'appliquer dans tout système permettant de définir une partition du domaine des attributs.

Dans cet article, nous rappelons les principes de l'indexation multidimensionnelle, nous présentons le système Osiris, contexte de notre travail, et enfin notre méthode et son évaluation.

II. INDEXATION MULTIDIMENSIONNELLE

La transformation de données multidimensionnelles au moyen des courbes de remplissage de l'espace (Space Filling Curve : SFC) permet l'utilisation d'index mono-attributs tels que les B-arbres. L'objectif de la transformation d'un espace multidimensionnel en un espace unidimensionnel est que chaque n-uplet de données possède un identificateur unique dans ce nouvel espace. Cet identificateur permet d'utiliser une structure d'indexation mono-attribut, avec les optimisations de temps et d'espace qui lui sont attachées [14].

A. Les courbes de remplissage de l'espace

Dans un espace multidimensionnel, où chaque attribut participant à l'index représente une de ses dimensions, un n-uplet de données est représenté par un point ; la projection du point sur une des dimensions d_i donne la valeur du n-uplet pour l'attribut d_i. Les coordonnées d'un point dans un espace multidimensionnel représentent donc les valeurs des attributs d'un n-uplet.

Une méthode de remplissage de l'espace est une méthode qui génère et affecte à chaque point un identificateur unique, en utilisant les coordonnées du point. Cet identificateur définit la position du point dans un espace unidimensionnel. Chaque courbe de remplissage de l'espace utilise une stratégie de détermination de l'ordre de « visite » des points de l'espace multidimensionnel. La courbe de remplissage de l'espace impose donc un ordre linéaire sur les points. Cette linéarisation permet d'utiliser une structure d'indexation unidimensionnelle pour indexer les objets : ceux-ci sont indexés par leur identificateur. En conséquence, la taille de la structure d'indexation est plus petite et l'algorithme de recherche est plus performant. Un autre avantage de cette approche est que le phénomène de chevauchement (*overlapping*), bien connu dans les structures comme le R-arbre, est évité.

Les courbes de remplissage de l'espace les plus utilisées sont la courbe Z (Peano), la courbe de Gray, la courbe de Hilbert, la courbe Sweep et la courbe Scan. Les travaux de comparaison des cinq courbes [10] ont montré que les courbes Z et Gray deviennent équitables [1] quand le nombre de dimensions de l'espace augmente, et qu'elles assurent la conservation du voisinage[2]. Ces propriétés, associées au faible coût de l'algorithme de calcul de l'identificateur "Z" d'un n-uplet de données par rapport aux autres courbes, font de la courbe Z un bon choix pour la transformation d'un espace multidimensionnel en un espace unidimensionnel.

La courbe Z a été introduite par Peano [11]. L'identificateur de chaque n-uplet de données est appelé sa valeur Z. Le calcul de la valeur Z d'un n-uplet de données $<a_1, a_2, a_3, ..., a_n>$ est fait par entrelacement (*bit-interleaving*) de la représentation binaire des éléments du n-uplet. Le nombre de bits nécessaire pour représenter les valeurs d'un attribut Ai est égal à : $NBits = \lceil \log|D_i|/\log 2 \rceil$; $|D_i|$ est la cardinalité du domaine $|D_i|$. $i \in [1..n]$. Les représentations binaires sont rangées en ordre descendant par rapport aux NBits des attributs. On prend de façon cyclique un bit du code binaire de chaque élément du n-uplet et on l'adjoint aux autres bits obtenus auparavant.

[1] Une courbe de remplissage de l'espace est *équitable* si elle a le même comportement dans toutes les dimensions dans un espace multidimensionnel.

[2] Si deux points sont proches (voisins) dans l'espace multidimensionnel, ils doivent le rester dans un espace unidimensionnel.

B. UB-Tree (Universal B-tree)

L'UB-arbre proposé par Bayer [12][13] est une structure d'indexation multidimensionnelle basée sur le B-arbre [14] et sur la courbe Z de remplissage de l'espace [9]. Pour chaque point de l'espace multidimensionnel, sa valeur Z est calculée et lui est affectée. La valeur Z est appelée *l'adresse Z du point* ou simplement *l'adresse Z*. Les données sont alors indexés par adresses Z en utilisant un B-arbre.

Un UB-arbre partitionne tout l'espace multidimensionnel en blocs disjoints et contigus, appelés *régions-Z*. Une région-Z représente une collection de points dans l'espace indexé. Chaque région-Z est bornée par deux adresses Z : celle de la borne inférieure et celle de la borne supérieure de la région. L'adresse Z d'une région-Z est l'adresse Z de sa borne supérieure.

Dans un UB-arbre, les points appartenant à une région-Z sont stockés dans un nœud feuille. Les nœuds internes (non feuilles) de la structure de l'arbre constituent des régions appelées régions-Z-père. Au niveau inférieur, une région-Z-père a pour fils des régions-Z ou des régions-Z-père.

Une variante de l'UB-arbre, appelée BUB-arbre (Bounding UB-tree: BUB-tree) [15], n'indexe que les régions-Z contenant des données, c'est-à-dire l'*espace actif*.

La Figure 1 montre la partition par un UB-arbre d'un espace bidimensionnel. On a 15 régions-Z disjointes. Dans cet exemple, la capacité maximale d'une région-Z est de trois objets.

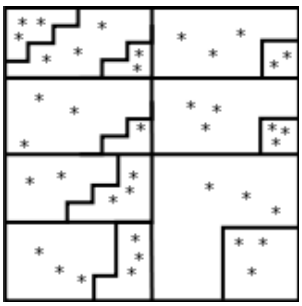

Figure 1. Partition d'un espace bidimensionnel en Régions-Z

Les solutions classiques UB-arbre ou BUB-arbre linéarisent uniquement le parcours de l'espace des objets. Dans notre travail nous introduisons un double niveau d'indexation en indexant non seulement les objets, mais également les blocs de la partition de l'espace (les Eq-classes, ou clusters). Cette solution permet de prétraiter les requêtes relativement aux clusters, et donc de diminuer les transferts de données vers la mémoire centrale.

III. CONTEXTE : LE SYSTEME OSIRIS

Le système Osiris est un système de représentation et de gestion des données et des connaissances (SGBD-BC) qui implante le modèle des P-types, P pour partagé [1]. Ce concept permet la description d'une famille d'objets du monde réel par la hiérarchie de vues selon laquelle on veut percevoir ses objets : on définit d'abord la *vue minimale* d'un P-type, puis ses autres *vues* par la spécialisation stricte d'une ou de plusieurs vues déjà déclarées. Lors de la spécification d'une vue, des attributs propres et des contraintes peuvent être données. Parmi ces contraintes, les contraintes de domaine et les Dépendances Inter-Attributs jouent un rôle particulier. En effet, les contraintes de domaine sont utilisées par Osiris pour partitionner l'espace des objets en classes d'équivalence, les Eq-classes, qui satisfont la condition « avoir la même valeur pour toutes les contraintes de domaine définis dans toutes les vues d'un P-type ». L'espace des Eq-classes est nommé Espace de Classement en Osiris. Cet espace est à la base de l'optimisation sémantique des requêtes : tout objet, créé ou modifié, est classé et (ré-)indexé par une Eq-classe (cas d'un objet complètement valué) ou par un ensemble d'Eq-classes (cas d'objets incomplètement valués).

A. *Un exemple*

Considérons l'exemple simplifié d'un P-Type PERSON avec les attributs nom, id, age, sex, salary et militaryService, et des vues MINOR, ADULT, SENIOR, SENIOR-MALE, SENIOR-FEMALE, EMPLOYEE, CEO. On a les contraintes suivantes :

1)Définition des domaines des attributs

age: INT **in** [0..120];
sex: CHAR **in** {m, f};
salary: INT ≥ 600;
militaryService: STRING **in** {no, ongoing, done, deferred, exempt};

2)Contraintes définissant les vues

View MINOR : PERSON // inheritance (IS-A)
age < 18

View ADULT : PERSON; age ≥ 18

View SENIOR : PERSON; age ≥ 60

View SENIOR_MALE : SENIOR; sex = m

View SENIOR_FEMALE : SENIOR; sexe= f

View EMPLOYEE : PERSON; salary ≥ 1200

View CEO : PERSONNE ; salary ≥ 9000

3)Dépendances inter-attributs (DIA)

Les DIAs sont des clauses de Horn dont les prédicats élémentaires sont des contraintes de domaine. Dans cet exemple, on considère les DIAs suivantes :

age<18 → salary < 1200
age<18 → militaryService=no
age≥18 **and** sex=m →
militaryService **in** {done, ongoing, deferred, exempt)
sex=f → militaryService = no

B. *Espace de Classement*

Considérons l'attribut $Attr_i$ de domaine Δ_i, et l'ensemble $P(Attr_i)$ des prédicats définissant les contraintes de domaine du P-Type. Toute contrainte de domaine sur $Attr_i$ peut se réécrire sous la forme : $Attr_i \in D_{ik}$, où $D_{ik} \subseteq \Delta_i$.

Une contrainte de domaine partitionne Δ_i en deux éléments : D_{ik} et $(\Delta_i - D_{ik})$. Le produit des partitions [6] définies par $P(Attr_i)$, définit la plus petite partition de Δ_i. Les blocs d_{ij} de cette partition sont appelés Sous-Domaines Stables (SDS). Tout SDS vérifie la propriété de stabilité : lorsque, pour un objet, la valeur de l'attribut $Attr_i$ varie à l'intérieur d'un SDS d_{ij}, l'objet continue de satisfaire exactement les mêmes prédicats de $P(Attr_i)$. En conséquence, toute mise à jour qui conserve les SDS de l'objet ne nécessite pas de reclassement de l'objet.

Dans l'exemple précédent, les attributs participant à la partition sont age, sex, salary et militaryService. Les contraintes ci-dessus définissent les SDS suivants:

age : d_{11} = [0,18[, d_{12} = [18,70[, d_{13} = [60,120]
sex: d_{21} = {f}, d_{22} = {m}
salary : d_{31} = [0,600[, d_{32} = [600,1200[,
d_{33} = [1200,9000[, d_{34} = [9000,SUP[3]]
militaryService: d_{41} = {no}, d_{42} = {ongoing, done, deferred, exempt}

La partition du domaine de chaque attribut d'un P-type se prolonge en la partition de l'espace de ses objets, et constitue l'*Espace de Classement* du P-type. La propriété de stabilité des SDS se prolonge aux Eq-Classes : tous les objets appartenant à la même Eq-classe satisfont les mêmes contraintes d'un P-type, et donc appartiennent au même sous-ensemble de ses vues.

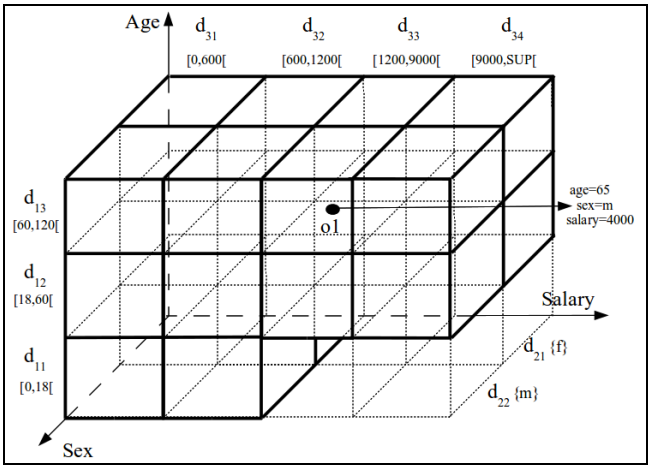

Figure 2. Espace de Classement du P-type PERSON (3 dimensions)

La Figure 2 représente l'Espace de Classement du P-type PERSON en considérant les dimensions sex, salary et age. Les Eq-classes (d11, *, d33) et (d11, *, d34) sont invalides car aucun de leurs objets ne satisfait la DIA

age<18→salary<1200

Le résultat d'une requête comportant une condition (age<18 et salary>1200) sera vide ; aucune recherche ne sera effectuée dans l'espace des objets car toutes les Eq-classes « contenues » dans la requête sont invalides [5].

[3] SUP désigne la borne supérieure d'un domaine numérique.

IV. 2L-UB-INDEX :UN DOUBLE NIVEAU D'INDEXATION

A. *Architecture*

Étant donné le nombre potentiellement exponentiel des Eq-classes par rapport au nombre de dimensions et au nombre de d'éléments de la partition du domaine d'un attribut, seuls sont indexées les **Eq-classes actives**, c'est-à-dire celles contenant au moins un objet. Nous utilisons donc les BUB-arbres, une variante des UB-arbres qui indexe l'espace actif (cf §II.B).

Dans l'approche que nous proposons, appelée **2L-UB-index** (2L pour 2-Level), l'indexation se fait à deux niveaux :

- au niveau des Eq-Classes,
- au niveau des objets d'une Eq-Classe.

Les Eq-classes actives d'un P-Type sont indexées par un arbre, nommé **EqUB-arbre** (UB-arbre des Eq-classes). Dans un EqUB-arbre, les Eq-classes sont stockées dans une région-Z[4] (nœud feuille).

Les objets d'une Eq-classe sont indexés par un arbre nommé **DUB-arbre** (D pour Données). Il y a autant de DUB-arbres que d'Eq-classes actives. Dans un DUB-arbre, les objets sont stockés dans une région-Z (nœud feuille). Cette structure d'indexation est présentée dans la Figure 3.

Les deux types d'arbres ont la même structure, avec les différences suivantes :

- La racine d'un DUB-arbre contient les Sous-Domaines Stables des attributs et le nombre d'objets de l'Eq-classe.
- Les éléments des nœuds feuilles d'un EqUB-arbre contiennent des pointeurs vers ses DUB-arbres. Les nœuds feuilles d'un DUB-arbre stockent les objets d'une Eq-classe.

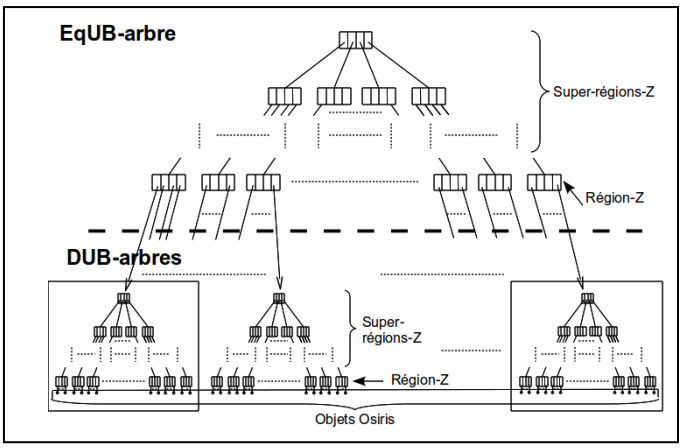

Figure 3. 2L-UB-index: un EqUB-arbre, ses DUB-arbres et les régions

B. *Algorithmes d'insertion et de recherche*

Les algorithmes d'insertion, de recherche et de suppression d'un objet dans un 2L-UB-Index se basent sur les algorithmes des UB-arbres, qui se basent eux-mêmes sur les algorithmes du B-arbre. Le classement de l'objet et le calcul des identifiants internes des Eq-Classes et des objets sont deux étapes préliminaires dans les algorithmes en Osiris. Nous détaillons les algorithmes d'insertion et de modification d'un objet.

1)Insertion d'un objet:

1) Déterminer l'Eq-Classe à laquelle l'objet appartient, en le classant dans son P-type. Si l'objet ne satisfait pas les contraintes de la vue minimale du P-type, il ne sera pas accepté par le système et il ne sera pas ajouté.
2) Calculer l'identifiant interne de l'Eq-Classe et l'identifiant interne de l'objet.
3) Vérifier si l'Eq-Classe est dans l'EqUB-arbre en y cherchant son identifiant interne :
   - si OUI, vérifier si l'objet est dans le DUB-arbre de l'Eq-Classe.
     - si OUI, un message est envoyé pour informer que l'objet est dans la base.
     - Si NON, l'objet est inséré dans le DUB-arbre.
   - Si NON, ajouter l'Eq-Classe dans l'EqUB-arbre, créer le DUB-arbre de l'Eq-Classe et y ajouter l'objet.

2)Modifier une ou plusieurs valeurs d'attributs d'un objet:

Deux cas sont possibles:

1) Les valeurs modifiées sont celles d'attributs non classificateurs. L'objet ne change pas d'Eq-Classe et il n'y a pas de changement au niveau des arbres.

2) Une (ou plusieurs) valeurs modifiées sont celles d'attributs classificateurs. Si aucun attribut ne change de SDS, l'objet appartient toujours à la même Eq-Classe, et il n'y a pas de changement au niveau des arbres. Autrement, l'objet change d'Eq-Classe : l'objet est supprimé de son DUB-arbre initial, puis l'objet modifié est ajouté dans son nouveau DUB-arbre.

C. *Requêtes par intervalles (Range query)*

Puisque l'indexation des objets en Osiris est réalisée au travers des Eq-Classes, l'ensemble des Eq-Classes répondant aux conditions de la requête sont d'abord recherchées, puis les objets de ces Eq-Classes sont récupérés.

L'intérêt de ce double niveau d'indexation est que, au lieu de vérifier individuellement si les objets appartiennent ou non à l'espace de la requête, on commence par déterminer les Eq-Classes actives qui englobent l'espace des solutions de la requête. Deux cas sont possibles : 1) l'espace de l'Eq-classe est **inclus** dans l'espace de la requête ; 2) il existe une **intersection** non vide entre l'espace de l'Eq-classe et l'espace de la requête. Dans le premier cas, tous les objets de l'Eq-classe satisfont la requête. Dans le deuxième cas, les objets doivent être testés individuellement [5].

L'algorithme DRU [17] (pour Down Right Up) a été proposé pour répondre aux requêtes par intervalles dans un seul UB-arbre. DRU détermine s'il y a une **intersection** entre une région-Z et la requête. Pour améliorer les performances de l'algorithme dans le système Osiris, nous avons ajouté le calcul de **l'inclusion**.

[4] On rappelle que dans un UB-arbre, les nœuds non-feuille constituent les super-région-Z (cf §II.B).Une super-région-Z a pour fils des région-Z ou des super-région-Z. Chaque région est bornée par deux identifiants internes: le plus petit et le plus grand des identifiants des points de la région.

- Les objets des Eq-Classes incluses dans la requête satisfaisant de manière sûre la requête, sont récupérés sans vérification supplémentaire. C'est par exemple le cas de l'Eq-Classe EC1 de la Figure 4.
- Les objets des Eq-Classes intersectant la requête doivent être examinés individuellement. C'est par exemple le cas de l'Eq-Classe EC2 de la Figure 4.

Nous avons appelé le nouvel algorithme DRU-Eq (pour Down Right Up-Eq-Classes).

Dans un espace à N dimensions, nous considérons deux hypercubes A et B, dont les intervalles sur la i^{eme} dimension sont $[A_{li},A_{ui}]$ et $[B_{li},B_{ui}]$, avec $A_{li} \leq A_{ui}$ et $B_{li} \leq B_{ui}$.

- A et B ont une intersection non vide ssi :
  $\forall i \in [1..N], |A_{li}-B_{li}| + |A_{ui}-B_{ui}| \leq |A_{li}-A_{ui}| + |B_{li}-B_{ui}|$
- Il y a une relation d'inclusion entre A et B ($A \subseteq B$ ou $A \supseteq B$) ssi A et B ont une intersection non vide et:
  $\forall i \in [1..N], |A_{li}-B_{li}| + |A_{ui}-B_{ui}| = ||A_{li}-A_{ui}| - |B_{li}-B_{ui}||$

L'algorithme DRU-Eq utilise ces propriétés pour déterminer la nature de la relation entre la requête et une super-Z-région (intersection, inclusion ou ni l'une ni l'autre).

Une requête par intervalles est encadrée (bornée) par deux points de l'espace multidimensionnel (qui possèdent deux valeurs-Z dans l'espace unidimensionnel) appelés : *le point inférieur* et *le point supérieur* de la requête. Exemple : La requête RQ2 dans la figure 4 est encadrée par le point inférieur A et par le point supérieur B. Dans l'approche 2L-UB-Index ce sont des objets appelés *l'objet inférieur de la requête, Oi*, et *l'objet supérieur de la requête, Os*. Ces deux objets sont des objets *fictifs* qui seront utilisés uniquement pour déterminer les Eq-Classes encadrant la requête. Dans un EqUB-arbre, nous les appelons l'Eq-Classe inférieure *ECi* et l'Eq-Classe supérieure *ECs* de la requête.

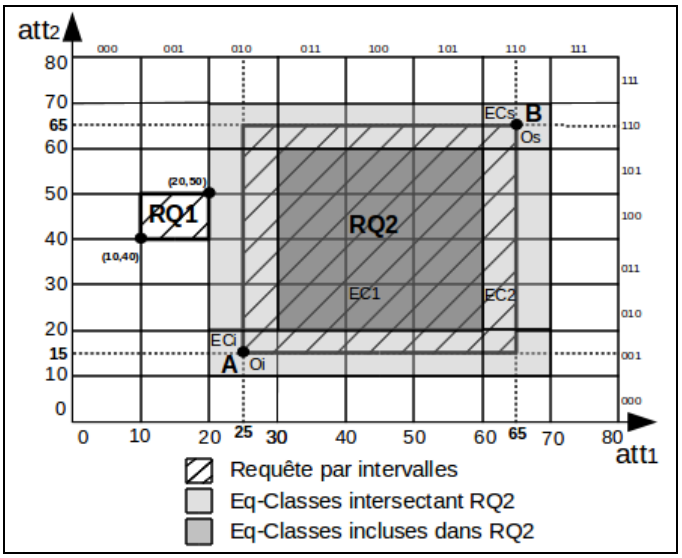

Figure 4. Requêtes par intervalles et Eq-Classes dans un espace bidimensionnel

Dans la suite, au travers de l'exemple de la Figure 4, nous montrons le calcul des identifiants de l'objet inférieur Oi et de l'objet supérieur Os de la requête, puis le calcul des identifiants de l'Eq-Classe inférieure et l'Eq-Classe supérieure de la requête. Le P-Type de l'exemple de la Figure 4 a deux attributs classificateurs att_1 et att_2, où chaque attribut a 8 sous-domaines stables:

- $SDS_{att1} = \{d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, d_{16}, d_{17}, d_{18}\}$
- $SDS_{att2} = \{d_{21}, d_{22}, d_{23}, d_{24}, d_{25}, d_{26}, d_{27}, d_{28}\}$

Considérons les requêtes :

RQ1:{ att1$\in$ [10,20] et att2 $\in$ [40,50] }
RQ2:{ att1$\in$ [25,65] et att2 $\in$ [15,65] }

Les valeurs de l'objet inférieur de la requête sont les valeurs inférieures de chaque intervalle des attributs de la requête, les valeurs de l'objet supérieur de la requête sont les valeurs supérieures de chaque intervalle des attributs de la requête.

Dans RQ1 l'objet inférieur est Oi ={10,40} et l'objet supérieur est Os ={20,50}.

Dans RQ2 l'objet inférieur est Oi ={25,15} et l'objet supérieur est Os ={65,65}.

En classant les deux objets fictifs Oi et Os, nous obtenons l'Eq-Classe inférieure *ECi* et l'Eq-Classe supérieure *ECs* de la requête.

- L'objet Oi appartient à l'Eq-classe inférieure
  $EC_i = \{d_{13} = [20,30], d_{22} = [10,20]\}$
- l'objet Os appartient à l'Eq-classe supérieure
  $EC_S = \{d_{17} = [60,70], d_{27} = [60,70]\}$.

Le calcul des identifiants internes de l'Eq-Classe inférieure et de l'Eq-Classe supérieure donne:

$EC_i = \{d_{13} = [20,30] = 010_2 , d_{22} = [10,20]=001_2\}$

Application de l'ordre-Z : $EC_i = 000110_2 = 6_{10}$.

$EC_S = \{d_{17} = [60,70] = 110_2 , d_{27} = [60,70]= 110_2 \}$

Application de l'ordre-Z : $EC_S = 111100_2 = 60_{10}$.

Les objets qui satisfont la requête RQ2 sont les objets appartenant aux Eq-Classes qui ont des identifiants internes entre [6, 60] et qui satisfont la requête.

Lorsque la requête est identique à une Eq-Classe, l'Eq-Classe inférieure EC_i et l'Eq-Classe supérieure EC_S sont identiques. Les objets de l'Eq-Classe satisfont tous la requête. C'est le cas de la requête RQ1 (Figure 4), où l'espace de la requête est identique à l'espace de l'Eq-Classe $\{d_{12} = [10,20], d_{25} = [40,50]\}$.

Comme DRU, DRU-Eq utilise une pile pour le parcours de l'arbre. Les entrées sont l'EqUB-arbre et les identifiants internes des Eq-Classes supérieure et inférieure de la requête. Les sorties sont un ensemble d'Eq-Classes incluses dans la requête et un ensemble d'Eq-Classes qui intersectent la requête. Les étapes importantes de l'algorithme DRU-Eq sont identiques à celles de DRU, avec la vérification d'inclusion.

V. EVALUATION

Pour évaluer les performances de la méthode d'indexation 2L-UB-Index avec celles de l'indexation par UB-arbre, le temps moyen d'exécution d'un nombre important de requêtes de point et de requêtes par intervalles a été comparé pour évaluer les performances de deux méthodes.

Trois familles de projets Osiris ont été écrites pour le test. Elles diffèrent par le nombre d'attributs classificateurs et par le nombre de sous-domaine stables de chaque attribut classificateur. Pour chaque projet, nous avons créé quatre bases de données de test avec différents nombre d'objets générés automatiquement et aléatoirement (entre 50 000 et 1 400 000). Au total 120 bases de test ont été créées : 60 bases indexées par des UB-arbres et 60 indexées par des 2L-UB-Index.

Nous avons testé les performances sur deux types de requêtes: les requêtes de point et les requêtes par intervalles. Pour ces dernières, nous avons distingué deux familles : les requêtes qui enveloppent une seule Eq-Classe (cf RQ1 Fig. 4) et celles qui enveloppent plusieurs Eq-Classes. Les résultats sont sensiblement identiques. Nous présentons les résultats du test de la dernière famille de requêtes dans le tableau I et les explications dans le tableau II.

TABLE I. GAIN DE 2L-UB-INDEX PAR RAPPORT A BUB-ARBRE

RQ enveloppant complètement plusieurs Eq-Classes Ex: RQ couvrant toute la partie grise foncée de la figure 4		
Pire cas	***En moyenne***	***Meilleur cas***
10% (1)	55%	85%
RQ enveloppant complètement un ensemble d'Eq-Classes et partiellement un autre ensemble d'Eq-Classes Ex: RQ2 dans la figure 4		
Pire cas	***En moyenne***	***Meilleur cas***
15% (2)	75%	90% (3)
RQ ne renvoie aucun objet		
Pire cas	***En moyenne***	***Meilleur cas***
Gain très important et constant (4)		

TABLE II. EXPLICATIONS DE L'EVALUATION

Cas	Explication
1	Requêtes retournant un très grand nombre d'objets (un tiers ou la moitié de la base). Nombre d'Eq-Classes très faible
2	Requêtes retournant un grand nombre d'objets (<15% de la base). Le nombre d'Eq-Classes qui sont enveloppées partiellement par la requête et plus important que le nombre d'Eq-Classes qui sont complètement enveloppées. Les objets du premier groupe d'Eq-Classes doivent être vérifiés, contrairement à ceux du deuxième groupe.
3	Cas contraire du cas 2 : Le nombre d'Eq-Classes qui sont enveloppées partiellement par la requête et moins important que le nombre d'Eq-Classes qui sont complètement enveloppées.
4	La valeur moyenne du temps de réponse est comprise entre 30 ms et 60 ms (entre 200ms et 3000ms pour le BUB-arbre). Ces requêtes ne renvoyant pas d'objet, il n'y a pas d'Eq-Classes qui les satisfont. Seules les Eq-Classes actives étant indexées, la recherche s'arrête au niveau de l'EqUB-arbre. En conséquence, la recherche dans les DUB-arbres n'est pas effectuée.

VI. CONCLUSION

La méthode 2L-UB-index d'indexation à deux niveaux des données multidimensionnelles a été élaborée dans le cadre du système Osiris, qui offre de manière native une partition de l'espace des objets basée sur l'analyse statique des contraintes d'intégrité définissant les vues d'un P-type [1]. C'est le travail qui a été présenté dans cet article.

Cette méthodologie, qui offre des gains significatifs pour des requêtes par intervalles par rapport aux gains offerts par les méthodologies classiques, peut être réutilisée par tout système gérant des données multidimensionnelles et qui permet une partition du domaine des attributs, ce qui est le cas des entrepôts de données et des SIG. Une telle partition peut être décidée par le concepteur, ou encore être basée sur des considérations résultant d'une analyse des données.

L'adaptation de notre approche aux SGBD classiques constitue la prochaine étape de notre travail.

REFERENCES

[1] A. Simonet and M. Simonet, "Objects with views and constraints : From databases to knowledge bases," in OOIS, 1994, pp. 182–195.

[2] "Oracle cluster," http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96524/c11schem.htm#25479.

[3] Postgresqlcluster www.postgresql.org/docs/8.2/static/sql-cluster.html.

[4] M. Roger, A. Simonet, and M. Simonet, "A DL-like model for a knowledge and data management system" in DEXA '00. London, UK: Springer-Verlag, 2000, pp. 563-572.

[5] A. Simonet, M. Simonet, "Classement d'instance et Evaluation des Requêtes en Osiris" in BDA'96 : Bases de Données Avancées, Cassis, France, Août 1996, pp 273-288.

[6] D. F. Stanat and D. F. McAllister, Discrete mathematics in computer science (4. pr.). Prentice Hall, 1977.

[7] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The x-tree: An index structure for high-dimensional data" 1996, pp. 28-39.

[8] J. A. Orenstein and T. H. Merrett, "A class of data structures for associative searching" in PODS, 1984, pp. 181-190.

[9] H. Sagan, Space-Filling Curves, 1st ed. Springer, September, 1994,.ISBN-10: 0387942653 ISBN-13: 978-0387942650.

[10] M. F. Mokbel, W. G. Aref, and I. Kamel, "Analysis of multi-dimensional space-filling curves" GeoInformatica, vol. 7, no. 3, pp. 179-209, 2003.

[11] G. Peano, "Sur une courbe qui remplit toute une aire plane" Mathematishe Annalen, vol. 36, no. 1, pp. 157-160, 1890.

[12] R. Bayer, "The universal B-tree for multidimensional indexing: general concepts" in WWCA, 1997, pp. 198-209.

[13] V. Markl, "Mistral: Processing relational queries using a multidimensional access technique" PhD thesis, D. T. U. Munchen 1999.

[14] R. Bayer and E. M. McCreight, "Organization and maintenance of large ordered indices" Acta Inf., vol. 1, pp. 173-189, 1972.

[15] R. F. Fenk, "The BUB-tree" in In VLDB '02, Hong Kong. Morgan Kaufman Publishers, 2002.

[16] S. Housseno, A. Simonet, and M. Simonet, "Ub-tree indexing for semantic query optimization of range queries" International Journal of Computer, Information and Mechatronic Engineering, issue 35, vol. 59, pp. 177–184, 2009.

[17] T. Skopal, M. Krátký, J. Pokorný, and V. Snásel, "A new range query algorithm for universal B-trees" Inf. Syst., vol. 31, no. 6, pp. 489-511, 2006.