REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de L'enseignement Supérieur et de la Recherche Scientifique

Université IBN KHALDOUN Tiaret

Faculté des sciences, de la technologie et des sciences de la matière

Département d'informatique

Mémoire de fin d'études
Pour l'obtention du diplôme

de mastère 2 en génie informatique

Option : système d'information et technologie web

Evaluation des requêtes à préférence dans un contexte distribuée

Présenté par :

M^lle^ Dihia REMMANE

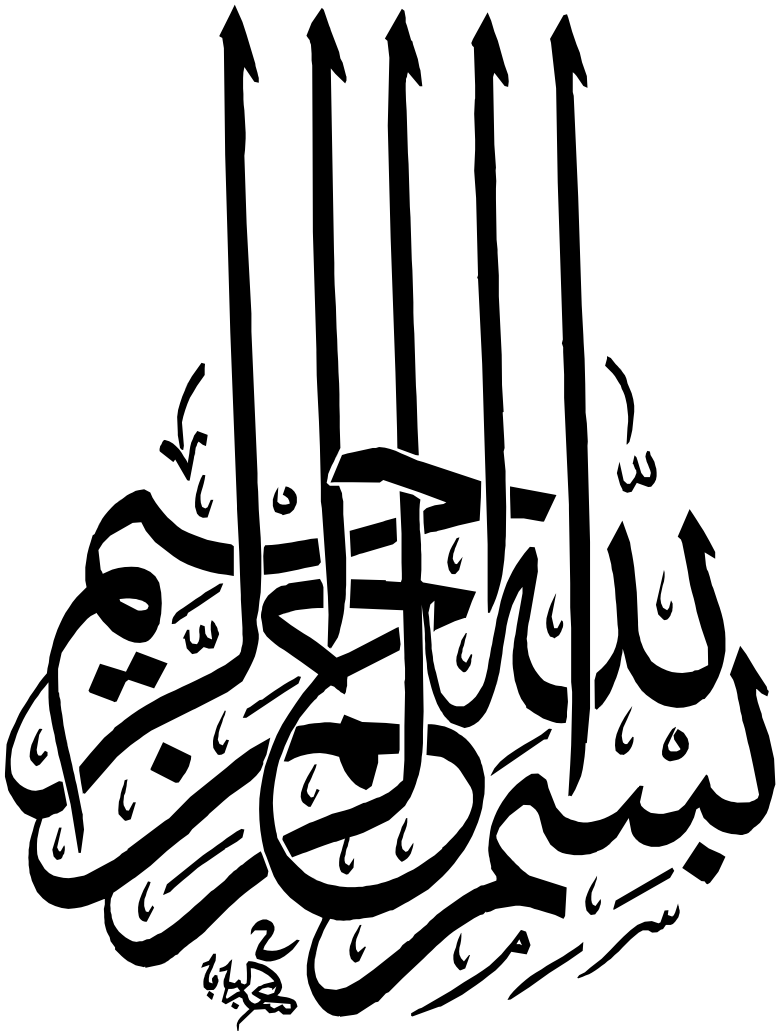
Promoteur :

- Abdelkader ALEM

Membres de jury :

- Sahraoui KHAROUBI
- Mokhtar MOSTFAOUI

Année Universitaire: 2011 – 2012


بسم الله الرحمن الرحيم

Remerciement

Au terme du présent ce travail, je tenue à remercier tout d'abord :

Au Allah, le clément, le miséricordieux puissant qui ma donné durant toutes ces années, la santé, le courage

C'est avec beaucoup de respect et d'estime que je remercie mon promoteur ***Mr Abdelkader ALEM****, son aide bénéfique qui a assuré la direction scientifique de ce travail et qui a pris le temps de me faire profiter avec patience de son expérience et ses connaissances, qu'il puisse croire à mes profondes gratitudes.*

Mes sincères remerciements à ***Mr KHAROUBI*** *et* ***Mr MOSTEFAOUI*** *qui acceptent d'être des jurys*

A tous les enseignants qui ont contribué a me fournir des connaissances qu'ils trouvent mes remerciement les plus sincère

Enfin je remercie tous ceux qui ont contribue de prés ou de loin à la réalisation de ce travail.

Tables des matières

Introduction Générale 1

Chapitre I : Requêtes à Préférences 3

1.Définitions 3

1.1.Requêtes à préférences 3

1.2.La notion de préférence 3

1.3.Requêtes à préférences 4

1.4.La notion de préférence 4

1.5.Relation de préférence 4

1.5.1.Définition 4

1.5.2.Propriétés 4

2.Approche quantitative [AWF 00] et [HKP 01] 5

1.1.Définition 5

2.1Top_k queries [CHE 05] 5

2.1.1Principe 5

3.Approche qualitative 7

3.1.Ordre de Pareto 7

4.Requêtes skyline 7

4.1.Principe de skyline 8

4.2.Fondements mathématique 9

4.2.1.Problème de vecteur maximal 9

4.3.Comparaisons des points 10

4.4.Principe de dominance 11

4.4.1.Propriétés de la relation de dominance (Ω_S) 11

4.5.Définition de Skyline 11

4.5.1Les propriétés de skyline 12

5.Expression des requêtes skyline 12

6.Conclusion.....14

ChapitreII: Approches d'Evaluation de Skyline dans un Contexte Centralisé.....15

1.Algorithme standard.....15

2.Algorithme BNL (Block-Nested-Loops).....16

2.1Les avantages et les inconvénients de BNL.....19

3.Algorithme Bitmap.....20

3.1.Description de l'algorithme.....20

4.Algorithme BBS (Branch-and-Bound Skyline Algorithm).....23

5.Discussion.....25

6.Conclusion.....27

ChapitreIII :Approches d'Evaluation de Skyline dans un Contexte Distribué.....28

1.Le WEB et le modèle Client/Serveur.....29

1.2.Définition.....29

1.3.Modèle client/serveur.....29

1.4.Avantages et inconvénients de l'architecture client/serveur.....30

2.Calcul de skyline dans un environnement distribué.....31

2.1.Algorithme BDS (Basic Distributed Skyline Algorithm).....31

2.2.Algorithme IDS (Improved Distributed Skyline Algorithm).....33

2.3.Algorithme PDS (Progressive Distributed Skyline algorithm).....34

2.3.1.La progressivité.....34

2.3.2.Estimation de rang.....35

3.Conclusion.....36

ChapitreIV :Contribution à L'Évaluation de Skyline Dans un ED.....37

1.Evaluation de requêtes skyline dans un contexte distribué : Une approche hybride.....38

1.1.Stratégie d'Evaluation.....38

2.Implémentation.....40

2.1.Le langage java.....40

2.2.Outil Netbeans.....40

2.3.MySQL.....41

2.4.La connexion à une base de données 42

2.5.Scenario d'exécution 42

Conclusion Générale 48

Bibliographie

Liste des figures

Figure I.1 : Relation voiture 6

Figure I.2 : Algorithme top-k Queries 6

Figure II.1 : L'ensemble de point en 2D 24

Figure II.2 : R-arbre 24

Figure III.1 : Architecture Client/serveur 29

Figure III.2 : Les tableaux qui sauvegardent les objets retournent dans la 1ère phase 33

Figure IV.1 : Les tableaux KI 38

Figure IV.2 : Le tableau P (Tuples retournés dans la 1ère phase). 39

Figure IV.3 : Interface Netbeans 40

Figure IV.4: Interface MySQL 41

Figure IV.5: Interface de créer une base de données 41

Figure IV.6 : Interface de l'application 43

Figure IV.7 : La barre menue de l'application 43

Figure IV.8 : Demande de connexion à la base de données 44

Figure IV.9 : Donnée sur le site 1 44

Figure IV.10 : Charger les données 45

Figure IV.11 : Traitement des données 45

Figure IV.12 : Algorithme BDS 46

Figure IV.13 : Algorithme BDS par Etapes 46

Figure IV.14: Algorithme BDS (Première phase) 47

Figure IV.15 : Algorithme BDS(deuxième phase) 47

Figure IV.16 : Algorithme BDS(deuxième phase) 47

Liste des tableaux

Tableau I.1 : Exemple des caméras avec leurs qualités et prix 8

Tableau II.1 : Algorithme standard 16

Tableau II.2 : Les étapes d'algorithme BNL. 18

Tableau II.3 : L'approche Bitmap. 22

Tableau II.4 : Contenu du tas. 25

Tableau II.5 : Comparaison entre les algorithmes BNL, Bitmap et BBS 26

Tableau III.1 : Exemples des hôtels avec 3 attributs sur 3 sites différents 32

Introduction Générale

Introduction Générale

Les systèmes de recherche d'information sont des outils qui ont permis, jusqu'à aujourd'hui d'améliorer la qualité des services d'accès à l'information. Au fur et à mesure que le volume de données s'accroit et les informations se diversifient, ces systèmes (moteur web, SGBD…) délivrent des résultats massifs, générant ainsi une surcharge informationnelle en réponse aux requêtes des utilisateurs dans laquelle il est difficile de distinguer l'information pertinent d'une information secondaire. Donc le problème n'est pas dans la disponibilité de l'information mais dans sa pertinence relative aux besoins précis de l'utilisateur, c'est pourquoi les travaux s'orientent actuellement vers la révision de cycle de vie d'une requête dans la perspective d'intégrer l'utilisateur comme composante du modèle globale de recherche et ce, dans le but de lui délivrer une information pertinente adaptée à ses besoins et préférences.

Dans les Systèmes Relationnels traditionnel, l'interrogation des bases de données (BD) nécessite une connaissance précise et détaillée des données et de leur organisation. L'intégration des préférences d'utilisateur dans ces requêtes tente de rendre l'interrogation classique des BD plus souple pour les utilisateurs. L'intérêt majeur des requêtes à préférences est, d'une part, de pouvoir traduire fidèlement ce que l'utilisateur désire exprimer et, d'autre part, de produire un ensemble de réponses ordonnées, ce qui est particulièrement intéressant dans le cas où l'ensemble des items satisfaisant la requête est de taille importante.

En général., il est possible de distinguer deux familles d'approches pour l'expression des préférences. Une approche quantitative (implicite) où chaque valeur d'attribut est associée à un score, une valeur étant préférée à une autre si elle a obtenu un meilleur score. Une approche qualitative (explicite) où les préférences sont définies en comparant les valeurs d'attributs deux à deux (relations de préférences binaires).

1. Objectif

Dans un premier temps, le travail consiste à étudier les requêtes à préférences sous l'angle des requêtes dites ***skyline*** de point de vue expression et implémentation dans un contexte distribué. Un deuxième point consistera à améliorer et compléter les méthodes existantes.

L'approche proposée consiste à combiner l'algorithme BNL (pour le cas des BD centralisés) et l'algorithme BDS (pour le cas des BD accessibles via le WEB).

2. Structure du mémoire

Le mémoire est structuré en quatre chapitres organisés comme suit : Le premier chapitre est dédié à la présentation d'une vue globale sur les requêtes à préférences ainsi que différents modèles de représentation des préférences. Ensuite le chapitre 2 présente les différentes approches proposées pour le traitement des requêtes Skyline dans un contexte centralisé. Dans le chapitre 3 on s'intéresse aux principales méthodes d'évaluation des requêtes Skyline dans le cas des bases de données distribuées. Pour chaque méthode, on décrit explicitement son principe et un exemple illustratif est proposé. Le chapitre 4 est consacré à la présentation de notre contribution et à la réalisation de notre application. Nous présentons des illustrations pratiques et les résultats d'exécutions. En fin nous terminons par une conclusion et les perspectives à prospecter.

CHAPITRE 1

Requêtes à Préférences

Le volume de l'information gérée par les systèmes de gestion de base de données (SGBD) devient de plus en plus important et, en conséquence, son interrogation se doit d'être de plus en plus performante. Cette performance peut être évaluée en termes de temps de réponse ou en termes de pertinence de la réponse délivrée. Dans ce dernier cas, l'intégration des préférences de l'utilisateur dans la requête est un moyen de personnaliser la recherche et d'obtenir des réponses plus adéquates. De telles requêtes impliquent la présentation à l'utilisateur d'un ensemble discriminé de réponses (des plus au moins préférées). Le problème de l'expression des préférences utilisateurs dans les requêtes a reçu beaucoup d'attention ces dernières années.

Des préférences élémentaires sont définies sur les valeurs d'attributs, puis composées ensuite afin de définir des préférences plus complexes. Pour un attribut donné, deux façons générales de procéder sont utilisées pour exprimer les préférences des utilisateurs : quantitative (implicite) ou qualitative (explicite).

1. Définitions

1.1. Requêtes à préférences

Les requêtes qui expriment (intègrent) les préférences des utilisateurs et retournent les éléments désirées dans la réponse sont dites requêtes à préférences.

1.2. La notion de préférence

Les préférences dans le monde réel se présentent sous des formes très différentes comme tout le monde est courant. Cependant, une attention examen de leur nature varient révèle qu'elles partagent un principe fondamental commun. Examiner les domaines de la vie quotidienne avec son abondance de préférences qui peuvent provenir d'un sentiment subjectif ou d'autres intuitives

1.3. Requêtes à préférences

Les requêtes qui expriment (intègrent) les préférences des utilisateurs et retournent les éléments désirées dans la réponse sont dites requêtes à préférences.

1.4. La notion de préférence

Les préférences dans le monde réel se présentent sous des formes très différentes comme tout le monde est courant. Cependant, une attention examen de leur nature varient révèle qu'elles partagent un principe fondamental commun. Examiner les domaines de la vie quotidienne avec son abondance de préférences qui peuvent provenir d'un sentiment subjectif ou d'autres intuitives influences. Dans ce cadre familier, il s'avère que les gens expriment leurs souhaits fréquemment dans des termes comme « ***je préfère X que Y*** » **[KIE 02]**.

1.5. Relation de préférence

Une relation de préférence consiste à spécifier dans quelle mesure telle ou telle réponse est plus intéressante que telle autre, l'intérêt majeur est de trouver les meilleures réponses dominantes plutôt que toutes les réponses. On peut définir les préférences comme une relation binaire entre les n-uplets présents dans la réponse à une requête adressée à une base de données

1.5.1. Définition

Soit une relation de schéma R (A1, .., Ak) tel que Ui, $1 \leq i \leq k$, est le domaine de l'attribut Ai, une relation Ω est une relation de préférence sur R si elle constitue un sous-ensemble de (U1x…xUk)x(U1x…xUk) On dit que le n-uplet **t1** *domine (ou est préférable à)* le n-uplet **t2** dans le contexte de la relation Ω, **si** *t1* Ω *t2* **alors** *t1* est au moins aussi bon que t2 pour tous les attributs et encore meilleur que t2 dans un attribut, Deux données a et b pour lesquelsles ni a Ω b, ni b Ω a sont incomparables (indifférents).

1.5.2. Propriétés

- ✓ Irréfléxivité : $\forall x ; \; x * x$
- ✓ ii) Assymétrie : $\forall x, y ; \; x \Omega y \Rightarrow y * x$
- ✓ iii) Transitivité : $\forall x, y, z ; \; (x \Omega y \wedge y \Omega z) \Rightarrow x \Omega z$
- ✓ iv) Transitivité Négative : $\forall x, y, z ; \; (x * y \wedge y * z) \Rightarrow x * z$
- ✓ v) Connectivité : $\forall x, y ; \; x \Omega y \vee y \Omega x \vee x = y$

2. Approche quantitative [AWF 00] et [HKP 01]

Dans le cas d'une expression quantitative des préférences, un score, généralement dans $\mathbb{R}$, est défini pour chaque valeur du domaine d'attribut. Une valeur est préférée à une autre si elle a obtenu un meilleur score. Par exemple, un score peut être une distance par rapport à une valeur optimale. Plus la distance est faible, plus l'élément est préféré.

1.1. Définition

Dans le cas où les préférences élémentaires sont exprimées par des scores, chaque n-uplet*t i* est associé à un vecteur de scores ($\boldsymbol{S}_{i1},\ldots, \boldsymbol{S}_{ni}$), chaque $\mathbf{S}_{ji}$ correspond à l'évaluation d'une préférence sur le n-uplet $\boldsymbol{t}_{is}$. Deux hypothèses sont à considérer selon que les scores sont commensurables ou pas.

Un n-uplet $\boldsymbol{t}_i$ est préféré à un autre $\boldsymbol{t}_j$ si et seulement si :

$$\text{Score}(\boldsymbol{t}_i) > \text{Score}(\boldsymbol{t}_j) \Leftrightarrow \boldsymbol{f}(\boldsymbol{s}_1^i,\ldots, \boldsymbol{s}_n^i) > \boldsymbol{f}(\boldsymbol{s}_1^j,\ldots,\boldsymbol{s}_n^j).$$

Où $\boldsymbol{f}$ est une fonction d'agrégation qui permet de calculer un score général pour chacun des n-uplet.

2.1 Top_k queries [CHE 05]

Les « **Top K Queries** » font un bon exemple de cette approche, Les requêtes top-k offrent la possibilité aux utilisateurs de qualifier les résultats d'une requête grâce à une fonction de scores. Cette fonction fournie par l'utilisateur permet de trier les résultats par rapport à ses préférences individuelles. Les requêtes top-k permettent également à l'utilisateur de limiter le nombre de résultats que le système doit lui retourner. Cela évite de submerger l'utilisateur par un grand nombre de résultats. Ainsi, les résultats retournés à l'utilisateur sont les meilleurs par rapport a la fonction de score.

2.1.1 Principe

Top k Queries = trouver les k objets qui ont les scores globaux les plus élevées

```
SELECT * FROM r1, r2, r3 WHERE c1 AND c2 | BY  p1, p2, p3 LIMIT
                         AND c3          |     k
```

Phase (1) **Phase (2) (3)**

Où r1, r2, r3 sont des relations donnés ; c1, c2, c3 sont des conditions booléennes (conditions simples sur une ou plusieurs attributs, ou les deux) et p1, p2, p3 présente les attributs exprimant les préférences de l'utilisateur comme numériques.

Phase1 : sélection des n-uplets vérifiant les conditions de la clause WHERE (condition booléenne).

Phase 2 : l'ordonnancement de ces n-uplet suivant une fonction d'ordonnancement, et la délivrance d'un nombre k de résultats souhaités (limitation des réponses).

Exemple

Soit la relation *R* de la figure suivante décrivant des voitures par leur marque, leur prix, et leur nombre de chevaux. La moins chère voiture décrite par un n-uplet est calculé par la fonction suivante:

$f(t) = t.prix + t.Nb\_Ch$

Id	Marque	Prix	Nb_Ch
#1	Bmw	30000	90
#2	Audi	15100	92
#3	Vw	15000	91

Figure I.1 : Relation voiture.

La requête « chercher les deux meilleurs réponses (k = 2) à : trouver les voitures moins chères» utilise la fonction *f* comme fonction d'ordonnancement.

Le résultat est donné comme suit :

Le résultat est donné comme suit : *f(t* 3*)*=15091, *f(t* 2*)*=15191 et *f(t* 1*)=30091* où les n-uplets *t* 1, *t* 2 et *t* 3 sont respectivement associés aux clés #1, #2, #3.

L'ordonnancement des n-uplets est alors : *t* 3, *t* 2puis *t* 1 et seuls *t* 3, *t* 2 sont retournés.

```
1. ARRAY result [1..N]  first N items of Data Set
2.  sort result in ascending order resp. S
3. worstScore = S ( result [k] )
4. foreach new data item d
5. if S( d ) < worstScore
6.  result[k] = d
7. sort result in ascending order resp. S
8. worstScore = S ( result [k] )
9. return result
```

Figure I.2 : Algorithme top-k Queries.

3. Approche qualitative

Dans certains cas, il est difficile pour l'utilisateur d'exprimer ses préférences au moyen de scores sur les valeurs d'attributs. Il est plus facile de comparer les valeurs d'attributs les unes par rapport aux autres. Dans ce contexte, la commensurabilité n'a pas d'objet. Lorsque l'on considère plusieurs préférences atomiques, la relation de préférence entre deux n-uplets consiste alors à comparer les valeurs de chaque attribut les unes par rapport aux autres.

3.1. Ordre de Pareto

On souhaite comparer deux vecteurs de valeurs, ***v*** et ***u*** tel que ***v*** **= (*v*1 ,…,*v n*),** ***u*** **= (*u*1 ,…,*u n*)**. L'ordre de Pareto est défini par la formule suivante :

v >Pareto ***u*** $\Leftrightarrow$ **(**$\forall$ **i *v i*** $\geq$ ***u i*** **et** $\exists$ **j *v j* > *u j*).**

La préférence « une voiture de couleur verte et qui consomme peu », est définie sur les attributs couleur et consommation. Chaque n-uplet est associé à un vecteur composé d'une couleur et d'une consommation. Par exemple ***v* 1** est associé à ***(verte, 8)***, ***v* 2** est associé à ***(verte, 9)*** et ***v* 3** est associé à ***(noire, 7)***. L'ordonnancement des vecteurs se base sur les relations de préférences atomiques et l'on peut utiliser une relation d'ordre partiel définie par l'ordre de Pareto ce qui interdit de prendre en considération des phénomènes de compensation entre différentes préférences contrairement à l'approche par score. Pour l'exemple précédent ***v* 1** est préféré à ***v* 2** et ***v* 3** ne peut pas être comparé à ***v* 1** ou ***v* 2**.

Parmi les bons exemples qui représentent cette approche on trouve:

Les «**Requêtes skyline**», «**Préférence SQL**» et «**l'approche logique**».

Note : L'approche qualitative est plus générale que l'approche quantitative car elle peut définir les relations de préférences sur des fonctions numériques si elles sont données explicitement, tandis que toutes les relations de préférences ne peuvent pas être capturées par des fonctions numériques.

4. Requêtes skyline

Le concept de requête Skyline a été introduit pour formuler des recherches multicritères. Lorsque ces critères sont conflictuels, l'opérateur skyline retourne les meilleurs compromis entre plusieurs critères. Ces résultats sont incomparables entre eux. Si les Skylines répondent efficacement au problème de recherche pour une base de données volumineuse et un nombre limité de critères, ils sont inadaptés lorsque le nombre de critères augmente. En effet, la probabilité d'objets incomparables augmente avec le nombre de dimensions et entraine une

surabondance de résultats conflictuels. La théorie de la décision apporte justement des solutions à la prise de décision en présence de choix conflictuels.

4.1. Principe de skyline

Etant donné un ensemble de points P dans un espace multidimensionnel, la requête skyline, notée S(P) retourne le sous-ensemble de P qui est constitué de l'ensemble des points qui ne sont dominés par aucun autre point : On dit que p1 domine p2 si (1) pour chaque dimension, la coordonnée de p1 est ou moins aussi bon à celle de p2 et (2) pour au moins une dimension, la coordonnée de p1 est meilleure à celle de p2. Par exemple, on cherche les restaurants les moins loin et les moins chers par rapport à un point donné. L'ensemble des restaurants représente l'ensemble P dont les coordonnées sont (Distance; Prix).

Exemple

Nous avons utilisé l'exemple de camera suivant pour exprimer le principe de skyline. Considérons l'exemple d'une personne désirant acheter une caméra. L'ensemble des caméras est décrit dans le tableau :

Modèle	Prix	Qualité (Méga Pixel)
S1	16500	7,2
S2	27500	8,1
S3	27000	8,1
S4	36900	13,6
S5	63500	10,1
S6	30000	13,6
S7	29000	9,5
S8	39500	8,2
S9	41200	14,7
S10	53700	14,7
S11	30000	12,3
S12	19000	7,1

Tableau I.1 : Exemple des caméras avec leurs qualités et prix.

La requête Skyline Q Considéré concerne la recherche des caméras qui sont moins chères (Min pour le Prix) et ayant la meilleur qualité (Max pour qualité).

Q = **SKYLINE OF** prix **Min**, qualité **Max**

Un utilisateur intéressé par le prix et la qualité par rapport à la caméra, peut employer une requête skyline pour rechercher les caméras qui ont une bonne pour qualité et ont le plus bas prix. Le résultat de cette requête inclura S1, S9 et le skyline de cet ensemble est S= {S1, S3, S6, S7, S9}. Ces caméras ne sont pas dominées par d'autres caméras. Par conséquent, ils seront intéressants à l'utilisateur. S2 est dominé par S3 alors que S3 a une bonne qualité et a aussi un bas prix. De même, S4, S5, S8 et S11 sont dominés par S6 et S10 est dominé par S9 et de même S12 est dominé par S1. En éliminant les caméras qui sont complètement dominées, le processus décisionnel peut être facilité à l'utilisateur et c'est à lui de peser leur préférence selon le prix et la qualité.

4.2. Fondements mathématique

Dans la littérature le calcul de skyline est connu comme le problème de vecteur maximum. Parfois il est considéré comme l'optimum de Pareto, mais contrairement à une distribution de probabilité de Pareto qui peut être exprimé par une formule close, le skyline ne peut pas être exprimé par une formule.

Note : le mot vecteur et point sont synonymes, et nous avons la plupart du temps d'utiliser le point au lieu du vecteur.

4.2.1 Problème de vecteur maximal

Le problème de vecteur maximum a été discuté dans **[KLP 75]**. Il décrit le problème de trouver le maximum d'un ensemble de vecteurs. Quel est le maximum d'une série de vecteurs ou de points ? Par exemple :

A(5,5) et B(2,2)

Si le point **A** est plus grand dans toutes les dimensions que le point **B** alors **A** est clairement le maximum de l'ensemble de point **{A, B}.** Mais que devient-il si **A** est plus grand dans toutes les dimensions que **B** l'exception d'une dimension ?

A(5,2) et B(2,3)

Ensuite, une mesure plus sophistiqués pour les points de comparaison doit être appliquée. La mesure est parle de dominance et a été défini dans **[PRS 85].**

4.3 Comparaisons des points

Comparer deux points pour des caractéristiques du skyline est un peut plus compliqué qu'une comparaison « normale ».chacune des dimensions doit être considéré et selon la comparaison de la dimension précédente, la fonction complète de comparaison de skyline est effectuée ou doit continuer les dimensions restantes.

Il ya quatre états que la comparaison de skyline peut retourner :

- **Egale** : deux points ont les mêmes valeurs dans chaque dimension, c-a-d, pour deux points **p** et **q** ayant la même dimensionnalité **d** les prises suivantes :

$$\forall\, i,\ 1 \leq i \leq d,\ p_i = q_i.$$

- **Plus grand** : deux points ont la même valeur dans chaque dimension, mais un point a une plus grande valeur dans au moins une dimension, c-a-d, pour deux point **p** et **q** ayant la même dimensionnalité **d** et **p** plus grand que **q**, les prises suivantes :

$$\forall\, i,\ 1 \leq i \leq d,\ p_i = q_i \wedge p_k > q_k,\ 1 \leq k \leq d.$$

- **Moins** : deux points ont la même valeur dans chaque dimension, mais un point a une plus petite valeur dans au moins une dimension, c-a-d, pour deux points **p** et **q** ayant la même dimensionnalité **d** et **p** moins que **q** les prises suivantes :

$$\forall\, i,\ 1 \leq i \leq d,\ p_i = q_i \wedge p_k < q_k,\ 1 \leq k \leq d.$$

- **Incomparable**

Deux points ont la même valeur dans chaque dimension, mais un point a une plus grande valeur dans au moins une dimension et une plus petite valeur dans au moins une autre dimension, c-a-d, pour deux point **p** et **q** ayant la même dimensionnalité **d** et **p** incomparable à **q** les prises suivantes :

$$\forall\, i,\ 1 \leq i \leq d,\ p_i = q_i \wedge p_k > q_k,\ 1 \leq k \leq d \wedge p_1 < q_1,\ 1 \leq 1 \leq d.$$

Comme on peut le voir facilement, pour deux points incomparables, il suffit que la première doive être plus (moins) et la seconde dimension doit être moins (plus). Dans ces cas, il n'est pas nécessaire d'examiner les dimensions restantes et les comparaisons skyline sont faites avec les deux points étant incomparable. Ce n'est pas vrai pour le moins, plus, ou l'égalité des résultats.

4.4 Principe de dominance

Soit un ensemble D d'objets, et n fonctions de score S = {s_1, s_2, …, s_n} et soient d1 et d2 deux éléments de D, une relation ***domine*** (dont le symbole est $\mathbf{\Omega_S}$) peut être définie comme suit:

d1 ***domine*** ($\mathbf{\Omega_S}$) d2 ssi

(i) $\forall$ s $\in$ S, s(d1) $\geq$ s(d2), et

(ii) $\exists$ s $\in$ S, s(d1) > s(d2).

C'est-à-dire, *d1* ***domine*** *d2* si *d1* est au moins aussi bon que *d2* pour toutes les fonctions de score et meilleur que *d2* dans une fonction de score, deux objets a1 et a2 pour lesquels ni a1 $\mathbf{\Omega_S}$ a2, ni a2 $\mathbf{\Omega_S}$ a1 sont appelés incomparables (indifférents).

4.4.1 Propriétés de la relation de dominance (Ω_S)

- ✓ ***Transitive***

La transitivité signifie que si un point domine un autre point et ce point domine un troisième point alors le troisième point est également dominé par le premier point.

Considérons trois points, a, b et c, le point a domine b et b domine c alors a domine également c.

- ✓ ***Asymétrique***

La relation de dominance est asymétrique, veut dire que si a domine b alors b est dominé par a (b ne domine pas a).C'est clair que a Ω_S b et b Ω_S a conduit à une contradiction d'un côté $\forall$ s $\in$ S, s(a) $\geq$ s(b) et de l'autre côté $\exists$ s $\in$ S, s(b) > s(a).

- ✓ ***Irreflexive*** **:**

Aucune fonction de score **s** ne peut exister pour s(a) > s(a) et la deuxième condition de principe de dominance ne peut être vérifié**.**

4.5 Définition de Skyline

L'ensemble des éléments maximums de D par rapport à Ω_S est appelé skyline de D par rapport à S (dont le symbole est $\boldsymbol{S}_S$) et il est défini comme suit:

S_S (D) = {d $\in$ D / $\neq$ d1 $\in$ D, d1 Ω_S d}

S_S (D) : contient les éléments qui ne sont dominés par aucun élément sur toutes les dimensions.

4.5.1 Les propriétés de skyline

- **Nombre de point de skyline**

Il ya deux propriétés importantes sur le nombre de points de skyline (‘’taille skyline’’) :

La taille de skyline augmente avec la dimensionnalité croissante.

La taille de skyline augmente également avec l’augmentation du nombre de points dans la base de données.

- **Traitement des points en double**

Par exemple, si un algorithme, en quelque sorte, produit un sous-ensemble de skyline, et encore, ce sous-ensemble contient quelque point doublons donc un algorithme de skyline ne trouvera pas ces doublons. Il permettra de caractériser tous les doublons de points skyline.

- **Aucune suffisance sur certaines dimensions**

Une autre propriété de skyline d’un ensemble **D** est pour toute fonction de score monotone

S:D→$\mathbb{R}$

Si **p** $\in$ **D** maximise (minimise) cette fonction **S**, alors **p** appartient au skyline (votre objet préféré est toujours dans le skyline).

- **Transitivité**

La transitivité signifie que si un point domine un autre point et ce point domaine le troisième point alors le troisième point est également dominé par le premier point. Considéré trois points **p**, **q** et **r**, le point **p** domaine **q** et **q** domaine **r** alors **p** domaine aussi **r**, qui est :

$$p \prec q \wedge q \prec r \Rightarrow p \prec r.$$

5. Expression des requêtes skyline

Borzsonyi, Kossmann et Stocker **[**BKS01**]** étaient les premiers qui ont discuté l'idée du problème de vecteur maximum dans le contexte de base de données. Ils ont proposé une extension de SQL. Le rapport SELECT du SQL est prolongé par une clause **SKYLINE OF** facultatif décrivant les attributs (dimensions) qui devraient être considérés dans le calcul de skyline.

SELECT ……. **FROM**……..**WHERE**

GROUP BY ……. **HAVING**…….

SKYLINE OF [DISTINCT] d1 [MIN | MAX | DIFF],…, dm [MIN | MAX | DIFF]

ORDER BY…

- *d1,..., dm* sont les dimensions de skyline (par exemple: prix, distance,…).
- *[MIN | MAX | DIFF]* spécifie si la valeur dans cette dimension doit être minimisée ou maximisée ou simplement indifférente.
- *DISTINCT* est facultatif et indique comment traiter les duplications (points identiques). Si p=q pour tous i=1…n alors p et q sont incomparables et peuvent tous les deux faire partie de skyline si aucun DISTINCT ni indiqué, sinon p ou q est maintenu.
- *SKYLINE OF* : la clause SKYLINE OF décrit pour chaque dimension si elle doit être maximisée, minimisée ou être indifférente. Cette clause est exécutée après le: **SELECT** ……. **FROM**……..**WHERE….GROUP BY…..HAVING** de la requête.

La clause SKYLINE OF fait sortir tous les n-uplets intéressants (n-uplets qui ne sont pas dominés par tout autre n-uplet).

Par exemple soient deux points p et q de dimension n

$p = (p1, \ldots, pk, pk+1, \ldots, pl, pl+1, \ldots, pm,)$

$q = (q1, \ldots, qk, qk+1, \ldots, ql, ql+1, \ldots, qm,)$

p domine q pour la requête skyline : *SKYLINE OF dim1 min, ..., dimk min*

dimk+1 max, ..., diml max

dim l+1 diff, ..., dimm diff

Si les conditions suivantes sont satisfaites :

$pi \leq qi \quad \forall\, i = 1, \ldots, k$

$pi \geq qi \quad \forall\, i = k+1, \ldots, l$

$pi = qi \quad \forall\, i = l+1, \ldots, m$

Remarque: Les autres dimensions dim **m+1**……dim **n** sont non pertinentes pour le calcul de skyline.

6. Conclusion

Les préférences sont principalement employées pour filtrer et personnaliser l'information recherchée par les utilisateurs. Nous avons présenté deux familles d'approches pour les représenter : quantitative et qualitative.

Dans l'approche quantitative des mécanismes de scores numériques commensurables ou non sont utilisés pour représenter les préférences. Un ordre total est établi sur l'ensemble des résultats lorsque les préférences sont commensurables et un ordre partiel est obtenu si les préférences sont non commensurables. Dans l'approche qualitative les préférences sont spécifiées par des relations binaires de préférences et dans la plupart des cas, un ordre partiel est obtenu sur les n-uplets.

L'approche qualitative est plus générale que l'approche quantitative, elle permet de définir les préférences qualitativement. Le chapitre suivant, présente l'algorithme BNL en détails avec un exemple pour bien illustrer son principe de fonctionnement.

CHAPITRE 2

Approches d'Evaluation de Skyline dans un Contexte Centralisé

L'efficacité des requêtes de type « skyline » a été largement étudiée dans la communauté des bases de données. « Les requêtes skyline qui retournent des objets qui sont meilleure ou égale dans toutes les dimensions et mieux dans au moins une dimension, sont utiles à la prise de nombreux décision et gérer des applications » **[LPY 07].** Par exemple, une requête classique suffit à un touriste pour rechercher un hôtel bon marché. Cette recherche devient plus complexe lorsqu'il formule plusieurs critères. Il faut alors rechercher le meilleur compromis entre les différentes solutions. Les requêtes de type Skyline permettent de formuler ces requêtes multicritères et d'obtenir par exemple les dix meilleurs hôtels, bon marché et à proximité de la plage. Dans ce chapitre on va présenter les différents algorithmes et approches de calcul de skyline dans un contexte centralisé.

1. Algorithme standard

Le calcul de skyline par l'algorithme standard est simple, il suffit simplement de comparer chaque n-uplet à tous les autres n-uplets de la BD sauf à lui-même, en respectant les propriétés de la relation de dominance à savoir, deux n-uplets identiques font partie de skyline tous les deux.

Cet algorithme est le plus facile et il est très performant dans le cas de petites bases de données (ou petites dimensions), mais il n'est pas possible de comparer chaque élément à tous les autres dans des applications modernes qui utilisent des bases de données astronomiques (volumineuses).

Le skyline de l'ensemble de données du **Tableau I.1** est calculé comme suit :

	Modèle	Prix	Qualité (Méga Pixel)
	S1	16500	7,2
éliminé par S3{	S2	27500	8,1
	S3	27000	8,1
éliminé par S6{	S4	36900	13,6
éliminé par S4{	S5	63500	10,1
	S6	30000	13,6
	S7	29000	9,5
éliminé par S6{	S8	39500	8,2
	S9	41200	14,7
éliminé par S9{	S10	53700	14,7
éliminé par S6{	S11	30000	12,3
éliminé par S1{	S12	19000	7,1

Tableau II.1 : Algorithme standard

2. Algorithme BNL (Block-Nested-Loops)

L'idée directrice de cet algorithme **[BKS 01]** est de limiter le nombre d'E/S effectuée rechargeant en mémoire un ensemble de tuples candidats au Skyline appelé « fenêtre ». Les accès disque sont faites par bloc, les comparaisons ont lieu directement en mémoire centrale et sont donc bien plus rapides. Cependant, cette fenêtre a une taille limitée. De plus, comme la taille du Skyline est du même ordre de grandeur que la relation d'entrée, il est vraisemblable que la fenêtre ne puisse pas accueillir tous les candidats. C'est pour cela que l'algorithme gère, en plus de la fenêtre, un fichier temporaire stocké en mémoire secondaire dans lequel sont écrits tous les candidats non considère (limité d'espace). Ils seront traités lors d'une prochaine itération. Il ya trois cas :

- **t** est dominé par un tuple de la fenêtre **t** est alors directement écarte et ne sera plus pris en compte pour le reste du calcul : il est dominé et ne fera donc jamais partie du Skyline. On élimine **t** dès la première occurrence de ce cas et il n'est pas nécessaire de poursuivre la comparaison avec les autres tuples de la fenêtre.
- **t** domine un ou plusieurs tuples de la fenêtre Les tuples de la fenêtre domines par **t** sont écartes et ne seront plus pris en compte pour les itérations futures. **t** est insérée dans la fenêtre sans problème, puisqu'il y a au moins une place libre.

- **t** est incomparable avec l'ensemble des tuples de la fenêtre C'est le cas le plus problématique : **t** doit être ajouté à la fenêtre mais il n'a élimine aucun tuple sur son passage et il se peut que celle-ci soit pleine. Si ce n'est pas le cas, **t** est ajoutée normalement à la fenêtre. Sinon, **t** est mis de coté dans le fichier temporaire et sera examine à nouveau au cours de la prochaine itération de l'algorithme.

L'algorithme de BNL dans notre exemple est affiché dans le tableau

L'algorithme accorde une attention particulière au fait que les points de Skyline candidat, la fenêtre, ne rentrent pas dans la mémoire principale de plus (indiquer par un drapeau ordinateur memAvailable. Puis le nouveau point de lecture est échangé dans un fichier temporaire sur le disque. Le moment où un point a été échangé à fichier temporaire est désigné par un timestamp (timestampOut).

Les points candidats sont vérifiées pour tous les points qui ont déjà été identifiés comme des points Skyline, ces points skyline sont tous les points qui ont été comparés à tous les points dans l'ensemble de données. Un point candidats de Skyline doit être un point de skyline si la condition suivante est vérifie :

timestampIn < timestampout courant.

Quand un nouveau point lu à partir du fichier temporaire est affecté du timestampout courant. Il est comparé à tous les points de la fenêtre Si elle est dominée par un point de la fenêtre, il est supprimé et l'algorithme se poursuit avec le point suivant du fichier temporaire. Si elle domine un ou plusieurs points dans la fenêtre, alors tous les points dominés sont supprimés de la fenêtre et insérer ce point dans la fenêtre.

Après l'algorithme se fait avec l'ensemble de données, il vérifie s'il ya des points, le fichier temporaire est chargé et l'algorithme se poursuit avec ces points. A la fin tous les points qui sont encore dans la fenêtre sont copiés dans le Skyline.

Exemple : en prenant l'exemple des caméras traite précédemment dans le **chapitre I**.

La taille de la fenêtre est fixée à quatre

étape	lire	Les Skylines candidats	tsIn	tsOut	temporaire	skyline
Charger D			0	0		
1	S1	S1	1	0		
2	S2	S1, S2	2	0		
3	S3	S1, S3	3	0		
4	S4	S1, S3, S4	4	0		
5	S5	S1, S3, S4	5	0		
6	S6	S1, S3, S6	6	0		
7	S7	S1, S3, S6, S7	7	0		
8	S8	S1, S3, S6, S7	8	0		
9	S9	S1, S3, S6, S7	9	1	S9	
10	S10	S1, S3, S6, S7	10	1	S9	
11	S11	S1, S3, S6, S7	11	1	S9	
12	S12	S1, S3, S6, S7	12	2	S9	
Charge temporaire	-	-	0	0	-	-
13	-	-	1	0	-	S1, S3, S6, S7
14	S9	-	2	0	-	S1, S3, S6, S7, S9

Tableau II.2 : Les étapes d'algorithme BNL.

Le tableau 1 représente la description d'algorithme BNL comme suit :

Charger les points de BD

Initialement la fenêtre est vide.

Le premier point (S1) est inséré dans la fenêtre (timestampIn = 1)

Le point suivant (S2) est insérer dans la fenêtre parce que il est incomparable avec le point (S1) (timestampIn = 2)

Le point (S3) domine le point (S2) donc supprimer le point (S2) et ajouter le point (S3) dans la fenêtre (timestampIn = 3)

Le point (S4) est insérer dans la fenêtre puisqu'il incomparable avec le point (S3) (timestampIn = 4)

Le point (S5) est éliminé car il est dominé par le point (S4) (timestampIn = 5)

Le point (S6) domine le point (S4) donc supprimer le point (S4) et ajouter le point (S6) dans la fenêtre (timestampIn = 6)

Le point (S7) est insérer dans la fenêtre puisqu'il incomparable avec le point (S6) (timestampIn = 7)

Le point (S8) est éliminé car il est dominé par le point (S7) (timestampIn = 8)

Le point (S9) est incomparable avec les point skyline candidats de la fenêtre et puisque la fenêtre est pleine dans ce cas le point (S9) est inséré dans la file temporaire avec timestampOut = 1)

Le point (S10) est éliminé car il est dominé par le point (S9) (timestampIn = 10)

Le point (S11) est éliminé car il est dominé par le point (S6) (timestampIn = 11)

Le point (S12) est éliminé car il est dominé par le point (S1) (timestampIn = 12)

Charger les points de la file temporaire

A la fin de cette itération les points candidats de skyline sont S1, S3, S6, S7 et la file temporaire contient S9.

Après les points de la fenêtre qui ont comparés a tous les points de l'ensemble (qui ont un timestamp inferieur a 9), sont sorties (insérés dans le skyline) dans ce cas les points S1, S3, S6, S7 sont insérés dans le skyline timestampIn < timestampOut) et l'algorithme continue le traitement des éléments de la file temporaire

Les points S9 est incomparables avec les skyline

Le résultat de skyline est S= {S1, S3, S6, S7, S9}.

Pour gagner de temps et réduire le nombre de comparaisons on trouve dans **[BKS 01]** deux variantes à cet algorithme:

- considérer la fenêtre comme une liste à organisation automatique
- remplacement des n-uplets dans la fenêtre

2.1 Les avantages et les inconvénients de BNL

✓ ***Avantage***

- BNL a une large applicabilité, car il peut être utilisé pour toute dimension, sans indexation ou de tri du fichier de données.

✓ ***Inconvénients***

- Ses problèmes principale sont le recours à la mémoire principale (a petite mémoire peut conduire à de nombreuses itérations).
- Ses insuffisances de traitement en ligne (il faut lire l'ensemble des données fichier avant qu'il ne retourne en premier point skyline).

3. Algorithme Bitmap

Cet algorithme a d'abord été publié dans **[TAN 01].** Quelque amélioration a été publiée dans **[EOT 03].** Nous expliquons l'algorithme original de Bitmap en l'appliquant à l'exemple précédant.

3.1. Description de l'algorithme

Les algorithmes précédents avaient à regarder les points de l'ensemble de données au moins une fois afin de déterminer tous les points skyline (l'algorithme requis multiples passe sur l'ensemble de données).

Dans cette section, nous décrivons un algorithme qui requiert exactement une passe sur l'ensemble de données pour calculer tous les points de skyline. Cet algorithme utilise des bitmaps pour coder toutes les informations nécessaires pour décider si un point fait partie de skyline ou non.

Un point $p = (p_1, p_2,\ldots, p_d)$ avec d étant le nombre de dimensions est mappé à un vecteur de bits. Ce vecteur de bits contient des informations sur le rang de chaque valeur $p_1, p_2,\ldots, p_d$ par rapport à d'autres valeurs de la même dimension. La longueur du vecteur de bits est déterminée par le nombre de valeurs distinctes sur toutes les dimensions. Soit k_i le nombre de valeurs distinctes dans l'ième dimension puis m est la somme de tous les k_i.

Les bitmaps codent le grade où la position de la valeur du point dans chaque dimension. Supposons que p_i, la valeur du point p dans la i^{eme} dimension, est j_i^{eme} plus petite valeur en dimension i, ce qui signifie que les valeurs de j-1 dans la dimension i sont plus petit que cette valeur. P_i de cette valeur va être encodé avec k_i bits où le ki-j$_i$ + 1 bits les plus significatifs sont définies sur 1 et celles restantes sont définies sur 0.

En accédant à ce point, nous voulons décider si elle fait partie de skyline ou non. Dans ce cas l'algorithme crée deux bits silice $a_1,a_2,\ldots a_d$ pour chaque valeur $p_1, p_2, \ldots, p_d$, le résultat d'un ET logique entre ces bit silice est notée par A

$$A = a_1 \wedge a_2 \wedge \ldots \wedge a_d$$

Ces bits silice contenant D bits, une entrée pour chaque point (D est le nombre de points dans la BD). Les bits-silice sont obtenus par «lire» les images bitmap dans le tableau qui représente la structure bitmap, La colonne est déterminée par le point qui doit être vérifiée pour le Skyline. Il est la colonne moins significative qui contient un 1.

Le bit-slice A a la propriété que le bit «n» est mis à 1 si le $n^{ième}$ point a des valeurs de dimension inférieure ou égale à la valeur de la dimension le point d'être vérifiée pour les Skylines

Maintenant, l'algorithme crée un deuxième bit-slice $b_1, b_2, \ldots b_d$ pour chaque dimensions. Ce deuxième bit-slice est le bit précédent de bit-slice a. Les résultats d'une opération OU est

$$B = b_1 \vee b_2 \vee \ldots \vee b_d$$

Le Bit-slice B a la propriété que le bit n est mis à 1 si le $n^{ième}$ point a une certaine dimension des valeurs inférieures à la valeur de la dimension correspondante du point à son tour. S'il n'ya pas de bit précédent d'un bit silice le résultat de cette opération est fixé à 0

La dernière opération est de faire un ET logique entre A et B :

$$C = A \wedge B$$

Suivant le résultat dans c, s'il un 1 a la position i alors la i^{ime} point est aussi bon que p dans toutes les dimensions et meilleur dans au moins une dimension (P_i domine p), et si le résultat contient que ; des 0 alors p est un point skyline.

Exemple

Pour notre exemple on va traiter l'exemple du chapitre I. Le **tableau I.1** montre toutes les valeurs distinctes et leur rang de notre exemple. Le nombre de valeurs distinctes dans la première dimension est $k1 = 11$et le nombre de valeurs distinctes dans la seconde dimension est $k2 = 9$. Cela signifie $m = k1+k2=11+9= 20$.

Dans le tableau, le prix de (S1) est à 1, qui est la $1^{ère}$ plus petite valeur de tous les points, la présentation bitmap contiendra 11-1 + 1 = 11 pour tous les bits, il est représenté par 11111111110. Le point (S1) à une qualité de 7,2 qui est la $8^{ème}$ plus grande valeur de tous les points. Cela signifie que le 9-8+1 = 2 bits plus significatif de l'image bitmap représentant dimension 2 aura la valeur 1 et les bits restants seront définis sur 0, 110000000.

Pour le point (S2), c'est la $4^{ème}$ plus petite valeur pour la dimension prix (11-4+1 = 8 bits sont 1 : 11111111000) et la $7^{ème}$ plus grande valeur pour la dimension qualité (9-1+1 = 9 bits sont 1 : 111111111).

En fait la même opération avec tous les autres points

Le tableau suivant montre le codage de tous les points :

caméra	prix	Qualité	Dimension1	Dimension2
S1	1	8	11111111111	110000000
S2	4	7	11111111000	111000000
S3	3	7	11111111100	111000000
S4	7	2	11111000000	111111110
S5	11	4	10000000000	111111000
S6	6	2	11111100000	111111110
S7	5	5	11111110000	111110000
S8	8	6	11110000000	111100000
S9	9	1	11100000000	111111111
S10	10	1	11000000000	111111111
S11	6	3	11111100000	111111100
S12	2	9	11111111110	100000000

Tableau II.3 : L'approche Bitmap.

En accédant à tous ces points, nous voulons décider s'ils font partie de skyline ou non.

On commence avec le point (S1), l'algorithme crée deux bits silice, c'est pour a1 la 10ème colonne, et la 2ème colonne pour a2, a1 = 100000000000 et a2 = 111111111110.

Les bits silice contenant 12 bits (12 = le nombre de point de la BD)

Le résultat d'une opération ET est

$A = a_1 \wedge a_2 = 100000000000 \wedge 111111111110 = 100000000000$

Dans ce cas le point (S1) est meilleur que les autres points pour la dimension prix.

Maintenant, l'algorithme crée un deuxième bits-silice pour chaque dimension.

Puisqu'il n'y pas de bit précédent, le résultat de OU opération est fixé à 0

$B = b_1 \vee b_2 = 00000000000 \vee 011111111110 = 011111111110$

Cela signifie que les points à la position 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, (les points S2, S3, S4, S5, S6, S7, S8, S9, S10, S11 respectivement) sont meilleurs que S1 pour la dimension qualité.

L'opération finale détermine la propriété skyline de (S1)

$C = A \wedge B = 100000000000 \wedge 011111111110 = 000000000000$

Donc le point (S1) est appartient à skyline

On passe par ces opérations avec tous les points de la BD.

Pour le point S2

$A = a_1 \wedge a_2 = 111000000001 \wedge 011111111110 = 011000000000$

$B = b_1 \vee b_2 = 101000000001 \vee 000111111110 = 101111111111$

$C = A \wedge B = 011000000000 \wedge 101111111111 = 001000000000$

Donc le point (S2) est dominé par (S3)

Pour le point S3

$A = a_1 \wedge a_2 = 101000000001 \wedge 011111111110 = 00100000000$

$B = b_1 \vee b_2 = 100000000001 \vee 000111111110 = 100111111111$

$C = A \wedge B = 001000000000 \wedge 100111111111 = 000000000000$

Donc (S3) est appartient à skyline

Pour le point S4

$A = a_1 \wedge a_2 = 1111O1100011 \wedge 000101001100 = 000101000000$

$B = b_1 \vee b_2 = 111001100011 \vee 000000001100 = 111001101111$

$C = A \wedge B = 0001010000000 \wedge 111001101111 = 0000100000000$

Donc (S4) est dominé par (S5)

Ainsi de suite en fait la même opération avec tous les points. Finalement le skyline obtenue est S= {S1, S3, S6, S7, S9}.

4. Algorithme BBS (Branch-and-Bound Skyline Algorithm)

BBS, est basé sur des recherches de plus proche voisins. Nous utilisons R-arbre en raison de leur simplicité et leur popularité. Les mêmes concepts peuvent être appliqués avec d'autres méthodes d'accès multidimensionnels pour haute-espaces de dimension, où la performance de R-arbre est connue pour se détériorer. Selon **[KRR 02],** la plupart des applications implique jusqu'à cinq dimension pour lesquels R-arbres sont encor efficaces. Pour la discussion qui suit, nous utilisons l'ensemble des points de données en *2D* de la **Figure II.1**, organisés dans R-arbre de la **Figure II.2** avec une capacité de nœud = 3. Une entrée intermédiaire e_i correspond au minimum rectangle englobant (*MBR*) d'un nœud Ni au niveau supérieur, alors qu'une entrée feuilles correspond à un point de données. Les distances sont calculées en fonction de norme *L1*,

que est, le *mindist* d'un point est égale à la différence de ses coordonnées (prix-qualité) et le *mindist* d'un MBR (entrée intermédiaire) est égale à la *mindist* de son point d'angle en haut à gauche.

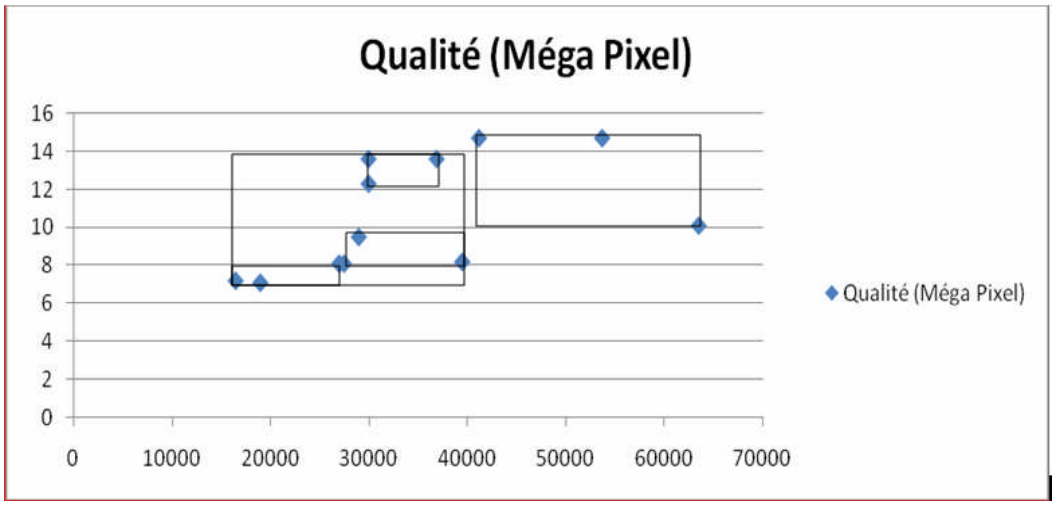


Figure II.1 : L'ensemble de point en 2D.

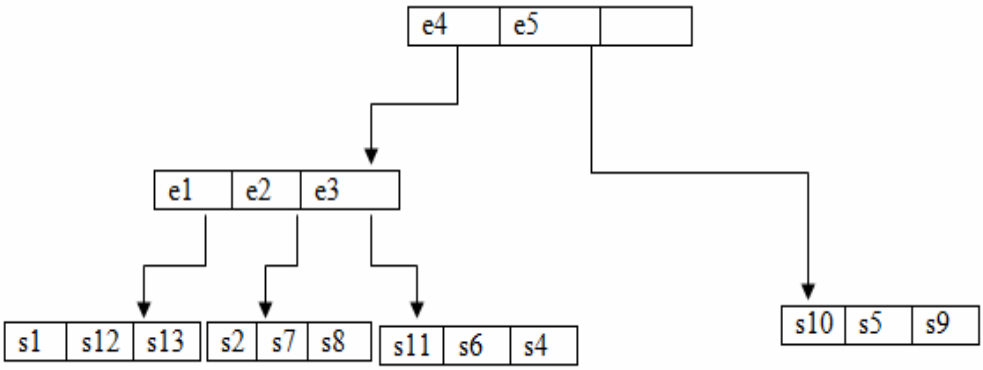


Figure II.2 : R-arbre.

Initialement, tous les fils du nœud racine *r* de l'arbre sont insérés dans un tas ordonné selon leurs distances *mindist*. Dans chaque itération, l'entré *e* qui a le *mindist* minimum est supprimée du tas et examinée contre les points de skyline courant. Si *e* est donné par un ou plusieurs points de skyline, *e* est éliminé. Autrement si *e* est un nœud de l'arbre, il est dépensé en insérant tous ses fils dans le tas s'ils ne sont pas dominés par des points de skyline courant. Si *e* est un point de données, il est inséré dans la liste des points de skyline comme un nouveau point,

et l'algorithme termine lorsque les tas est vide. Pour notre exemple l'algorithme commence par le nœud racine *r* de l'arbre et insère toutes les entées (e4, e5) dans le tas triés selon leurs *mindist*.

Ensuit, l'entrée avec le *mindist* minimum (e4) est "élargie". Ce processus d'expansion supprime l'entrée (e7) à partir du tas et insère ses enfants (e1, e2, e3).

L'entée élargie est la 1ère *mindist* minimum (e1), dans lequel les premiers les plus proche voisins (S1) et (S3) sont trouvés. Ces points (S1) et (S3) sont insérés à la liste des points skyline S.

L'entrée élargie suivante est (e2), dans le quel le plus proche voisin (S7) est trouvé. Ce point (S7) est inséré à la liste des points skyline S.

L'entée élargie suivante est (e3), dans le quel le plus proche voisin (S6) est trouvé, ce dernier est inséré à la liste skyline S.

La dernière entrée élargie est (e4), dans lequel le plus proche voisin (S9) est trouvé. Ce point (S9) est inséré à la liste des points skyline S.

Le tableau présente les identifiants et les *mindist* des entrées insérées dans le tas.

Action	Contenu du tas	Liste de skyline(S)
Racine R	<e5, 16491,9><e4, 41189,9>	ø
Expand e5	<e1, 16491,9><e2, 27490,5><e3, 29986,4><e4, 41185,3>	ø
Expand e1	< (s1, 16492.8), (s3, 26991,9)><e2, 27490,5><e3, 29986,4> <e4, 41185,3>	S1, S3
Expand e2	<s7, 28990,5><e3, 29986,4><e4, 41185,3>	S1, S3, S7
Expand e3	<s6, 29986,4><e4, 41185,3>	S1, S3, S7, S6
Expand e4	<s9, 41185,3>	S1, S3, S7, S6, S9
	Φ	S1, S3, S7, S6, S9

Tableau II.4 : Contenu du tas.

5. Discussion

Cette section présente une comparaison des algorithmes précédents. Pour cela, on va se baser sur les critères présentés dans **[KRR 02]** et **[HAC 99]** pour évaluer le comportement et l'applicabilité des algorithmes de skyline en ligne. Un algorithme est dit en ligne (Online Algorithm) s'il satisfait les six propriétés suivantes :

Progressivité : Les premiers résultats devraient être retournés immédiatement et la taille de skyline augmente graduellement.

Complétude des réponses : L'algorithme devrait générer le skyline entier (complet) : Toutes les réponses qui satisfont la requête et qui ne sont dominées par aucune autre réponse.

Stabilité des réponses : L'algorithme devrait seulement renvoyer les points qui font partie de skyline. L'algorithme ne devrait pas retourner une réponse et après la remplacer par une autre réponse qui est meilleure que la première.

Justesse : L'algorithme ne devrait pas favoriser les points qui sont particulièrement bons dans une dimension. C'est une propriété importante pour garantir une grande image des réponses (les points retourner en premier sont bons dans toutes les dimensions).

Incorporation des préférences : L'utilisateur doit contrôler le processus de calcul, il est possible que l'utilisateur change ou ajoute des préférences pendant la phase d'exécution de l'algorithme.

Universalité : L'algorithme devrait être applicable à n'importe quelle distribution de données et pour n'importe quelle dimensionnalité.

	BNL	Bitmap	BBS
Progressivité	NON	OUI	OUI
Complétude des réponses	OUI	OUI	OUI
Stabilité des réponses	NON	OUI	OUI
Justesse	OUI	NON	OUI
Incorporation des préférences	NON	NON	OUI
Universalité	OUI	NON	OUI

Tableau II.5 : Comparaison entre les algorithmes BNL, Bitmap et BBS

Dans le **Tableau II.5**, on résume les propriétés de chaque algorithme. Un OUI indique la satisfaction d'une propriété par l'algorithme, alors qu'un NON indique la non satisfaction. Comme il est indiqué dans le tableau et selon les études réalisées dans **[BKS 01, TEO 01, KRR 02, PTF 05]**, l'algorithme BBS est le meilleur dans tous les cas. Notons que BNL est utilisé dans diverses applications avec n'importe quelle dimensionnalité quelconque. Son problème majeur est la dépendance totale de la mémoire centrale. L'algorithme à base d'index est très rapide pour

calculer les premiers points de skyline, le problème est qu'il retourne les réponses qui sont meilleures dans une dimension en premier, ceci n'est pas utile dans les applications interactives ou l'utilisateur nécessite une grande image (réponses qui sont meilleures dans toutes les dimensions) pour réagir en spécifiant d'autre préférences.

6. Conclusion

Dans ce chapitre on a présenté les principaux algorithmes pour le traitement des requêtes skyline dans l'environnement centralisé. Le principe de fonctionnement de chaque algorithme est illustré au travers d'un exemple décrivant un ensemble des caméras avec leurs prix et qualités. La requête considérée comme la recherche d'une caméra la moins *chère* et *de bonne qualité*.

Les algorithmes présentés dans ce chapitre sont conçus pour être utilisés dans un contexte centralisé (une seule base de données centrale). Aucun de ces algorithmes ne peut être utilisé pour résoudre le problème de skyline dans un environnement distribué **[HOS 05]**. Le chapitre 3 traite ce problème et discute les solutions proposées.

CHAPITRE 3

Approches d'Evaluation de Skyline dans un Contexte Distribué

Depuis plusieurs années et avec l'évolution des technologies, les systèmes informatiques centralisés sont progressivement remplacés par des systèmes distribués. Ces systèmes sont constitués d'un ensemble d'entités autonomes de calcul (ordinateurs, périphériques, processeurs, processus, etc.) interconnectées entre eux via un réseau et qui peuvent communiquer.

D'autres parts tous les algorithmes présentés dans le chapitre précédent se situent dans le cadre de calcul de skyline dans les bases de données centralisées. L'évaluation des requêtes skyline dans un environnement distribué (ED) représente un défi car, dans de nombreuses applications web, les attributs cibles sont disponibles sur différents sites qui sont accessibles seulement à l'aide d'interfaces externes très restreintes.

Les méthodes précédentes ne peuvent pas être appliquées dans un ED (elles sont conçues pour le cas centralisé) à cause de deux points :

- Il est supposé que les valeurs des attributs peuvent être retournées à un faible coût, qui n'est pas vrai pour le cas distribué.
- Les techniques centralisées se fondent sur des indices construits sur les attributs impliqués dans la requête. Cependant, ces structures de données centralisées ne sont pas disponibles dans un ED.

Dans ce chapitre, on va présenter quelques notions de deux principaux exemples de systèmes distribués et en particulier le Web qui se base sur le paradigme Client/serveur comme mode de fonctionnement, ensuite quelques algorithmes d'évaluation des requêtes skyline dans un ED sont présentés.

1. Le WEB et le modèle Client/Serveur

La technologie Web a été inventée au *CERN* (Centre Européen de Recherche Nucléaire) en 1989 par Tim Berners-Lee. L'objectif était la diffusion de l'information en physique nucléaire. L'idée essentielle était d'organiser les informations sous forme d'un hypertexte permettant les liens entre les documents.

Ainsi est née l'idée d'un vaste système d'information distribué à l'échelle mondiale sur lequel les utilisateurs peuvent surfer depuis leurs terminaux : la toile d'araignée mondiale des documents désignée par l'acronyme WWW ou le diminutif Web (World Wide Web).

1.2. Définition

Le web est le service le plus connu, le plus récent et maintenant le plus utilisé de consultation d'hyper documents. C'est un ensemble infini de sites reliés entre eux, hébergés sur les réseaux d'Internet et accessible par l'utilisation d'un logiciel : le browser (ou navigateur) capable de lire le langage *HTML*. Les applications web fonctionnent selon un environnement Client/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en termes d'entrée/sortie, qui leur fournit des services.

1.3. Modèle client/serveur

Bien que le terme client/serveur soit devenu le maître mot en informatique, il n'existe aucun consensus sur sa signification exacte. Voici donc une de ces définitions. Comme l'indique le terme, ce sont deux entités distinctes, fonctionnant entre elles sur un réseau pour accomplir une tâche.

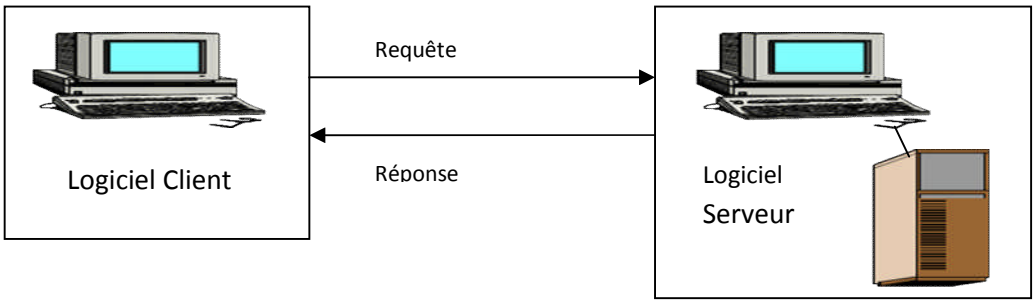


Figure III.1 : Architecture Client/serveur

Le client émet une requête vers le serveur grâce à son adresse et le port, qui désigne un service particulier du serveur. Le serveur reçoit la demande et répond à l'aide de l'adresse de la

machine cliente et de son port. Mais malheureusement, cette architecture présente de nombreux inconvénients :

- Le nombre de clients, le téléchargement et la demande de bande passante grandissants peuvent amener les serveurs à ne plus servir davantage de clients.
- Une caractéristique de cette architecture, est qu'elle nécessite très peu de puissance de calcul et de ressources du côté client, ce qui été justifié à l'ère où cette architecture a vu le jour, alors qu'à notre ère, la plupart des systèmes clients sont devenus très puissants.
- Le client dans une telle architecture agit de manière passive : il demande des services auprès des serveurs mais il est incapable d'offrir des services aux autres.

1.4. Avantages et inconvénients de l'architecture client/serveur

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

- Ressources centralisées : étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.
- Une meilleure sécurité : car le nombre de points d'entrée permettant l'accès aux données est moins important.
- Une administration au niveau serveur : les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.
- Un réseau évolutif : grâce à cette architecture il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeure.

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles :

- Un coût élevé dû à la technicité du serveur.
- Un maillon faible : le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui ! Heureusement, le serveur a une grande tolérance aux pannes.

Imaginons le potentiel de stockage et de puissance de calcul que représentent les clients, libres ou emprisonnées dans leurs réseaux locaux : en considérant que seulement 10 millions de machines sont connectées à Internet à n'importe quel moment et que chacune possède seulement 100 Mbit d'espace de stockage non utilisé, 1kbit/s de bande passante et 10% du pouvoir de calcul inexploité. A n'importe quel moment ces clients représentent donc 10^6 Go d'espace de stockage libre, 10 Gbit/s de bande passante disponible et 10^8 Mhz de «traitement» perdu.

2. Calcul de skyline dans un environnement distribué

Tous les algorithmes présentés dans le deuxième chapitre se situent dans le cadre de calcul de skyline dans les bases de données centralisées. L'évaluation des requêtes skyline dans un environnement distribué (ED) représente un défi car, dans de nombreuses applications web, les attributs cibles sont disponibles sur différents sites qui sont accessibles seulement à l'aide d'interfaces externes très restreintes.

Dans ce chapitre, quelques algorithmes d'évaluation des requêtes skyline dans un ED sont présentés. On distingue trois algorithmes pour évaluer ces requêtes (BDS, IDS et PDS).

2.1. Algorithme BDS (Basic Distributed Skyline Algorithm)

L'algorithme distribué de base de skyline BDS **[BGZ 04]** comprend deux phases. (i) La première phase identifie un sous ensemble d'objets qui inclut tous les objets skyline probablement avec quelque autre objets non skyline, (ii) la deuxième filtre cet ensemble en éliminant tous les objets non skyline.

Dans la première phase les données (objets) sont retournées par l'accès trié (*S: sorted access*) des différentes sources de donnée. Chaque source est invoquée dans un mode ROUND ROBIN jusqu'à ce que tous les objets qui peuvent faire partie de skyline soient retournés. Au cours de cette phase et pour chaque attribut D_i un tableau K_i est créé pour sauvegarder les valeurs d'attributs des objets retournés par l'accès trié (Figure 3-1). La condition d'arrêt est atteinte lorsque un objet O est détecté (tous les valeurs d'attribut de O sont vues dans les différents tableaux K_i), l'objet qui a toutes ses valeurs d'attributs retournées via l'accès trié est appelé l'objet de terminaison.

Site1		Site2		Site3	
hôtel	prix	hôtel	Distance à la plage	hôtel	Temps à l'aéroport
b	0	*i*	0	*e*	1
a	1	*f*	1	*c*	2
c	2	*d*	3	*j*	3
f	3	*e*	4	*f*	4
d	4	*c*	5	*b*	5
g	5	*b*	6	*g*	6
j	6	*h*	7	*h*	7
e	7	*a*	8	*i*	8
h	8	*g*	9	*d*	9
i	9	*j*	10	*a*	10

Tableau III.1 : Exemples des hôtels avec 3 attributs sur 3 sites différents

Concernant l'objet de terminaison un important lemme est démontré dans **[BGZ 04]**, quand un objet de terminaison T est détecté est aucun accès trié peut retourner une valeur égale à la valeur de T, tous les objets non retournés ne font pas partie de skyline.

La deuxième phase utilise des accès sélectifs ou aléatoires (*R: Rondom access*) focalisés pour trouver les autres valeurs d'attributs des objets retournés dans la première phase, et performe des comparaisons par pairs entre les objets des différents tableaux, les objets dominés sont éliminés avant de renvoyer le skyline à l'utilisateur.

Pour réduire le nombre de comparaisons ; il est prouvé dans **[BGZ 04]** qu'un objet O peut être dominé seulement par les objets du même tableau que O.

Pour mieux comprendre le fonctionnement de cet algorithme, on va appliqué cet algorithme à l'exemple des hôtels **Tableau III.1** et donc on a trois sources de donnée (attributs) sur trois sites différents.

Puisque on a trois attributs, alors trois tableaux K_i sont créés :

Hôtel	D1	D2	D3
b	0		5
a	1		
c	2	5	2
f	3	1	4
d	4	3	

*K*1

hôtel	D1	D2	D3
i		0	
f	3	1	4
d	4	3	
e		4	1
c	2	5	2

*K*2

hôtel	D1	D2	D3
e		4	1
c	2	5	2
j			3
f	3	1	4
b	0		5

*K*3

Figure III.2 : Les tableaux qui sauvegardent les objets retournent dans la 1ère phase

Pour les trois attributs de notre exemple d'hôtel prolongé (en supposant que les valeurs d'attribut en double sont autorisées). Ces tableaux sont passés à la phase deux, au cours de laquelle les objets non-skyline sont filtrés pour récupérer les valeurs d'attribut manquantes par accès aléatoire (par exemple, *R* (*D*2, *b*, 6)) et d'interprétation une comparaison par paire entre tous les objets dans les tableaux. Objets trouvés à être dominé par une d'autres sont filtrés.

Afin de réduire le nombre de comparaison, il est observé dans **[BGZ 04]**, que l'objet O ne peut être dominé par d'autres objets qui existent dans toutes les tables qui contiennent O. par exemple, depuis la table *K*1 contient objet *b* mais pas *e*, *b* ne peut pas être dominé par e. Il découle du fait que la valeur de e à l'attribut *D*1 ne doit pas être inférieure à celle de *b* car il n'est pas récupéré avant la fin de la phase une. Par conséquent, les comparaisons par paires doivent être effectuées entre des objets qui existent dans la même seule table.

En utilisant le même argument, une valeur manquante pour l'attribut *D*i dans un tableau ne peut pas être inférieure à la plus grande valeur *D*i stockées dans la table. Par exemple, selon le tableau *K*1, la valeur de *d* à *D*3 devrait d'au moins 5. Cela signifie *d* est prouvé à être dominé par *f* (puisque *d* a aussi des valeurs d'attributs plus que *f* en *D*1 et *D*2), sans pour autant obtenir la valeur de *d* à *D*3 par un accès aléatoire. Certains une surcharge du réseau peut ainsi être évitée.

2.2. Algorithme IDS (Improved Distributed Skyline Algorithm)

Dans l'algorithme précédent, il faut avoir au moins un objet (l'objet de terminaison) avec toutes ses valeurs d'attributs connues pour que l'algorithme se termine. Si cet objet est connu a

priori, il est possible d'effectuer moins d'accès trié dans la première phase, et peu de comparaisons dans la deuxième phase. Pour cela les auteurs de **[BGZ 04]** ont proposé l'algorithme IDS basé sur une heuristique pour trouver l'objet de terminaison qui fait terminer l'algorithme plus rapidement.

Au début IDS réalise deux accès (triés et sélectifs) pour obtenir toutes les informations concernant les objets, et le nombre d'accès trié requis pour retourner toutes les valeurs d'attributs d'un objet O est obtenu on calculant un score s (s est la somme de différences entre les valeurs d'attributs de O et les derniers valeurs retourner par l'accès trié dans les différents tableaux).

IDS permet d'améliorer le comportement de l'exécution globale en réduisant le nombre d'accès de données, son succès dépend de l'exactitude de l'heuristique pour prévoir correctement l'objet de terminaison mais l'heuristique ne réussi pas dans toutes les situations en plus dans les deux algorithmes (BDS et IDS). Un objet est confirmé comme objet de skyline seulement après que la comparaison dans la deuxième phase est accomplie mais dans des applications en temps réel ceci peut être vu comme inconvénient à cause du temps de réponse important avant que le premier objet de skyline soit retourné et pour cela *Lo et ses collègues* **[LYL 05]** ont proposé leur algorithme PDS (Progressive Distributed Skyline algorithm) qui permet de déterminer si un objet fait partie de skyline des qu'il est retourné la première fois et il le fait sortir immédiatement s'il n'est pas dominé par d'autres objets.

2.3. Algorithme PDS (Progressive Distributed Skyline algorithm)

PDS est basée sur le principe de progressivité et n'attend pas la fin de l'algorithme pour retournés les objets skyline, il (retourner les objets skyline progressivement), ainsi une méthode d'estimation de rang est nécessaire pour définir l'objet de terminaison.
Les deux principales idées de **PDS** (la progressivité et l'estimation de rang) sont discutées dans **[LYL 05]** :

2.3.1. La progressivité

Pour permettre la progressivité il faut déterminer si un objet appartient au skyline dés que ses valeurs d'attributs seront retournées et pour cela les auteurs de **[LYL 05]** ont prouvés le suivant :

Lemme : si l'accès trié d'une source de donnée *D* renvoi les valeurs des attributs dans un ordre strictement croissant, un objet *O* de *D* peut seulement être dominé par des objets qui sont retournés avant *O*.

La vérification de dominance est faite en comparant un objet *O* avec tous les objets retournés avant lui dans la même source et pour accélérer le processus et réduire le nombre de comparaisons les autres de **[LYL 05]** ont choisis un arbre R^*(index multidimensionnel dans la mémoire centrale) et chaque attribut i impliqué dans la requête correspond à un arbre *R*i contient les objets skyline découvert basés sur l'accès trié à la source *D*i. Un objet *O* retrouvés de *D*i est comparé à tous les objets dans *R*i et s'il n'y a aucun qui domine *O* alors *o* est défini comme objet de skyline et il est inséré dans *R*i.

Quand un nouveau objet *O* est retourné, la vérification de dominance est faite en exécutant une requête Q_0 qui à l'origine comme coin inférieur gauche et *O* comme supérieur droite. Si le résultat de Q_0 est vide alors o est un objet de skyline, autrement il existe au moins un objet *O*' qui est meilleur que *O* dans toutes les dimensions et par conséquent *O* est éliminé. Le cas général ou il y a des valeurs dupliquées et l'accès trié ne peut renvoyer des valeurs dans un ordre strictement croissant, le lemme précédent ne peut être appliqué directement, ce problème est résolu par les auteurs de **[LYL 05]** en gardent un buffer *B*i pour chaque source de donné Di, si O est défini comme un possible objet de skyline, il est inséré dans *B*i au lieu de le rapporté comme objet de skyline immédiatement. Le buffer *B*i est un autre arbre R^* qui tient les objets possibles d'être dans le skyline avec la même valeur dans l'attribut i et par conséquent, pour chaque objet *O* inséré dans *B*i ; une vérification est faite (si *O* est dominé ou non). Apres qu'un objet avec une grande valeur d'attribut de la source *D*i est inséré dans *B* ; les objets qui restent dans *B*i sont retournés comme objets de skyline définitivement et le buffer est vidé.

2.3.2. Estimation de rang

[LYL 05] ont supposés qu'un bon objet de terminaison est caractérisé par son rang dans les listes Ki (tableau ordonné de valeurs d'attribut). Pour un objet O Il est noté que :

- $Rank_i(O)$: le rang de l'objet *O* dans la source *D*i.
- $Sumrank_i(O)$ = est la somme des rangs de *O* dans tous les sources de donnée *D*i

Si *T* est un objet de terminaison alors le nombre minimum d'accès exigés pour le localiser par l'accès trié est $Sumrank_i(T)$, un bon objet de terminaison est celui qui a le minimum *Sumrank* et par conséquent le nombre d'objets qui doivent être comparés augmente avec l'augmentation de $Sumrank_i(T)$,

PDS à les informations concernant les objets qui sont retournés par l'accès trié et le rang d'un objet *O* retourné par le j^{ime} accès trié dans la source D_i est *j* (le rang du premier objet retourné est 1) mais l'accès sélectif retourne l'attribut d'un objet sans aucune information sur son rang, les auteurs de **[LYL 05]** ont proposés d'employer la régression linéaire entant que méthode d'estimation de rang. Pour une source D_i, l'équation qui calcule le rang d'un objet O est donnée comme suit:

$$Rank_i(O)=\boldsymbol{a_i}\times value_i+\boldsymbol{b_i} \qquad (1)$$

$value_i$ est la valeur de l'objet O dans la source D_i, supposons qu'on a fait K accès trié à la source D_i, les coefficients a_i et b_i peuvent être obtenus par les formules suivantes:

$$a_i=\frac{K\sum_i Rank_i\ value_i-(\sum_i value_i)(\sum_i Rank_i)}{K\sum_i value_i^{\,2}-(\sum value_i)^2} \qquad (2)$$

$$b_i=\frac{\left(\sum_i [Rank_i\right)(\sum_i value_i^2)]-(\sum_i value_i)\left(\sum_i [Rank_i\ value_i\right)]}{K\sum_i value_i^{\,2}-\left(\sum value_i\right)^2} \qquad (3)$$

3. Conclusion

Dans ce chapitre, premièrement nous avons présenté quelques notions sur le WEB et le modèle client/serveur. Puis on est intéressé aux approches d'évaluation de skyline dans un contexte distribué, on a présenté les algorithmes (BDS, IDS et PDS) qui permet d'évaluer les requêtes skyline dans un environnement distribué **[BGZ 04]** (Bases de données accessibles via le web)**.**

CHAPITRE 4

Contribution à L'Évaluation de Skyline Dans un ED

Ce chapitre aborde le problème de l'évaluation des requêtes flexibles dans un contexte distribué (BD accessible via le WEB). Récemment, quelques travaux ont été proposés pour tenter de résoudre ce problème, voir par exemple **[LYL 05]** et **[BGZ 04]**. L'approche proposée consiste à améliorer l'algorithme proposé dans **[BGZ 04]**. L'idée suggérée permet de combiner l'algorithme proposée dans **[BGZ 04]** (voir section2.1 du chapitre III) et l'algorithme BNL de **[BKS 01]**. La première phase permet de construire un index global de routage qui décrit les données des différentes sources, afin de trouver l'ensemble des pairs pertinents pour la requête posée. La seconde phase présente un algorithme de recherche des réponses satisfaisant le mieux la requête.

Dans ce chapitre, premièrement on décrire l'approche proposée dans la section suivante, puis la deuxième partie présente notre implémentation avec quelques outils de résiliation et des illustrations pratiques du prototype développé.

1. Evaluation de requêtes skyline dans un contexte distribué : Une approche hybride

Dans cette partie, le problème considéré concerne l'évaluation des requêtes Skyline **[BGZ 04]** dans un contexte de bases de données distribuées accessibles via le WEB. Ce type de requêtes permet de retourner un ensemble d'objets qui ne sont dominés par aucun autre objet de la base de données.

L'approche proposée combine l'algorithme décrit dans **[BGZ 04]** pour retourner les tuples à partir de différents sites comme première phase et l'utilisation de l'algorithme BNL **[BKS 01]** pour faire sortir le skyline de l'ensemble. On reprend l'exemple des hôtels **chapitre III** pour bien illustrer cette approche.

Les bases de données sont supposée décrites par des schémas différents (les problèmes soulevés par la médiation de données ne sont pas considérés ici).

1.1. Stratégie d'Evaluation

Dans cette section, on présente une stratégie efficace pour retourner les tuples pertinents à une requête posée, l'approche comprend deux phases. (i) La première phase identifie un sous ensemble d'objets qui inclut tous les objets skyline probablement avec quelque autre objets non skyline, (ii) la deuxième utilise l'algorithme BNL pour filtrer cet ensemble en éliminant tous les objets non skyline. Reprenant toujours l'exemple des hôtels avec trois attributs distribués sur 03 sites, puisque on a trois attributs, alors trois tableaux K_i sont créés :

Hôtel	D1	D2	D3
b	0		5
a	1		
c	2	5	2
f	3	1	4
d	4	3	

*K*1

hôtel	D1	D2	D3
i		0	
f	3	1	4
d	4	3	
e		4	1
c	2	5	2

*K*2

hôtel	D1	D2	D3
e		4	1
c	2	5	2
j			3
f	3	1	4
b	0		5

*K*3

Figure IV.1 : Les tableaux K_I.

Dans cette première phase on va ajouter au autre tableau P qui contient tous les tuples retournés à partir des différents sites **Tableau III.1**

Hôtel	b	i	e	a	f	c	d	j
D1	0	?	?	1	3	2	4	?
D2	?	0	4	?	1	5	3	?
D3	5	?	1	?	4	2	?	3

Figure IV.2 : Le tableau P (Tuples retournés dans la 1ère phase).

Pour la deuxième phase, on procède de la même manière que dans le chapitre 03 et on fait des accès aléatoires sur le tableau P seulement et après identifier tous les objets skyline par l'algorithme BNL

❖ **Algorithme de Calcul**

0. Initialiser l'ensemble de données $P := \emptyset$ contenant les enregistrements d'un identificateur et n valeurs réels indexées par les identificateurs, initialiser les n listes de $K_1 \ldots K_n := \emptyset$ contenant les enregistrements des identificateurs et des valeurs réels, et initialiser les n valeurs réels $P_1 \ldots P_n := 1$.
1. Initialiser le compteur $i := 1$.

   1.1 Tirer le prochain objet O_{nouv} par accès stockée sur la liste S_i

   1.2 Si $O_{nouv} \in P$, mettre à jour son enregistrement les $i^{ème}$ valeurs réels avec $S_i(O_{nouv})$, sinon créer un tel enregistrement en P.

   1.3 Ajouter O_{nouv} et $S_i(O_{nouv})$ à la liste K_i.

   1.4 Mettre $P_i := S_i(O_{nouv})$ et $i := (i \bmod n) + 1$.

   1.5 Si tous les scores $S_i(O_{nouv})$ $(1 \leq j \leq n)$ sont connus, passer à l'étape 2 sinon à l'étape 1.1.
2. Pour i=1 jusqu'à n fair

   2.1 Tant que $P_i = S_i(O_{nouv})$ faire accès triés sur la liste S_i et traiter les objets trouvés comme dans l'étape 1.2 jusqu'à 1.3.
3. Si plus d'un objet est parfaitement connu, BNL(P).
4. Pour i=1 jusqu'à n faire

4.1 faire tous les accès nécessaires aléatoire pour les objets dans P_i

5. retirer les objets dominés dans P par BNL(P).
6. En sorties *P* l'ensemble de tous les objets non dominées.

2. Implémentation

2.1. Le langage java

Java est un langage de programmation à usage générale, évolué et orienté objet dont la syntaxe est proche du C. Il existe 02 types de programmes en java : les applets et les applications. Une application autonome (standard alone program) est une application qui s'exécute sous le contrôle direct du système d'exploitation. Une applet est une application qui est chargée par un navigateur et qui est exécutée sous le contrôle d'un plugin de ce dernier.

2.2. Outil Netbeans

Netbeans IDE6.9.1 (interface définition language). Est un environnement de développement en java open source écrit en java. Le produit est composé d'une partie centrale à la quelle il est possible d'ajouter des modules tel que Poseidon pour la création avec UML. L'interface de création d'un projet dans Netbeans est représentée dans la **Figure IV.3**

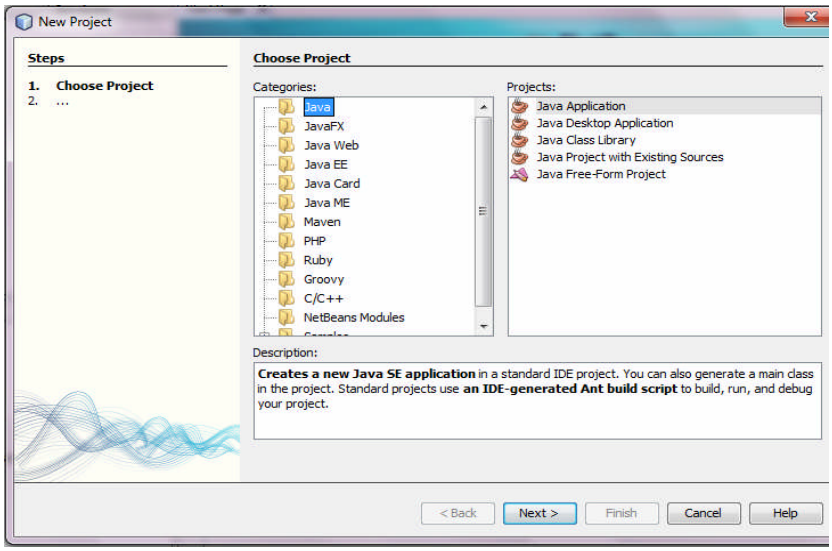


Figure IV.3 : Interface Netbeans.

2.3. MySQL

MySQL est un système de gestion de base de données (SGBD). Selon le type d'application, sa licence est libre ou propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, Informix et Microsoft SQL Server. Pour crée une base de donnée, procédez la façon suivant :

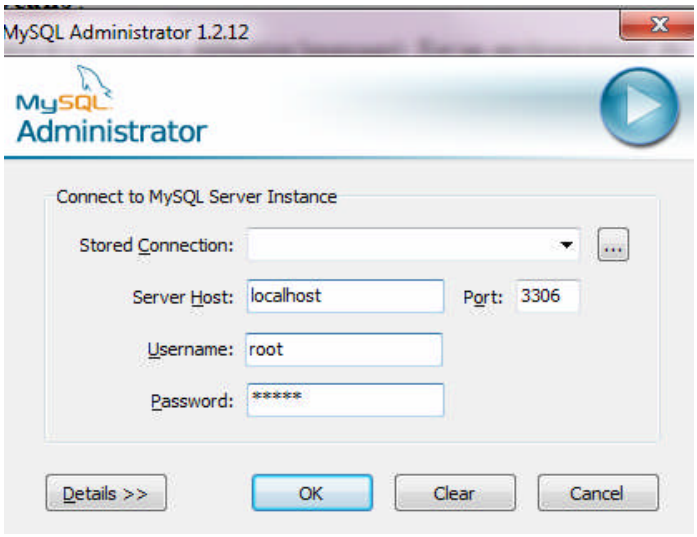


Figure IV.4: Interface MySQL.

Après on clic sur catalogs et avec un clic droit la fenêtre suivante s'affiche, on clic sur create new schema pour créer une base de données.

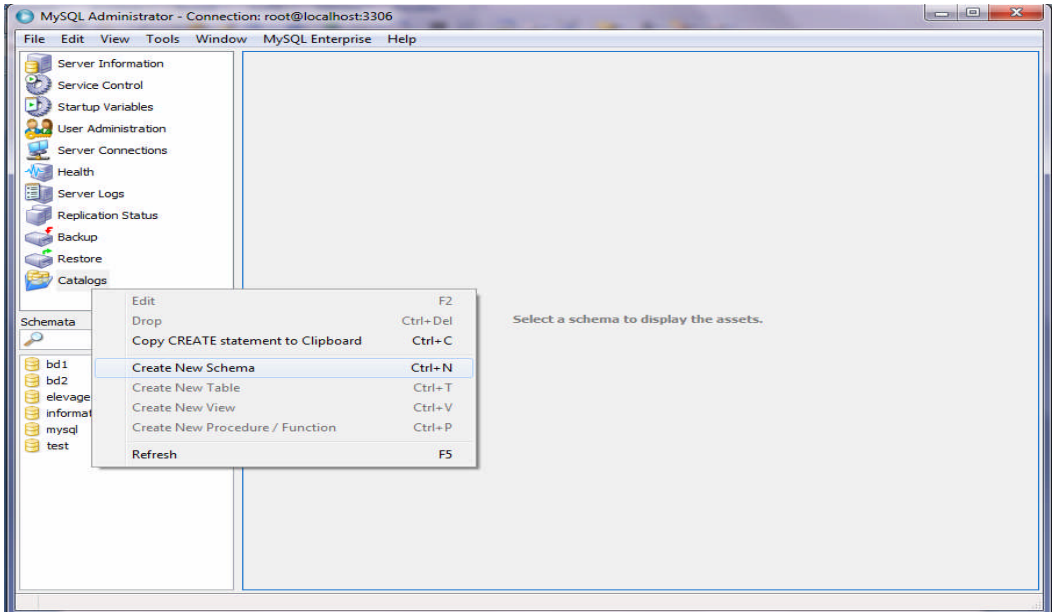


Figure IV.5: Interface de créer une base de données.

2.4. La connexion à une base de données

Pour la liaison entre MySQL et java on utilise le code suivant :

```
Class.forName("com.mysql.jdbc.Driver").newInstance();

        String s1 = "root";

        String s2 = "admin";

        String s3 = "jdbc:mysql://localhost/bd1";
```

2.5. Scenario d'exécution

Ce scénario d'exécution présente un exemple d'utilisation de l'application.

- ✓ Lancement de l'application. Ce qui nous donne l'interface, ci-dessous : cette interface est composée de :
  - ➢ Fichier qui contient un sous menu 'Quitter'
  - ➢ Connexion BD qui contient deux sous menu 'Connexion au site 1' et 'Connexion au site 2'
  - ➢ Sources de données qui contient deux sous menu 'Données sur site 1' et 'Données sur site 2'
  - ➢ Traitement skyline qui contient deux sous menu 'Algorithme BDS' et 'Algorithme par étape'

Figure IV.6 : Interface de l'application.

Nous allons voir la première interface de notre application : **Figure IV.6**

Après avoir présenté l'application on fait un clic sur la barre menu de la **'connexion BD'** qui nous fait remarqué bien qu'il existe (connexion au site 1, connexion au site 2), **Figure IV.7**

Figure IV.7 : La barre menue de l'application

Pour la réalisation de notre base de données on clic connexion BD qui est composée des éléments suivant :

1. **Connexion au site 1 :** on clic pour demander la connexion de la base de données pour exécuter et après nous allons voir le message suivant « la connexion au site 1 est

établies » lors d'une connexion qui réussit puis on clic OK pour fermer la fenêtre qui s'affiche dans la

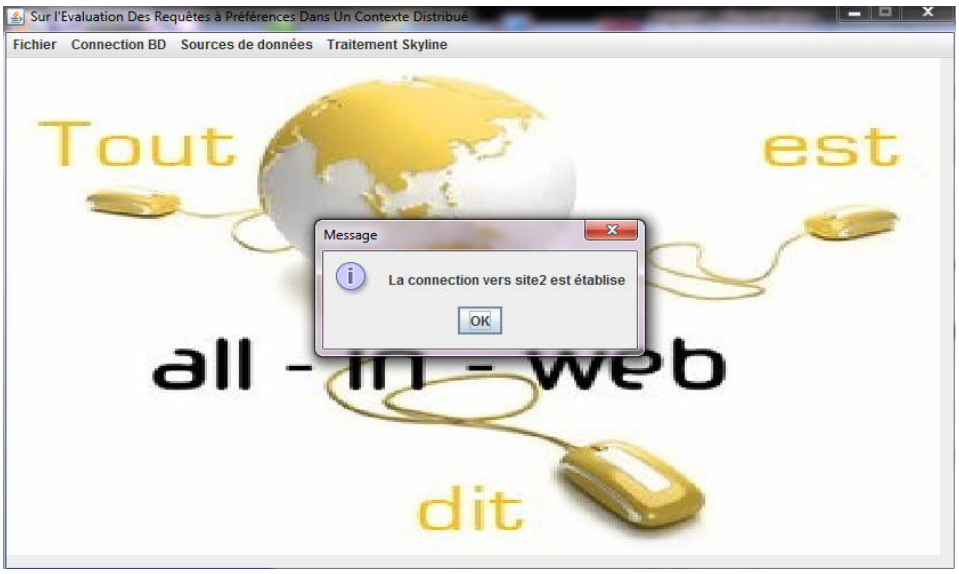


Figure IV.8 : Demande de connexion à la base de données

Après acceptation de la connexion, cliquez sur **'Source de données'** puis cliquez sur **'Données sur site 1'** nous allons voir une fenêtre qui s'affiche dans la **Figure IV.9**

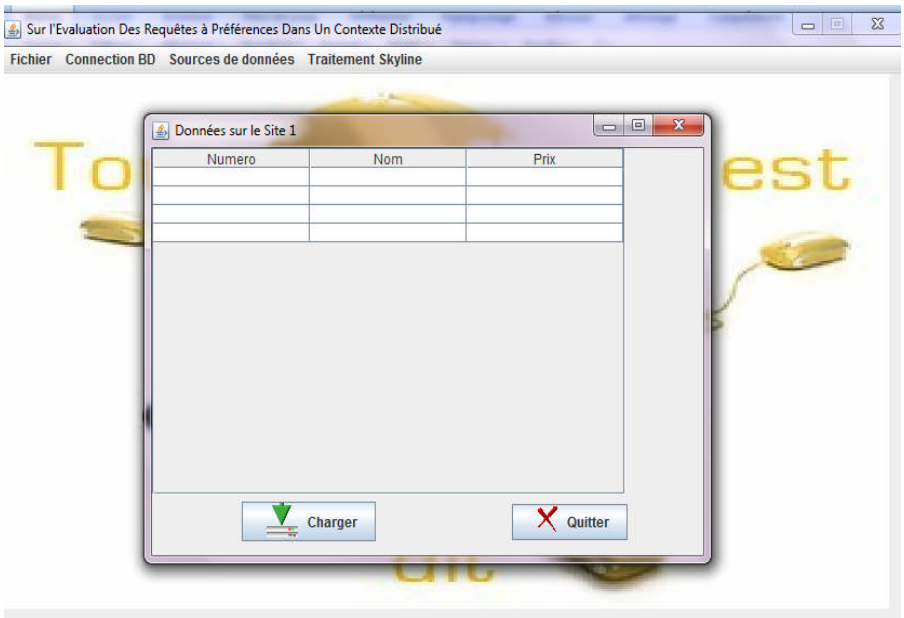


Figure IV.9 : Donnée sur le site 1.

2. **Charger les données :** cliquez sur le bouton charger pour afficher les enregistrements de la base de données comme nous le remarquons dans la **Figure IV.10**

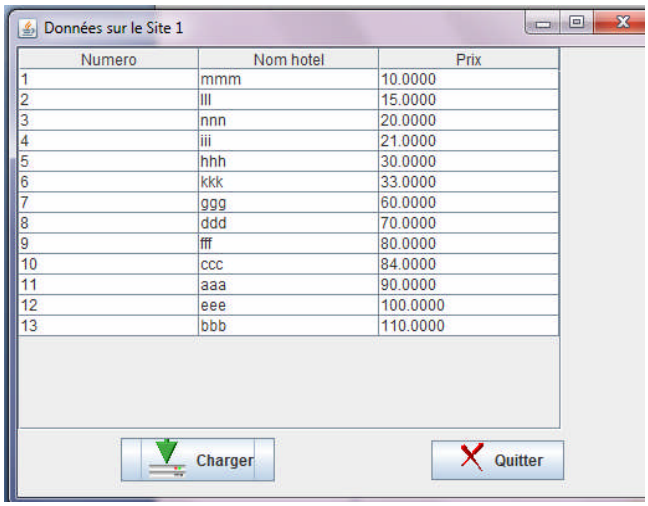


Figure IV.10 : Charger les données

Avez-vous déjà vue cet interface lors d'une ouverture de notre application et suite d'exécutions sur les données qui nous chargées, cliquer sur **'Traitement skyline' Figure IV.11**

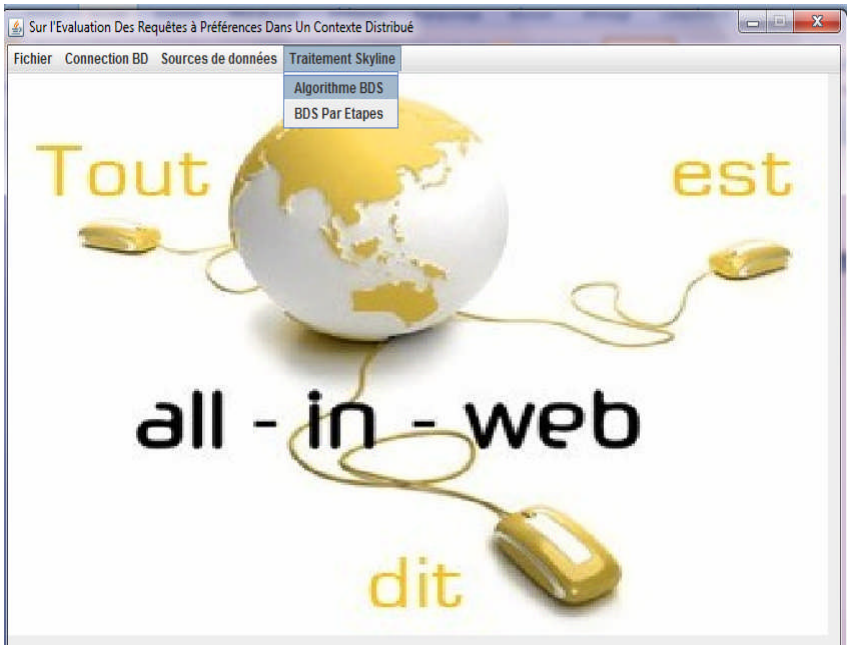


Figure IV.11 : Traitement des données

Pour exécuter l'algorithme BDS on clic sur **'Algorithme BDS'** et la fenêtre suivante s'affiche dans la **Figure IV.12**

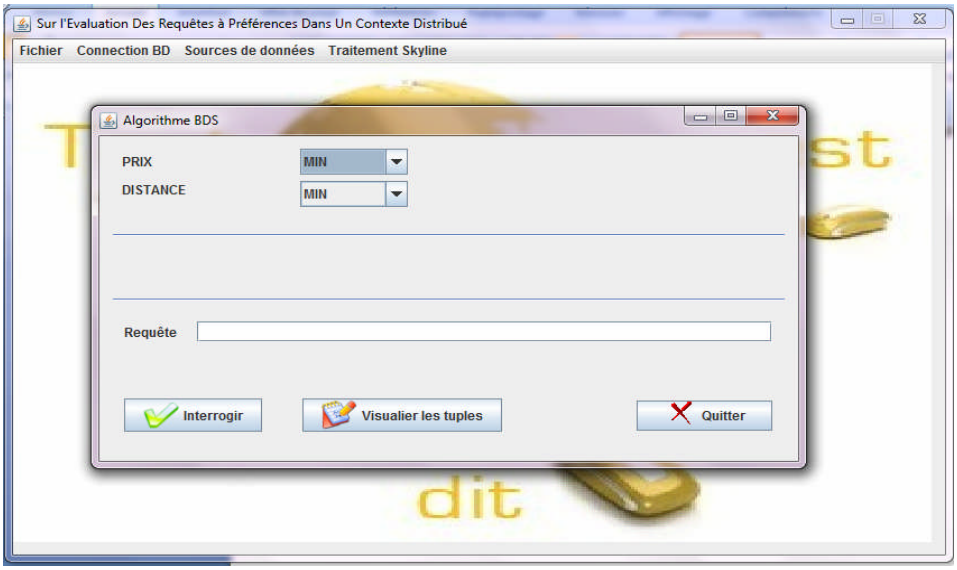


Figure IV.12 : Algorithme BDS

Dans cette fenêtre on clic sur **innteroger** pour exécuter la requête, puis on clic sur **visualiser les tuples** pour afficher les résultats.

BDS par étape : si on veut afficher les étapes d'exécution de BDS on clic sur **traitement Skyline** puis **BDS par étapes**

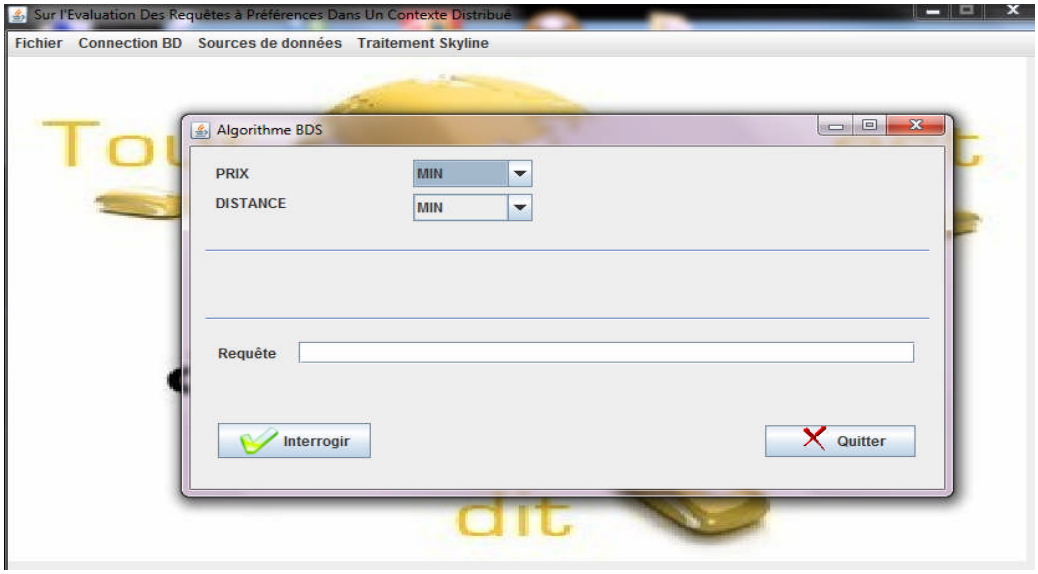


Figure IV.13 : Algorithme BDS par Etapes

Après avoir cliquer sur le bouton interroger, la fenêtre suivante s'affiche et montre la première phase de l'algorithme BDS(Accès trié pour chaque source).

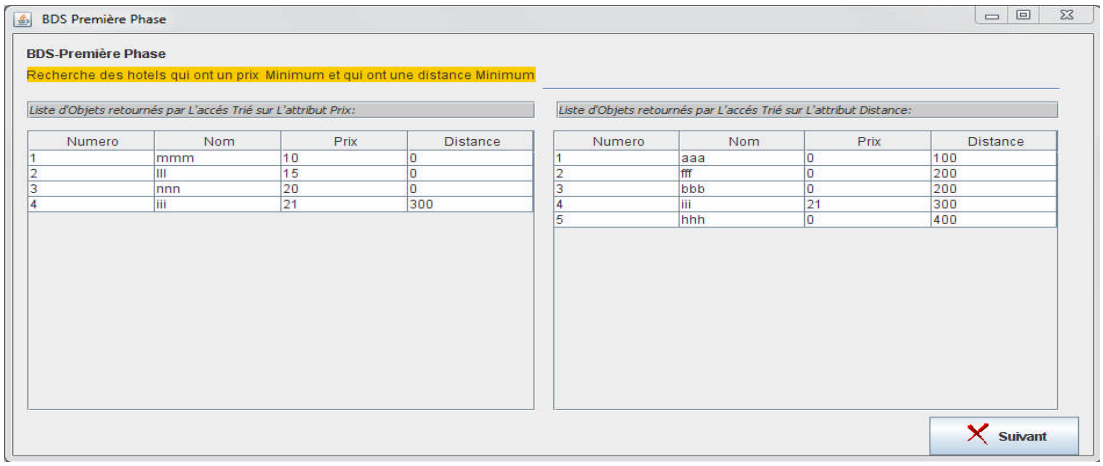


Figure IV.14: Algorithme BDS (Première phase)

Pour passer à la deuxième phase de BDS, on clic sur suivant et un accès aléatoire est faite sur toutes dimensions comme il est montré dans la figure suivante :

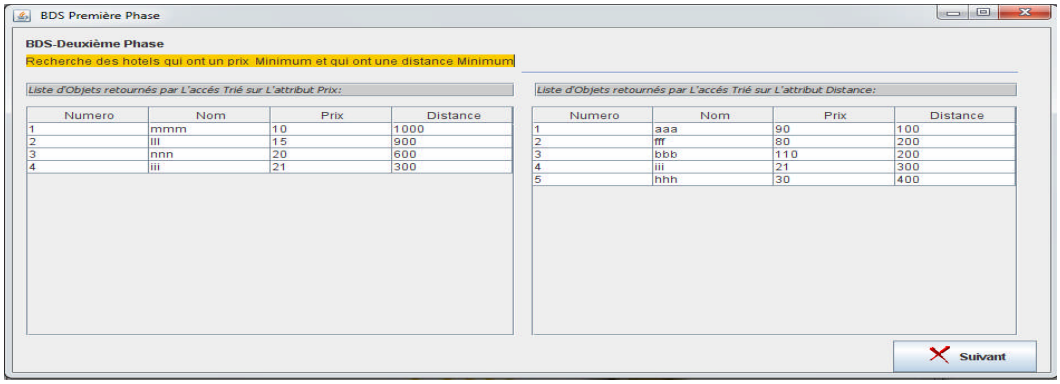


Figure IV.15 : Algorithme BDS(deuxième phase)

Finalement l'ensemble des résultats (skyline) est montré dans la figure suivante :

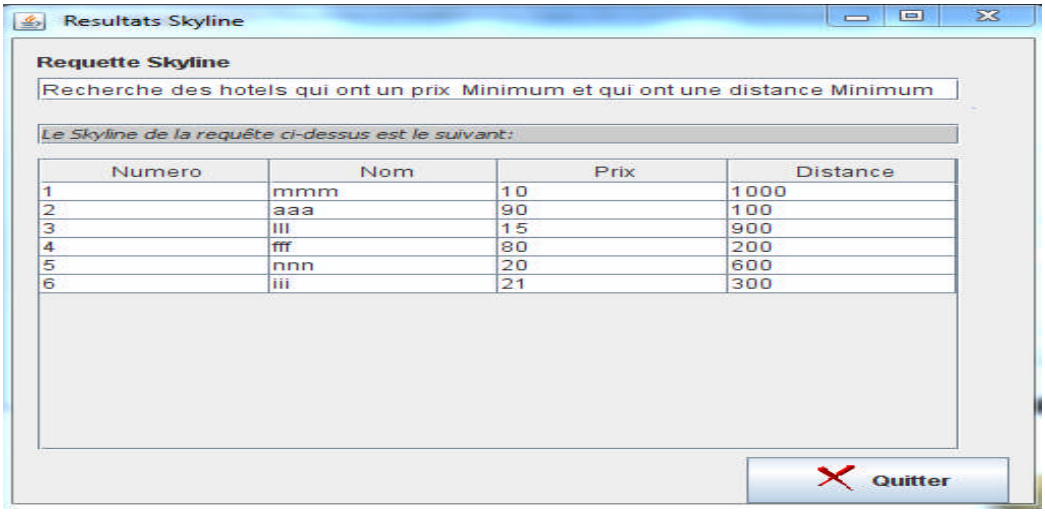


Figure IV.16 : Algorithme BDS(deuxième phase)

Conclusion Générale

Conclusion Générale

Habituellement, l'interrogation classique de bases de données est qualifiée d'interrogation booléenne (ou rigide) dans le sens où l'utilisateur lorsque il formule une requête, avec SQL par exemple, obtient une réponse ou rien du tout. Ce type d'interrogation s'avère inadéquate et inefficace pour certaines applications, qui supportent des données vagues, imprécises et/ou des préférences dans les attributs. L'interrogation avec préférences constitue alors une alternative à l'interrogation booléenne pour ces types d'applications.

Les requêtes à préférences est actuellement un thème de recherche qui retient l'intérêt de nombreux chercheurs. Un des formalismes les plus utilisés pour représenter les préférences des utilisateurs dans leurs requêtes est les requêtes dites "Skyline". Ce type de requêtes permet de retourner un ensemble d'objets qui ne sont dominés par aucun autre objet de la base de données. La plupart des travaux sur les Skylines se situent dans le cadre des bases de données centralisées.

L'étude présentée dans ce mémoire traite le problème d'évaluation des requêtes à préférences dans un contexte distribué (Bases de données accessibles via le Web). L'évaluation des requêtes Flexibles dans ce contexte représente un défi car, dans de nombreuses applications Web, les attributs ciblent sont disponibles sur différents sites qui sont accessibles seulement à l'aide d'interfaces externes très restreintes.

Dans le premier chapitre, nous avons décrit le principe des requêtes à préférences et sa modélisation dans le cadre des SGBD relationnel. Nous avons également étudié l'approche des requêtes Skyline, qui représente un des formalismes les plus utilisés pour représenter les préférences des utilisateurs dans les requêtes. Dans le chapitre 2, premièrement nous avons présenté quelques approches d'évaluation des requêtes skyline dans un contexte centralisé, puis dans le troisième chapitre on est intéressé aux approches d'évaluation de ces requêtes dans un environnement distribué.

A l'issu de cette étude et dans le chapitre 4, nous avons dans un premier temps proposé une démarche (pour l'évaluation des requêtes skyline dans le WEB) qui combine l'approche BDS et l'algorithme BNL.

Plusieurs perspectives futures s'imposent. Nous en citons les suivantes :

- Compléter et adapter les méthodes existantes pour le cas des requêtes flexibles (ou les préférences sont exprimés au moyen des prédicat flous).

Etude des performances des algorithmes pour différentes distributions de données

Référence Bibliographiques

Bibliographie

[AWF 00] : R. Agrawal and E. Wimmers. A. Framework, *for Expressing and Combining Preferences*. In Proc. SIGMOD, Texas, USA, 2000, pp.297-306.

[BGZ 04] : W.-T. Balke, U. Guntzer, J.X. Zheng, *Efficient distributed skylining for Web information systems*, in: Proceedings of Extending Database Technology (EDBT), 2004, pp. 256–273.

[BKS 01] : S. Borzsonyi, D. Kossmann, K. Stocker, *The skyline operator*, in: Proceedings of ICDE, 2001, pp. 421–430.

[CHE 05] : Changkai Li, K. Chen_Chuan Chang, Ihab F. Ilyas, Sumin Song, RankSQL : *Query Algebra and Optimization for Relational Top-k Queries*. In SIGMOD 2005, pages 131-142, 2005.

[EOT 03] : P-K. Eng, B.C. Ooi, and K-L, Tan. *Indexing for progressive Skyline computation. Data & Knowledage Engineering,* 46(2) :169-201, 2003.

[HGH 99] : Hjaltason, G. Samet, H. Distance *Browsing in Spatial Database*. ACM *computation. Data & Knowledge Engineering*, 46(2) :169-201, 2003.

[HOS 05] : K. Hose, *Processing Skyline queries in P2P systems*, VLDB 2005 PhD Workshop, Trondheim, 2005.

[KIE 02] : W. Kiessling. *Foundations of Preferences in Database Systems*. Proc. Very Large Data Bases, 2002.

[KLP 75] : H. T. Kung, F. Luccio, and F. P. Preparata. *On finding the maxima of a set of vectors*. Journal of the ACM, 22(4) :469-476, 1975.

[KRR 02] : D. Kossmann, F. Ramsak, S. Rost, Shooting stars in the sky: *An online algorithm for skyline queries*, in: Proceedings of VLDB, 2002, pp. 275–286.

[LPY 07] : L. Su, P. Zou and Y. Jia. *Adaptive Mining the Approximate Skyline over Data Stream*.Springer Berlin/Heidelberg, Vol. 4489/2007, Computational Science – ICCS 2007, 742-745.

[LYL 05] : X. Lin, Y. Yuan, W. Wang, H. Lu, Stabbing the sky: *Efficient skyline computation over sliding windows*, in: Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5–8 April 2005, Tokyo, Japan, IEEE Computer Society, pp. 502–513.

[PRS 85] : F. P. Preparata and M. I. Shamos. *Computation Geometry – An Introduction.* Springer-Verlag, New York, 1985.

[PTF 05] : D. Papadias, Y. Tao, G. Fu, and B. Seeger. *Progressive skyline computation in database systems*. ACM *Transactions on Database Systems (TODS)*, 30(1):41–82, 2005.

[TEO 01] : K-L. Tan, P-K. Eng, and B.C. Ooi. *Efficient progressive Skyline computation. In Proceedings of the International Conference on Very Large Data Bases* (VLDB), pages 301-310, Roma, Italy, Septembre 11-14-2001.