REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseaux et Télécommunication

Par :

LOUMANI Tayeb

Sur le thème

Conception et implémentation d'un outil assistant pour la conception physique des bases de données relationnelles

Soutenu publiquement le .. / 06 / 2018 à Tiaret devant le jury composé de :

Mr CHIKHAOUI Ahmed	Dr.	Université de Tiaret	Président
Mr OUARED Abdelkader	MAA	Université Tiaret	Encadreur
Mme LAKHDARI Aicha	MAA	Université Tiaret	Examinateur

Résumé

La *conception physique* est l'une des phases cruciales du cycle de vie de la base de données. Cela est dû à son rôle important dans la sélection des structures d'optimisation tels que les vues matérialisées, les index et le partitionnement pour accélérer les performances des requêtes. Cette phase a été amplifiée par les besoins continus de stockage et de gestion efficace du déluge de données dans les systèmes de stockage.

Pour procéder à l'interrogation de cet ensemble gigantesque de données, des requêtes très complexes doivent être déployées sur la base de données, ceci dit, ces requêtes sont caractérisées par des opérations de sélection sur les tables, suivies de jointures. Sans techniques d'optimisation, le coût d'exécution de cette interrogation peut être très important (des heures voir des jours), parmi elles, on trouve les techniques redondantes comme les index, les vues matérialisées et la fragmentation verticale, et les techniques non redondantes comme la fragmentation horizontale et le traitement parallèle ; durant la conception de la base de données, l'administrateur a le choix de sélectionner un ensemble de techniques d'optimisation de manière isolée ou combinée en tenant compte du degré de similarité entre ces dernières. Cette situation motive les éditeurs de systèmes de gestion de base de données commerciaux et non commerciaux (par exemple, *SQL Tuning Advisor - Oracle et Parinda -PostgreSQL*) pour proposer des outils (appelés *advisors*) pour aider les administrateurs de base de données dans leurs tâches lors du choix de leurs structures d'optimisation pertinentes pour un Schéma base de données et une charge de travail.

Dans le cadre de ce mémoire, nous nous intéressons aux techniques d'optimisation des indexes afin de proposer un ensemble de solutions visant à optimiser les requêtes SQL. Nous proposons une approche globale générique pour sélectionner un ensemble de structure physique en intégrant les préférences des utilisateurs. Enfin, une étude de cas de notre contribution est présentée pour souligner son importance.

Mots-clés : Conception Physique, Traitement de Requêtes, Structure d'Optimisation, Advisors, Optimisation.

مُلَخَّصٌ

في عصرِ البيَاناتِ الضخمةِ، أصبحت إدارَةُ استهلاك الطَّاقة بالنِّسبَة للخَوَادِمِ ومراكز البيانات تحدِّيًا كبيرا للشركات والمؤسسات والدُّوَل. من بين التطبيقات المنتشرة بكثرة على مراكز البيانات، نَجِدُ أنَّ أنظمة إدارة قواعد البيانات واحدة من أكبر مستهلكي الطَّاقة الكهرَبائيَّة، وذلك أثناء تنفيذ الاسْتِغْلاَمَات المعقَّدة التي تنْطوي على حجمٍ كبير جدًّا من البيانات. وعِلاوةً على ذلك، عِلاجُ هذا النوع من قواعد البيانات يتطلَّب تكنولوجيا وبُنيَة تحتية باهظةَ الثَّمن ومستهلكةً كبيرةً للطَّاقة الكهربائيَّة. إلى زمن غيْرِ بعيد، كانت تَكْلُفَةُ الطَّاقة أثناء معالجة الاستعلامات الخاصَّة باستخدام وتشغيل قواعد البيانات كبيرةِ الحجم مُهْمَلَةً تماما، سواءا من قِبَلِ المستخدمين أو من قبل المُصمِّمين. حيث أنَّ العامل الأكثَرَ أهمِّية للمستخدِم هو تقْليلُ زمَن مُعالجَة الاستعلامات وتَلَقِّي النَّتائج بسُرعة. في هذه الأُطروحة نقترحُ صِياغةً مُتعدِدة الأهداف لمشاكل استغلال قواعد البيانات، وهذا عن طريق الأخْذِ بعيْن الاعتبار كلاًّ من الاحتياجات غيْرِ الوظِيفِيَّة: تحسينُ الأداءِ وتقليلُ استهلاك الطَّاقة عند تشغيل مجموعةٍ من الاستعلامات. هذه الصِّياغةُ سَمَحَت بالاستفادة من التِّقنيَّات المُتقدِّمة المُقترَحة في حالة التِّقنيِّة الصِناعيَّة السابقة من أجل حلِّ مُشكل الأَمْثَلَةِ مُتعدِّدَةِ الأهداف. لهذا، في أوَّلِ الأمر قُمنا بتطوير نماذجِ التَّكلُفَة لتقدير تكلُفة الطَّاقة اللاَّزمة لتشغيل الاستعلامات، بطريقة معزولةٍ أو متوازيةٍ. بعد ذلك، قمنا بدَمج هذه النَّماذج في واحِدَة من أهمِّ الوحدات في نُظم إدارة قواعد البيانات، ألا وهي وِحدة معالجة الاستعلام. الهدف الجديد لهذه الوِحدة هو اختيارُ خُطَطِ تنفيذ الاستعلام مع الأخذ بعيْن الاعتبار لمجموعةٍ من قِيَمِ التَّسْوِية والمُقايضة بيْن وقت التشغيل والطَّاقة المُستهلكة، هذه القِيمُ المستخدمون هُمُ المسؤولون عن إدخالها. أيضا، اقترحنا صيغَةً لدمجِ البُعدِ الطَّاقويِّ في مرحلة التصميم الماديِّ لقواعد البيانات، وذلك عن طريق اختيار هيَاكِلِ الأَمْثَلَةِ مع الأخذ بعين الاعتبار جانِبَ استهلاك الطَّاقة. نتيجة لذلك، قمنا بدراسة تقنيِّة حِفظ نتائج الاستعلامات، والتي تُعدُّ واحدةً من أهمِّ هياكِل الأمثَلة المُستعملة بكثرة. في كلِّ مساهمة من مساهمات هذه الأُطروحةِ، قمنا بإجراء تجاربَ واسعةِ النِّطاق باستخدام جهاز فعليٍّ لقياس استهلاك القُدرة الكهربائيَّة للخادِم، واعتمادًا على البيانات والاستعلامات الخاصَّة بمقاييسِ كلٍّ من TPC-DS، TPC-H وSBB مع الحِرص على تنْويع المكوِّنات المادِّية والبرمجيَّة في كلِّ تَجربة.

الكلمات المفتاحيَّة: كفاءةُ استخدام الطَّاقة، نماذجُ التَّكلفة، معالَجَةُ الاستعلام، التَّصميم الماديُّ، إدارة الطَّاقة، أمثَلةٌ متعدِّدة الأهداف.

Abstract

The physical design is one of the crucial phases of advanced database design life cycle. This is due to its important role in selecting optimization structures such as materialized views, indexes, and partitioning to speed up the performance of queries. This phase has been amplified by the continually needs of storing and managing in efficient way the deluge of data in storage systems. This situation motivates the editors of commercial and non-commercial Database Management Systems (e.g. SQL Tuning Advisor - Oracle and Parinda -PostgreSQL) to propose tools (called advisors) to assist database administrators in their tasks when selecting their relevant optimization structures for a given database/data warehouse schema and a workload. The maturity of research performed in the physical design motivates us to go further and capitalize the knowledge and expertise in terms of processes, the algorithms, the cost models used to quantify the benefit of the selected optimization structures, etc. used by the research community. In this paper, we first propose a physical design language that allows describing all inputs and outputs of the physical design phase. Secondly, we propose a a generale interactive approach to select a set of physical structure by intergrating user preference. Finally, a case study of our contribution is presented to stress our approache and highlights its importance.

Keywords : Database ; physical design ; advisors ; optimization structures, model-driven engineering. and software configurations.

Remerciements

C'est avec grand plaisir que je réserve cette page, en signe de gratitude et de reconnaissance à tous ceux qui m'ont aidé à la réalisation de ce travail.

Je remercie, tout d'abord, **Abdelkader OUARED** pour sa rigueur et la pertinence de ses jugements qui ont été très constructifs et m'ont permis de faire ce travail. Je lui suis également reconnaissant pour sa contribution à l'amélioration judicieuse de la qualité de ce document.

Je remercie cordialement **Mme LAKHDARI Aicha** d'avoir acceptée d'être rapporteur de cette thèse de master. Je souhaite adresser également mes remerciements à **Mr. CHIKHAOUI AHMED** d'avoir fait l'insigne honneur d'accepter d'examiner mon travail ;

Je voudrais exprimer à mes proches toute ma gratitude : mes très chers parents, mon frère et mes sœurs, ma belle femme, pour leur soutien moral et mon fils adorable Anes.

Et enfin, merci à tous ceux qui ont participé de près ou de loin à l'aboutissement de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Première partie

Introduction Générale

Chapitre

1

Introduction Générale

Sommaire

"Nothing is particularly hard if you divide it into small jobs".
- Henry Ford (1863-1947)

1.1 Introduction Générale

1.1.1 Contexte et Motivation

De nos jours, l'informatique décisionnelle est devenue un outil stratégique décisif pour la gestion d'entreprise. Les systèmes décisionnels permettent aux organisations l'exploitation des données dans le but de faciliter la prise de décision, or ses derniers manipulent de très importantes masses de données émanant de différentes sources hétérogènes et distribuées, archivées dans un socle informatique.

Les base de données sont souvent modélisés par un schéma permet d'organiser les données afin de facilitant leur analyse. Ce schéma est caractérisé par un ensemble de tables qui contient des attributs et des clés étrangères.

Pour procéder à l'interrogation de cet ensemble gigantesque de données, des requêtes très complexes doivent être déployées sur la base de données, ceci dit, ces requêtes sont caractérisées par des opérations de sélection sur les tables de base de doonées $\mathcal{BD}$, suivies de jointures. Sans techniques d'optimisation, le coût d'exécution de cette interrogation peut être très important (des heures voir des jours), parmi elles, on trouve les techniques redondantes comme *les index*, *les vues matérialisées* et *la fragmentation verticale*, et les techniques non redondantes comme *la fragmentation horizontale* et *le traitement parallèle*; durant la conception de la base de données, l'administrateur à le choix de sélectionner un ensemble de techniques d'optimisation de manière isolée ou combinée en tenant compte du degré de similarité entre ces dernières.

1.1.2 Problématique

Les principales caractéristiques des base de données décisionnelles sont leur grande taille et la complexité des requêtes qu'ils exploitent. Cette complexité est due aux opérations de jointure et d'agrégation utilisées par les requêtes de jointure en étoile (dites aussi OLAP [1]), qui détériorent de manière significative les performances de la base de données lors d'une interrogation portant sur de gros volumes de données.

L'administrateur de base de données utilise plusieurs techniques d'optimisation pour réduire le temps de réponse des requêtes. Pour utiliser ces techniques, l'administrateur est confronté à un certain nombre de problèmes. Il pose souvent les questions suivantes :

- Une seule technique est-elle suffisante ? Si oui laquelle ?
- Sinon, quelles sont les techniques les plus intéressantes pour ma base de données ?
- Est-ce que ces techniques sont dépendantes ?
- Pour chaque technique, quelles sont les instances intéressantes pour ma base de données ?

La sélection des techniques d'optimisation, a été largement étudiée dans le but d'offrir à l'administrateur une meilleure optimisation de sa base de données. Chacune d'elles présente un problème de sélection classé comme NP-Complet, et leur combinaison (séquentielle ou conjointe) augmente la complexité du problème bien qu'elle peut offrir de meilleures performances. Plusieurs travaux ont traité ces problèmes mais présentent certaines insuffisances comme, le contrôle du nombre de fragments générés, la gestion de l'espace de stockage alloué pour les index ou pour les vues matérialisées.De plus, peu de travaux ont été réalisés sur plus de deux techniques car la plupart des travaux considèrent seulement deux techniques. Il est intéressant donc de proposer des démarches permettant de combiner le maximum de techniques afin d'optimiser au mieux la base de données.

1.1.3 Objectif

L'objectif principal de notre travail vise à proposer un ensemble de méthodes d'optimisation et de gestion de base de données relationnel modélisé par un schéma. Ces méthodes prennent en compte trois techniques d'optimisation, à savoir la fragmentation horizontale, les index de jointure binaire, et les vues matérialisées. Nous visons aussi le développement d'un outil d'administration assistant les administrateurs dans leurs tâches d'optimisation. Plus concrètement, les méthodes d'optimisation proposées s'articulent autour des grands axes suivant :

1. Analyse de domaine des adviors des structures de physiques des base de données.
2. Proposition d'une approche de sélection d'une configuration d'index des base de données.
3. Développement d'un outil d'aide à l'administration de la conception physique.

1.1.4 Plan de mémoire

Ce document est divisé en deux parties principales.

- **La Partie I** présente une synthèse bibliographique, elle comporte les deux chapitres suivantes :
  - Le chapitre 1 décrit une synthése bibliographique sur la conception physique de base données.

1. anglais OnLine Analytical Processing, OLAP

- le chapitre 2 présente une synthése bibliographique sur les techniques d'optimisation.

— **La Partie II** concerne notre approche proposée, elle comporte les trois sections suivantes :
- Le chapitre 3 décrit notre contribution.
  - la section 1 correspond à la description formelle des dépendances des entités, c'est-à-dire les dimensions d'un advisor.
  - la section 2 correspond à la description de langage de description des advisors.
  - la section 3 correspond à l'usage de notre méta-modèle.
- Le chapitre 4 décrit notre outil prototype.

— Enfin,nous clôturons ce mémoire par une conclusion générale et les perspectives liées à ce projet de fin d'études.

La (Figure 1) schématise les grands axes de ce travail et leur répartition dans chaque chapitre.

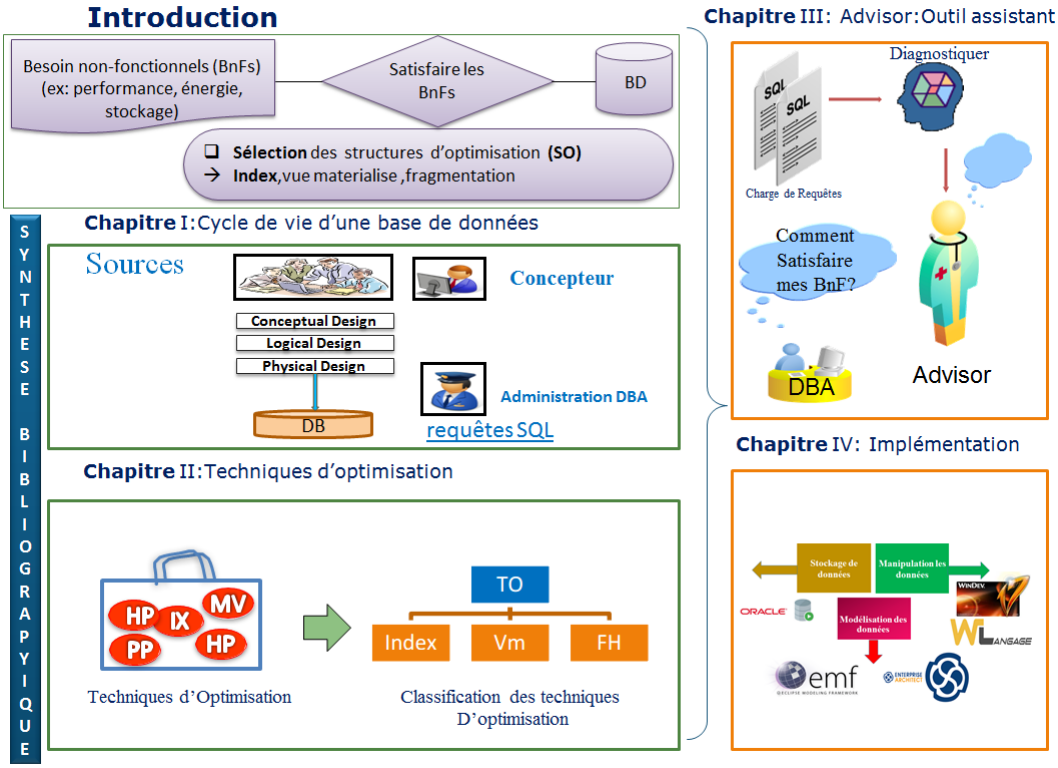


FIGURE 1 – La répartition des chapitres de manuscrit

Deuxième partie

Synthèse Bibliographique

Chapitre

2

Conception Physique des Bases de Données

Sommaire

2.1 Introduction

Dans ce chapitre, nous présentons une synthése bibliographique portant sur la technologie des bases de données, son historique, son cycle de vie de conception et d'exploitation. Nous nous focalisons particulièrement sur la phase de conception physique et le traitement des requêtes. Nous allons détailler les différentes techniques utilisées et travaux proposés pour chacune d'entre elles. En second lieu, nous abordons les problèmes d'optimisation en citant les méthodes de résolution proposées dans le cas des bases de données.

2.2 Cycle de vie des bases de données

Une base de données est un ensemble structuré de données enregistrées sur des supports accessibles par ordinateur, représentant des informations du monde réel et pouvant être interrogées et mises à jour par une communauté d'utilisateurs.

La gestion et l'accès à une base de données sont assurés par SGBD. Un Système de Gestion de Bases de Données est un logiciel de haut niveau permettant aux utilisateurs de structurer, d'insérer, de modifier, de rechercher de manière efficace des données spécifiques, au sein d'une grande quantité d'informations, stockées sur mémoires secondaires partagée de manière transparente par plusieurs utilisateurs.

Suite à la progression de la technologie dans les domaines des processeurs, de la mémoire, du stockage et des réseaux informatiques, les tailles, les capacités et les performances des bases de données et SGBD ont augmenté en ordre de grandeur. Le développement de la technologie de base de données peut être divisé en trois périodes basées sur le modèle ou la structure de données : navigationnelle, relationnelle et post-relationnelle. Une illustration de l'évolution de la technique des bases de données est donnée [5] dans la(Figure 2.1)

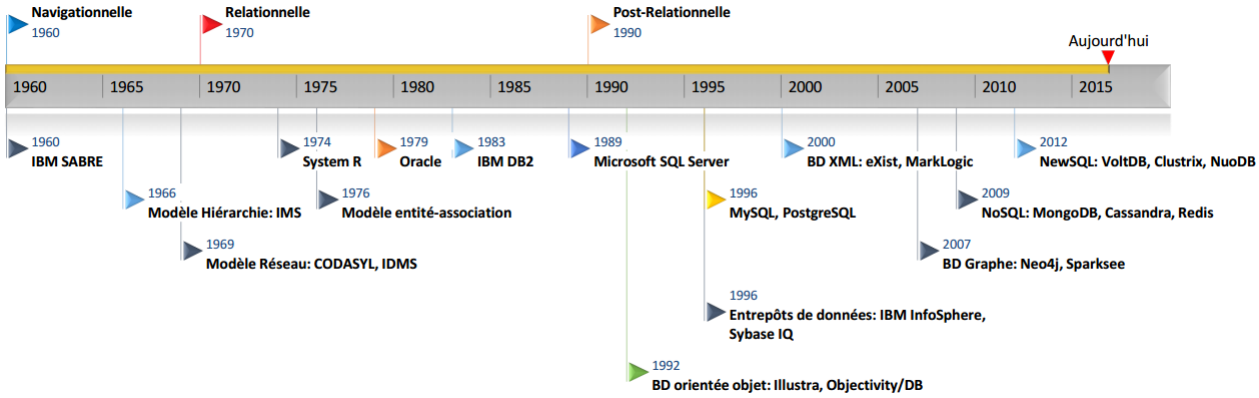


FIGURE 2.1 – Une brève évolution de la technologie des bases de données.

2.2.1 Évolution verticale des modèles de données

L'évolution des modèles de données dite "verticale" (voir Figure 2.2) a été unifiée dans l'architecture ANSI/SPARC [23] qui est actuellement universellement admise dans la communauté des BD.

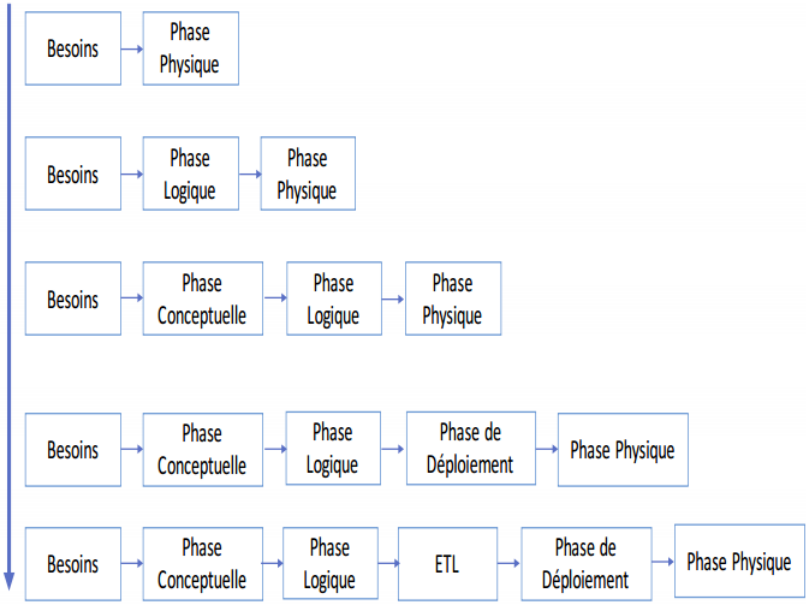


FIGURE 2.2 – Évolution verticale du cycle de conception des BD

Cette architecture distingue les trois niveaux d'abstraction : le niveau conceptuel, externe et interne (voir Figure 2.3) :

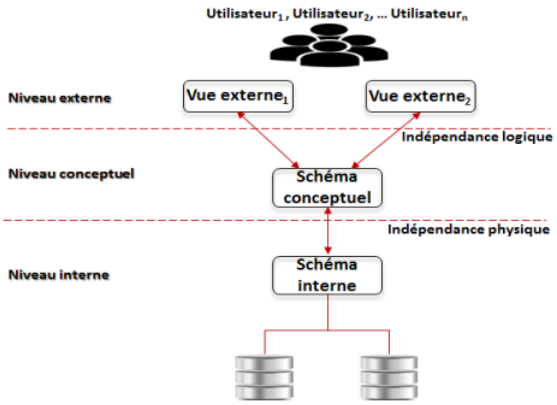


FIGURE 2.3 – L'architecture ANSI/SPARC

Le cycle de vie de la base de donnée intègre les étapes de base de la conception d'une base de données logique à partir de la modélisation conceptuelle des besoins des utilisateurs, des définitions de table spécifique au système de gestion de bases de données et une base physique indexée, partitionnée, regroupée et matérialisée de manière sélective pour maximiser la performance réelle.

Pour une base de données distribuée, *la conception physique* implique également l'allocation de données à travers un réseau informatique. Le cycle de vie de la base de données comporte les quatre phases suivantes comme illustrer dans (Figure 2.4) :

— Définition des besoins des utilisateurs,

— Modélisation conceptuelle,
— Modélisation logique,
— Modélisation physique.
— Déploiement et maintenance

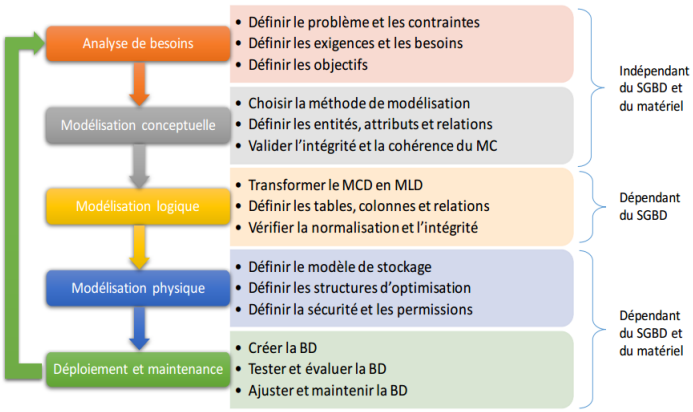


FIGURE 2.4 – Les phases du cycle de vie et leurs rôles

2.3 Niveau de modélisation

Le domaine relatif à la définition des besoins des utilisateurs est le domaine de l'ingénierie des besoins (IB). Rolland [22] définit **l'IB** comme suit : *ingénierie de besoins est concernée par l'identification de buts assignés au système envisagé et l'opérationnalisation de ces buts en contraintes et exigences imposées au système et aux agents qui en assureront le fonctionnement.*

L'IB peut être vue comme le processus qui permet de transformer une idée floue en spécification précise des besoins servant de support à la spécification du système et de ses interfaces avec l'environnement. Cette étape de *définition des besoins* consiste à définir de manière détaillée, l'ensemble des besoins et exigences des utilisateurs relatifs aux données et aux traitements [13]. En effet, parallèlement à la spécification des exigences relatives aux données, il est nécessaire de définir les exigences (fonctionnelles et non fonctionnelles) précisant l'ensemble des opérations et traitements que devra supporter l'application de BD. La phase de définition des besoins est assurée par un analyste ou un expert en ingénierie des besoins. Cette phase nécessite l'accomplissement de certaines tâches préliminaires comme :

— L'identification de la portée de l'application de BD ainsi que de l'ensemble de ses utilisateurs
— L'étude de l'environnement du système incluant l'analyse des types de traitements et de leur fréquence
— L'analyse des flux d'informations du système
— Des entrées et sorties des traitements
— Des caractéristiques géographiques des utilisateurs
— L'étude et l'analyse de la documentation existante relative à l'application de la BD et l'établissement des questionnaires destinés aux utilisateurs.

La phase de définition des besoins comprend les quatre étapes suivantes [15] :

1. **La collecte des besoins** : les besoins sont collectés auprès des utilisateurs finaux ou des clients du système. Plusieurs techniques de collecte des besoins peuvent être utilisées comme les interviews, les réunions ou les workshops.
2. **La spécification des besoins** : consiste à formaliser les besoins et à identifier leurs caractéristiques attendues. L'étape de collecte des besoins fournit généralement un premier ensemble de besoins informels, inconsistants ou incomplets. Trois principales techniques de spécification des besoins sont utilisées pour spécifier les besoins collectés d'une manière plus structurée [16] : la (Figure 2.2) montre l'intégration des besoins dans les différentes niveaux de modélisation de base de données.
3. **La validation des besoins** : consiste à valider le modèle obtenu lors de l'étape de spécification des besoins par les experts du domaine et les utilisateurs. Cette phase permet d'éviter la propagation des erreurs ou inconsistances des besoins durant les étapes de conception et d'implémentation de la base, qui peuvent s'avérer très coûteuses à corriger par la suite. Des outils et langages peuvent être utilisés pour faciliter les tâches de validation des besoins. Des langages formels comme le langage OCL ou le langage Z sont utilisés pour compléter la spécification des besoins afin de vérifier leur consistance [13].
4. **La gestion des besoins** : inclut toutes les activités permettant d'établir et de maintenir l'intégrité des besoins pendant que l'application de BD évolue. Quelques outils de traçabilité sont disponibles pour vérifier la traçabilité des besoins dans le cas où ces derniers évoluent très fréquemment.

2.3.1 Niveau Conceptuel

Cette phase prend comme entrée la spécification des besoins collectés auprès des utilisateurs du système. Cette phase est assurée par les analystes et concepteurs du système. Elle comprend deux principales étapes menées en parallèle : la conception du modèle conceptuel et la conception de l'application et des transactions [13].

- La conception du modèle conceptuel : cette étape consiste à élaborer le modèle conceptuel des données (MCD) décrivant les exigences des utilisateurs relatives aux données. Ce modèle comprend une description détaillée de la structure de la BD, exprimée en utilisant les concepts fournis par un modèle de données d'un haut niveau d'abstraction décrit indépendamment de toute contrainte d'implémentation. Plusieurs exemples de modèles de haut niveau peuvent être utilisés comme le modèle E/A, le digramme de classes d'UML, le modèle Express, etc. L'établissement de ce modèle conceptuel repose généralement sur un dictionnaire regroupant le détail de l'ensemble des données manipulées par le système.
- La conception de l'application de BD : les opérations et les transactions définies à partir des exigences des utilisateurs relatives aux traitements, sont utilisées pour spécifier les requêtes des utilisateurs de haut niveau. Cette étape permet de vérifier si le schéma conceptuel défini répond à toutes les exigences identifiées [13]. Les opérations peuvent être de trois différents types [13] : les opérations de restitution des données, les opérations de mise à jour et les opérations combinant la restitution et les mises à jour. Ces opérations doivent être spécifiées dès le niveau conceptuel, en identifiant les E/S de chaque opération et leur comportement fonctionnel [13]. Cette étape de conception peut inclure une tâche d'identification des processus qui consistent

en un ensemble d'opérations complexes. Des outils et notations sont utilisés pour spécifier les processus comme certains diagrammes UML (comme le diagramme d'activité et le diagramme de séquence), ou certains modèles de la méthode Merise qui permettent de représenter les traitements de l'application au niveau conceptuel comme le modèle conceptuel de traitements(MCT).

2.3.1.1 Exemple 1

Une société spécialisée dans la vente souhaite dynamiser sa politique de vente en construisant un site web pour la vente de ses produits. Après l'étude des besoins de l'entreprise, le concepteur de BD a créé un schéma conceptuel avec le modèle entité-association. Voici la description de chaque entité :

- Clients : Cette table conserve les informations sur les clients.
- Commandes : Contient la liste des commandes des clients.
- Détail commandes : Contient les informations détaillées sur les commandes.
- Produits : Garde la liste des produits avec leur détaille.
- Stock : Sauvegarde la quantité de chaque produit en stock.

Un exemple de schéma conceptuel défini en termes d'entités et d'associations entre ces entités est représenté dans la (Figure 2.5).

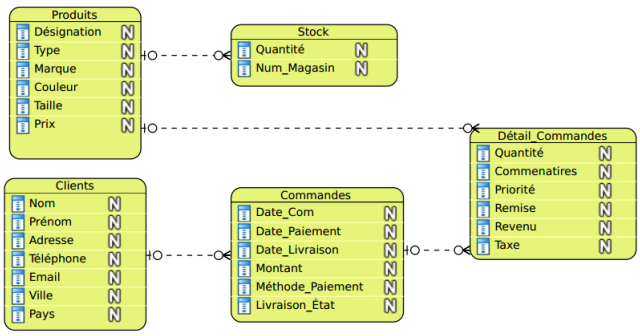


FIGURE 2.5 – Exemple d'un schéma de modélisation conceptuelle.

2.3.2 Niveau Logique

L'étape de modélisation logique prend en entrée le modèle conceptuel des données et fournit en sortie un modèle logique des données. Ce dernier représente l'organisation des données dans un modèle de données qui peut être implémenté. Cette organisation des données est également nommée déploiement logique des données. Le modèle logique de données possède des constructeurs de modélisation ignorants les détails physiques d'implémentation, et qui sont faciles à comprendre par les utilisateurs. La traduction du modèle conceptuel vers un modèle logique se fait de manière directe voire automatique, en suivant des règles de traduction prédéfinies. Certains auteurs considèrent une phase additionnelle consistant en le *Choix du SGBD* comme phase qui précède la phase logique [13]. Ce choix du SGBD peut dépendre de facteurs techniques, économiques et stratégiques, il peut être une contrainte à respecter dès le début de la conception.

L'application de BD peut aussi être conçue de manière générique pour pouvoir être déployée sur

plusieurs plate-formes de déploiement. Le choix du SGBD peut se faire dans ce cas lors de la phase de déploiement où l'administrateur choisit la plate-forme qui répond le mieux aux besoins de l'application. La phase de modélisation logique se fait en deux principales étapes :

1. **Mise en correspondance du modèle** : la première étape consiste à traduire le modèle conceptuel vers un modèle logique indépendamment des caractéristiques du SGBD.
2. **Adaptation du modèle** : une deuxième étape consiste à adapter le schéma logique obtenu aux spécificités (constructeurs de modélisation et contraintes) du ou des SGBD sur lesquels sera implémenté le modèle logique. Le modèle logique est décrit à la fin de cette phase dans un langage de définition des données (LDD), il peut être complété lors de la phase de modélisation physique. Plusieurs outils de conception permettent de générer automatiquement le modèle logique décrit selon un LDD à partir d'un modèle conceptuel de données comme ERwin, BPwin, Rational Rose et Visio.

2.3.2.1 Exemple 2

La (Figure 2.6) représente le schéma logique de notre exemple de gestion de vente. On peut remarquer que la définition des types de données des attributs et les clés primaires ont été établies sur les tables « *Stock* », « *Commandes* » et « *Clients* ».

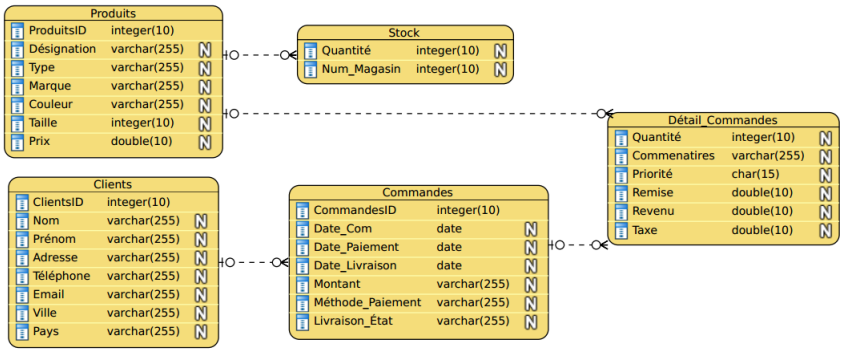


FIGURE 2.6 – Exemple d'un schéma de modélisation logique.

2.3.3 Niveau Physique

La phase de modélisation physique prend en entrée le modèle logique de données décrit selon un LDD et fournit en sortie un *MPD* correspondant aux SGBD choisis pour l'implémentation de la BD. Ce modèle physique est défini dans un langage de définition du stockage *LDS*. Cette phase nécessite une connaissance détaillée des paramètres de configuration, des structures physiques, des types de données ainsi que le langage de définition des données utilisé par le SGBD. Cette phase est généralement pilotée par l'administrateur du système *DBA*. Le but de *la conception physique* consiste à fournir une variété de choix pour le stockage des données en termes de regroupement, de partitionnement, d'indexation, etc [19]. Les choix des structures se font selon plusieurs critères comme le temps de réponse aux requêtes, l'espace de stockage utilisé par les fichiers de la base ou le débit moyen des transactions. Les requêtes et

transactions sont dans ce cas celles identifiées et spécifiées lors de la phase de définition des besoins et de modélisation conceptuelle.

2.3.3.1 Exemple 3

La (Figure 2.7) représente le schéma physique de notre exemple de gestion de vente. On peut remarquer que la définition des types de données des attributs et les clés primaires ont été établies sur les tables « *Stock* », « *Commandes* » et « *Clients* ».

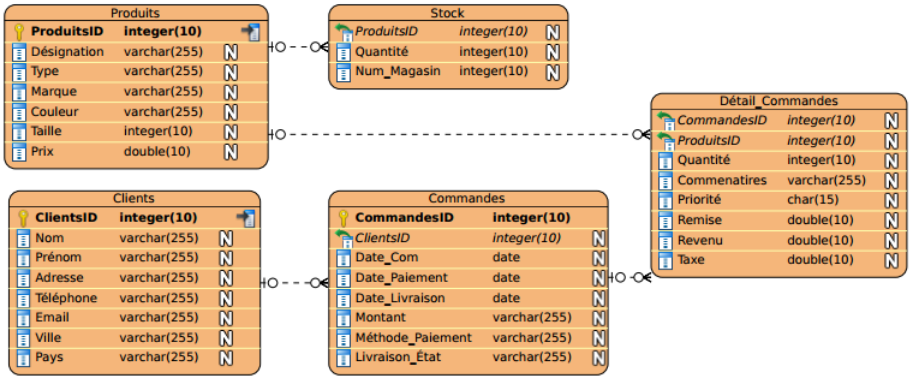


FIGURE 2.7 – Exemple d'un schéma de modélisation physique

Afin de respecter les trois niveaux imposés par l'architecture ANSI/SPARC, deux principaux langages de données sont utilisés pour assurer l'indépendance logique et physique des données :

Le *LDD* est utilisé pour spécifier le schéma conceptuel et logique, le *LDS* est utilisé pour spécifier le schéma interne. Dans la plupart des SGBD actuels, il n'existe pas de langage spécifique qui joue le rôle de LDS. Le schéma interne est alors défini par une combinaison de fonctions, de paramètres et de spécifications relatives au stockage des données. Un troisième langage devrait également exister pour définir les vues des utilisateurs ; mais la plupart des SGBD utilisent le LDD pour définir les schémas conceptuels et externes. Dans les SGBD actuels, les trois types de langages cités ne sont généralement pas considérés comme des langages distincts, mais un langage global et intégré est utilisé qui comporte des constructeurs pour la définition des différents schémas.

Pour l'exemple des BD relationnelles, le langage SQL est utilisé en tant que langage de définition des données, de définition des vues et de manipulation des données. Le LDS est un langage qui existait dans les premières versions de SQL, mais a été retiré du langage SQL afin de le maintenir aux deux niveaux conceptuel et externe uniquement [13].

2.3.4 La phase de déploiement

Durant cette phase le choix du SGBD et sa plateforme est identifié [21]. Dans la première génération de conception de BD, cette phase n'était pas bien identifié. Certains chercheurs considéraient que le choix du SGBD doit précéder la phase logique. Notons que ce dernier peut dépendre de facteurs techniques, économiques et stratégiques, il peut être une contrainte à respecter dès le début de la

conception. L'application de BD peut aussi être conçue de manière générique pour pouvoir être déployée sur plusieurs plateformes de déploiement. Le choix du SGBD et sa plateforme (voir Figure 2.9) peut se faire dans ce cas lors de la phase de déploiement

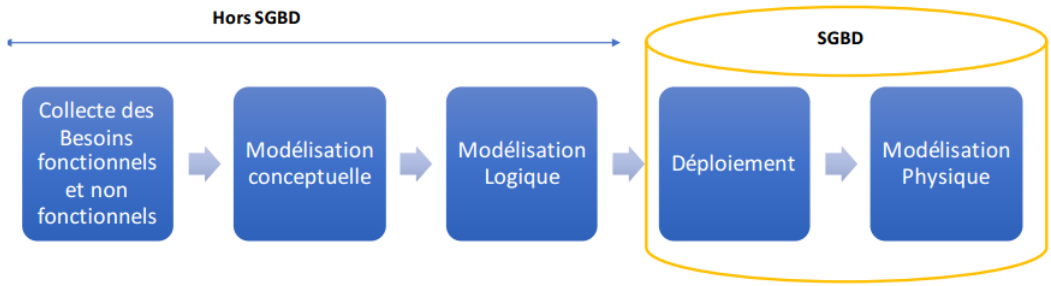


FIGURE 2.8 – Le choix du SGBD et sa plateforme

l'administrateur choisit la plateforme qui répond le mieux aux besoins de l'application (Figure 2.8). La phase de déploiement est devenue de plus en plus complexe avec l'arrivée des machines parallèles et distribués.

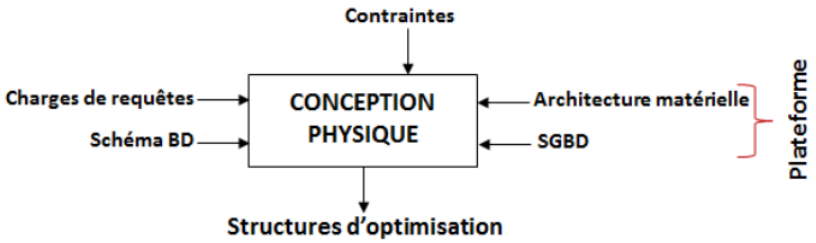


FIGURE 2.9 – La place de la phase de déploiement dans le cycle de vie

2.4 La conception physique

Après l'établissement d'un cahier des charges qui décrit les besoins fonctionnels des utilisateurs, une traduction des besoins auxquels sera dédié à la base de donnée,en un modèle conceptuel puis logique, est effectuée. Ce modèle doit être implémenté physiquement en prenant en compte les structures de données à utiliser et les contraintes de déploiement.

La conception physique est la phase centrale qui consiste a établir une **configuration physique** sur la base de donnée, elle implique des dizaine et souvent des centaine des variable qui sont difficile a suivre.

Dans cette section nous allons définir la phase de conception physique ensuite le processus d'exécutions d'une requête aprés nous allons expliquer le rôle de DBA.

2.4.1 Objectif

La conception physique d'une base de données consiste à établir une configuration physique sur le support de stockage. Cela comprend la spécification détaillée des éléments de données,la sélection des techniques d'optimisation et le type de selection de ces technique . La selection des techniques est le cœur de la conception physique [9]. Dans la première génération de moteurs d'exécution de requêtes dédiés aux bases de données traditionnelles, la conception physique n'avait pas autant d'importance. Aujourd'hui face à la complexité des requêtes à traiter et le volume important des données, la conception physique a reçu une importance phénoménale

L'étape de conception physique de la base de données implique la sélection d'index, le partitionnement, le regroupement et la matérialisation sélective des données.Elle commence après que les tables SQL ont été définies et normalisées. Il se concentre sur les méthodes de stockage et d'accès à ces tables sur le disque pour que la BD fonctionne. avec une efficacité élevée. L'objectif de la conception physique est de maximiser les performances de la base de données sur l'ensemble du spectre des applications écrites sur elle.

2.4.2 Les entrées sorties de la conception physique

Durant la phase de conception physique, le DBA doit sélectionner un ensemble de structures d'optimisation pour satisfaire ses requêtes . la conception physique consiste, à partir d'une charge de requêtes et un ensemble de contraintes, à sélectionner un ensemble de techniques d'optimisation vérifiant les contraintes en entrée et minimisant le coût d'exécution de la charge de requêtes. Cette conception est à la charge de DBA, qui doit avoir une parfaite connaissance de tous les paramètres indispensables pour la mise en œuvre de la base de donnée. La (Figure 2.10) montre les entrées est les sorties de la *CP*.

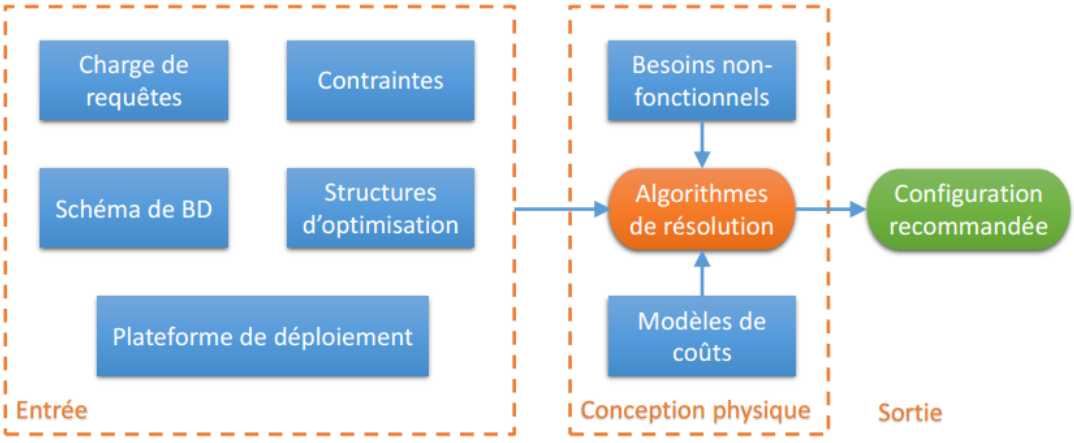


FIGURE 2.10 – Les entrés et les sorties de la *CP*

2.4.3 Le processus d'exécution des requêtes SQL

Avec la croissance du volume de données et la complexité des requêtes, la phase de conception physique est devenue primordiale pour les systèmes de bases de données. Selon Chaudhuri S. et Narasayya V [12]. La CP n'est nullement un problème nouveau, mais elle reste toujours un problème ouvert. Son importance majeure est de déterminer la façon dont une requête peut s'exécuter efficacement sur le système en exploitant les capacités de l'optimiseur et du moteur d'exécution. Cette importance s'est amplifiée car les optimiseurs deviennent de plus en plus sophistiqués afin de répondre aux besoins des utilisateurs finals.

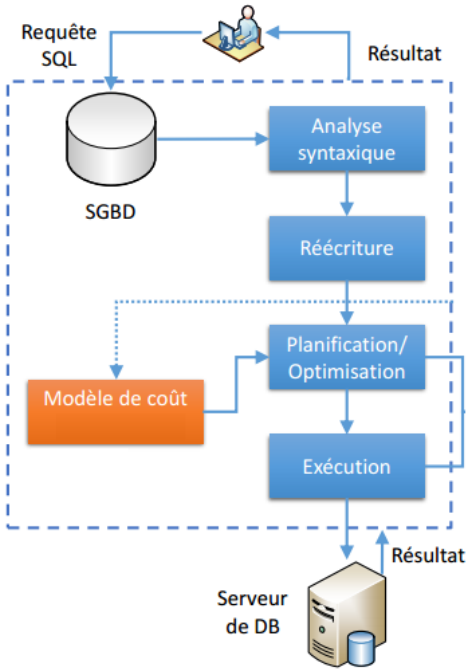


FIGURE 2.11 – Processus d'exécution du requête

Le traitement et l'optimisation des requêtes ont toujours été l'un des éléments importants de la technologie des bases de données. Cette composante traite principalement des données souhaitées par l'utilisateur à partir d'une base de données souvent importante et renvoie efficacement les résultats avec une précision acceptable. La Figure11 représente le processus du moteur de requête. Lorsque le système reçoit une requête SQL, le processeur de requêtes vérifie d'abord la justesse de la requête SQL (par exemple, si la syntaxe de requête est correcte, si les relations et les attributs sont stockés dans la base de données, etc.) Si la requête est acceptable. l'optimiseur de requêtes est utilisé pour identifier uniquement le meilleur plan de requête logique. Le plan de requête logique doit être transformé en un plan d'exécution physique.

2.4.3.1 Analyseur

Avant de commencer le traitement de la requête, le moteur de requête doit d'abord analyser la requête, qui construit une structure arborescente à partir de la forme textuelle de la requête Le tra-

vail de l'analyseur est de prendre du texte écrit dans une langue comme SQL et de le convertir en un arbre d'analyse, qui est un arbre dont les nœuds correspondent soit à : (1) *atomes*, qui sont des éléments lexicaux tels que des mots clés (par exemple, SELECT), des noms d'attributs ou de relations, des constantes, des parenthèses, des opérateurs tels que + ou <, et d'autres éléments de schéma, ou (2) des catégories syntaxiques, qui sont des noms pour des sous-parties de requête qui ont un rôle similaire. Ils sont représentés par un nom descriptif entre les symboles < et >. Par exemple, <Requête> est utilisé pour représenter une requête, et <Condition> représentera toute expression qui est une condition. La (Figure 2.12) illustre un exemple d'un arbre d'analyse.

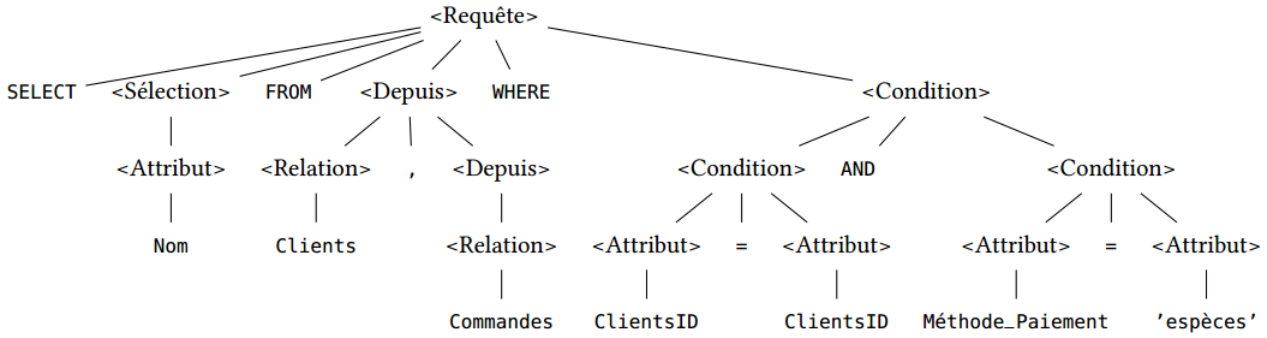


FIGURE 2.12 – Exemple d'un arbre d'analyse.

2.4.3.2 Contrôleur

La fonction principale de la Réécriture des requêtes est de traduire l'arbre d'analyse généré par l'analyseur dans un plan de requête initial. La vérification sémantique n'est pas nécessaire si le SQL est bien formé avant qu'il ne soit traité. (En fait, même si la requête est valable syntaxiquement, elle peut violer une ou plusieurs règles sémantiques sur l'utilisation des noms). la fonction de vérification sémantique est utilisé pour générer un arbre d'analyse valide. Si l'arbre d'analyse n'est pas valide, un diagnostic approprié est émis et aucun autre traitement ne se produit.

Tous les attributs doivent être d'un type approprié à leurs utilisations. De même, les opérateurs sont vérifiés pour vérifier qu'ils s'appliquent aux valeurs des types appropriés et compatibles. La fonction de vérification des types doit s'assurer que ce type de données existe ou non.

- **Le contrôle de la relation utilise** : Chaque relation mentionnée dans une clause FROM doit être une relation ou une vue dans le schéma contre lequel la requête est exécutée.
- **Le contrôle des attributs** : Chaque attribut mentionné dans la clause SELECT ou WHERE doit être un attribut d'une certaine relation dans la portée courante. Bien que la statistique de requête SQL soit une syntaxe valide, mais elle peut en fait violer certaines règles sémantiques sur l'utilisation des noms. Par exemple, une requête SQL simple comme « SELECT A FROM B WHERE A = 7 » On peut considérer que la syntaxe de cette requête est valable, mais si l'attribut A n'appartient pas à la relation B, cela provoquera l'erreur. Si l'utilisateur a plus d'expériences avec la base de données, il peut éviter de commettre des erreurs.
- **L'expression de l'algèbre relationnelle** : Il transforme les arbres d'analyse SQL en plans de requêtes logiques algébriques. Par exemple, il convertit une simple construction SELECT-FROM-

WHERE (SFW) en algèbre relationnelle. Si nous avons une <Requête> qui est une construction <SFW>, et que la <Condition> de cette construction n'a pas de sous-requêtes.

2.4.3.3 Optimiseur

L'optimiseur détermine le moyen le plus efficace d'exécuter une instruction SQL. Il s'agit d'une étape importante dans le traitement de toute instruction de langage de manipulation de données (DML) : SELECT , INSERT , UPDATE ou DELETE . Il existe souvent de nombreuses façons d'exécuter une instruction SQL ; Par exemple, en modifiant l'ordre dans lequel les tables ou les index sont accédés. La procédure utilisée par Oracle pour exécuter une instruction peut grandement affecter la rapidité avec laquelle l'instruction s'exécute.

L'optimiseur considère plusieurs facteurs parmi les chemins d'accès alternatifs. Il peut utiliser soit une approche basée sur les coûts (CBO) ou basée sur des règles (RBO) (voir Figure 2.13).

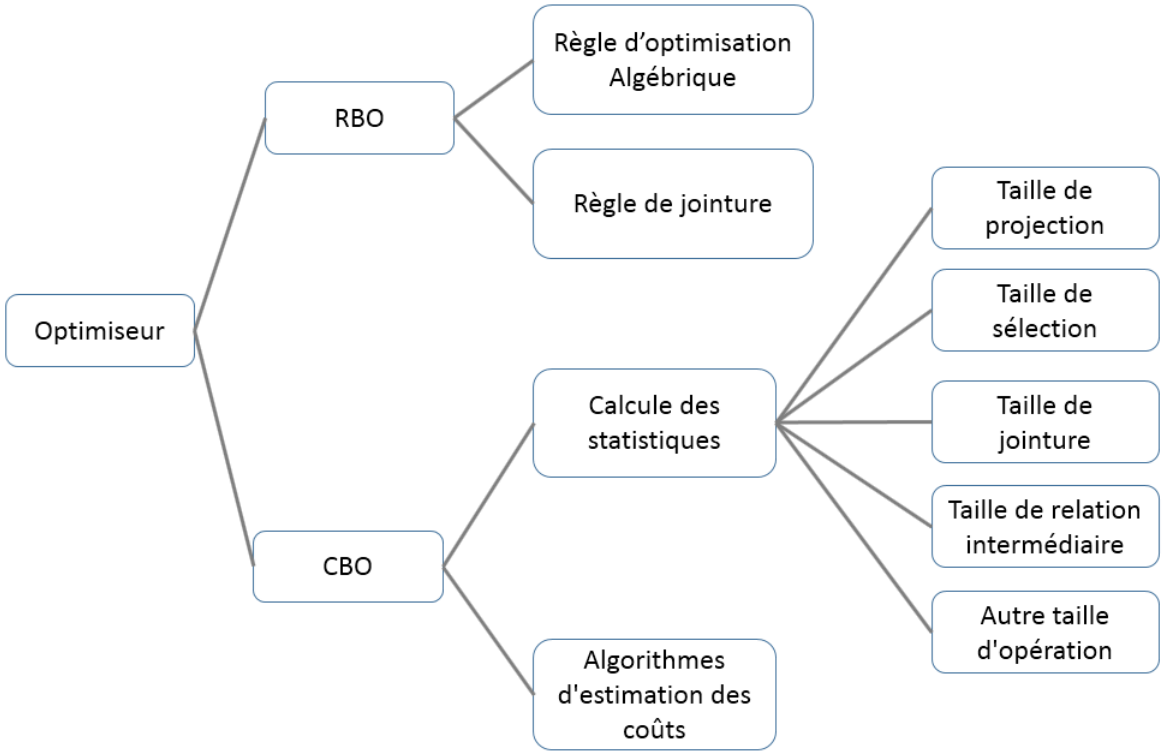


FIGURE 2.13 – Les deux modes d'optimisation.

1. **Optimiseur basé sur les règles *RBO*** : Il est très élégant pour sa simplicité et a souvent fait des choix d'exécution plus rapide que le CBO. Étant donné que le coût de la CBO est très coûteux, de nombreux systèmes utilisent la RBO pour réduire le nombre de choix qui doivent être faits en fonction des coûts. RBO utilise une méthode heuristique pour sélectionner parmi les chemins d'accès alternatifs à l'aide de certaines règles d'optimisation algébriques. Tous les chemins possibles ont été classés et choisis le plus bas.
2. **Optimiseur basé sur les coûts *CBO*** : Il détermine quel plan d'exécution est le plus efficace en

considérant les chemins d'accès disponibles et en factorisant les informations basées sur des statistiques pour les tables ou les index auxquels l'instruction SQL accède. Le CBO effectue les étapes suivantes :

- L'optimiseur génère un ensemble de plans potentiels pour l'instruction SQL, en fonction des chemins d'accès disponibles et des HINT.
- L'optimiseur estime le coût de chaque plan en fonction des statistiques du dictionnaire de données pour la distribution des données et les caractéristiques de stockage des tables, des index et des partitions auxquelles l'instruction accède. Le coût est une valeur estimée proportionnelle à l'utilisation de ressources attendue nécessaire pour exécuter la déclaration avec un plan particulier. L'optimiseur calcule le coût des chemins d'accès et des ordres de jointure, en fonction des ressources informatiques estimées, y compris les E/S, la CPU et la mémoire.
- L'optimiseur compare les coûts des plans et choisit celui avec le coût le plus bas.

Le CBO est composé de trois éléments suivants :

1. Le transformateur de requête : l'entrée de transformateur de requête est une requête analysé, l'objectif principal de transformateur de requête est de déterminer s'il est avantageux de modifier la forme de la requête, il permet la génération de meilleur plan de requête
2. Estimation : est le cœur de CBO elle estime trois type de mesure (sélectivité, cardinalité, coût) pour la sélectivité elle est située dans la plage de valeur (0.0, 1.0) Une sélectivité de 0.0 signifie qu'aucune ligne ne sera sélectionnée dans un ensemble de lignes et qu'une sélectivité de 1,0 signifie que toutes les lignes seront sélectionnées. Pour la cardinalité c'est le nombre de ligne dans une table. Et le coût représente une estimation du nombre d'E/S de disque.
3. Générateur de plan : la principale fonction du générateur de plan est d'essayer différents plan possible pour une requête donnée et de choisir l'optimum.

- ***Statistiques de système*** Le coût de l'évaluation des requêtes peut être mesuré en fonction d'un certain nombre de ressources système différentes. Les ressources ou paramètres les plus importants sont les suivants :
  - **Coût d'accès au stockage secondaire**. C'est le coût du transfert (lecture et écriture) des blocs de données entre le stockage de disque secondaire et les tampons de mémoire principale. Cela est également connu sous le nom de coût d'E/S (entrée/sortie) ou CES. Le coût de la recherche d'enregistrements dans un fichier disque dépend du type de structures d'accès sur ce fichier et la nature d'allocation des blocs de fichiers.
  - **Coût de stockage sur disque**. C'est le coût de stockage sur disque de tous les fichiers intermédiaires générés par une stratégie d'exécution de la requête.
  - **Coût de calcul**. C'est le coût d'exécution des opérations sur les enregistrements en mémoire tampon pendant l'exécution de la requête. Ces opérations comprennent la recherche, le tri, la fusion d'enregistrements pour une opération de jointure ou de tri et l'exécution de calculs sur les valeurs. Cela est également connu comme le coût de CPU (unité centrale de traitement).
  - **Coût d'utilisation de la mémoire**. C'est le coût correspondant au nombre de tampons mémoire nécessaires pendant l'exécution de la requête.
  - **Coût de la communication**. Il s'agit du coût d'expédition de la requête et de ses résultats à partir du site de base de données vers le site ou le terminal d'où provient la requête. Dans les bases de données distribuées, il inclut également le coût du transfert des tables et des résultats

entre différents ordinateurs au cours de l'évaluation des requêtes.

2.4.3.4 Exécution

La dernière étape du processus est l'exécution de la requête. Dans cette étape, toutes les opérations d'E/S et de CPU indiquées dans le plan physique sont exécutées. Un opérateur physique d'une requête peut être exécuté en mode matérialisé ou *pipeline* et/ou *parallèle*.

Différents algorithmes pour des opérations algébriques impliquent la lecture d'un ou plusieurs fichiers comme entrée, le traité, et ensuite la génération d'un fichier de résultat comme sortie. Si l'opération est mise en œuvre de telle sorte qu'elle produit qu'un tuple à la fois, elle peut être considérée comme un itérateur. Par exemple, une implémentation pour la jointure à boucle imbriquée génère un tuple à la fois comme sortie après chaque jointure. L'interface itérateur comprend généralement les méthodes suivantes :

- **Open()** : Cette méthode initialise l'opérateur en allouant des tampons pour son entrée et sa sortie et en initialisant toutes les structures de données nécessaires à l'opérateur. Il est également utilisé pour transmettre des arguments tels que les conditions de sélection nécessaires pour effectuer l'opération. Il appelle à son tour Open() pour obtenir les arguments dont il a besoin.
- **GetNext()** : Cette méthode appelle le GetNext() sur chacun de ses arguments d'entrée et appelle le code spécifique à l'opération exécutée sur les entrées. Le prochain tuple de sortie généré est renvoyé, et l'état de l'itérateur est mis à jour suivant la quantité d'entrée traitée. Lorsque aucun tuple ne peut être renvoyé, il place une valeur spéciale dans le tampon de sortie.
- **Close()** : Cette méthode met fin à l'itération après la génération de tous les tuples possible ou si le nombre requis/demandé de tuples a été retourné. Elle appelle également Close() sur les arguments de l'itérateur.

Cependant, certains opérateurs physiques ne supportent pas le concept d'interface itérateur et ne peuvent donc pas prendre en charge le pipeline. Comme par exemple l'opérateur de tri et de groupement, car ils ne produisent aucune sortie tant qu'ils n'ont pas consommé leurs entrées complètement Exemple Le consommateur du résultat du pipeline appelle *GetNext*() chaque fois qu'un tuple est nécessaire. Dans le cas d'une projection, il suffit d'appeler GetNext() une fois sur la source des tuples, ensuite projeter ce tuple de manière appropriée et retourner le résultat au consommateur. Pour une sélection c , il peut être nécessaire d'appeler *GetNext*() plusieurs fois à la source, jusqu'à ce qu'un tuple qui satisfait la condition C est trouvé. La(Figure 2.14) illustre ce processus.

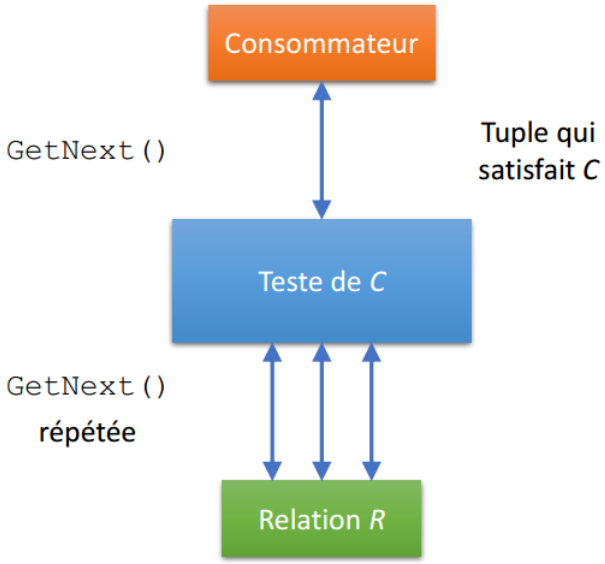


FIGURE 2.14 – Exécution d'une sélection en mode pipeline à l'aide d'itérateurs.

2.4.4 Rôle de L'administrateur de base de données

La combinaison des différents critères (plate-forme, SGBD etc.) engendre des nouveaux problèmes. Une fois la conception physique établie, il faudrait assurer une bonne gestion des structures physiques d'optimisation. En effet, l'architecture physique de la base de donnée doit pouvoir supporter l'ajout de nouvelles données, les opérations de maintenance ne doivent pas altérer l'exécution des requêtes de traitement et l'espace de stockage doit être optimisé.Pour pallier ce problème, l'administrateur effectue périodiquement des réglages sur cette configuration pour améliorer ses performances. Cette phase de réglage est appelée tuning.[9] est définir le tuning comme un processus de réglage continu dans le but d'atteindre une performance maximale de toutes les composantes d'un système de base de données [20]. La (Figure 2.15) montre Les changements effectués durant la phase de tuning qui peuvent consister à :

- Sélectionnée des techniques d'optimisation (les index, les vue matérialisé, les partitionnement, le traitement parallèle,clustiring..ect)
- L'administrateur peut choisir une seule technique d'optimisation (sélection isolée) ou plusieurs (sélection multiple), car chacune a ses propres avantages/inconvénients voire même des similarités avec d'autres techniques. De même, plusieurs solutions s'offrent aux administrateurs car il existe une interaction entre certaines techniques d'optimisation où le changement dans la sélection de la structure dépendante entraine un changement dans la sélection de la structure dominante
- Ajouter d'autres techniques d'optimisation qui permettent de mieux optimiser les performances.
- Chaque technique d'optimisation a plusieurs algorithmes et approches d'implémentation, L'administrateur peut sélectionner des algorithme de ces dernier par exemple : pour les Index (Glouton,motif frequant..ect)[2]
- Réaffecter l'espace de stockage alloué à chaque technique d'optimisation de façon à favoriser les techniques les plus intéressantes.
- Régler les paramètres physiques comme la mémoire qui constitue une tâche de tuning très

importante. Ce réglage consiste à distribuer la mémoire disponible entre plusieurs types de mémoires comme *la mémoire de tri, de hachage, de compilation, les buffers*, etc.

— la sélection des déférents dispositifs matériels est logiciel par exemple Dispositif de stockage (SSD, HDD, NV RAM), Dispositif de traitement(CPU,GPU,FPGA), Modèle de stockage(Columm Store, Row store).

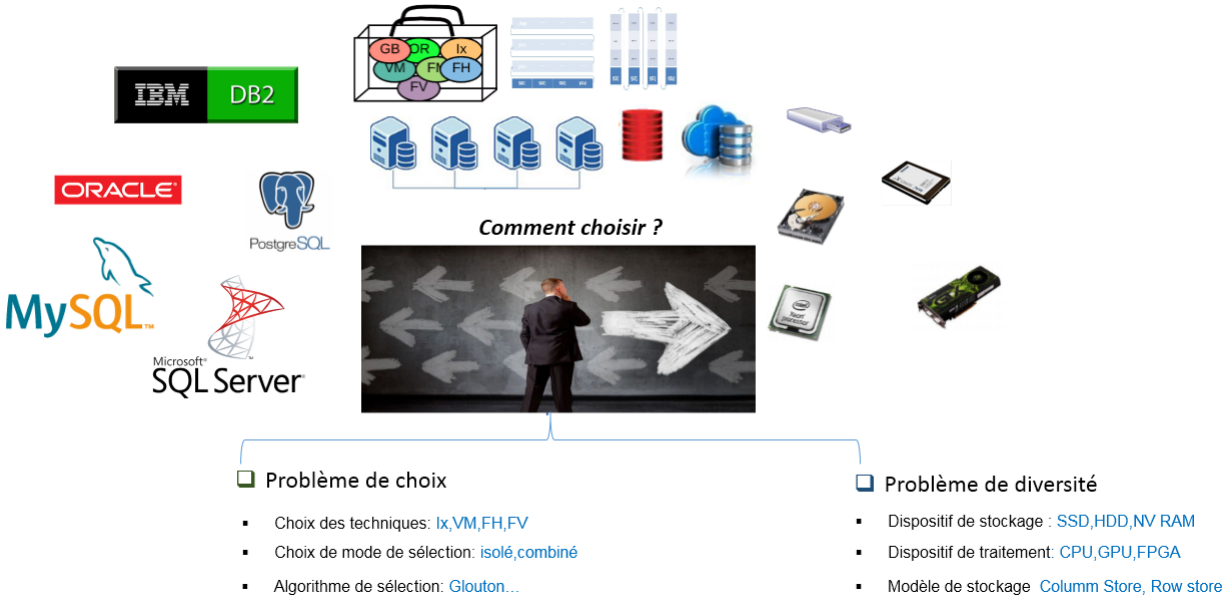


FIGURE 2.15 – Le problème de la conception physique

2.5 Conclusion

Dans les sections précédentes, nous avons présenté et étudié le cycle de vie de la conception des bases de données. Cette étude nous permet de comprendre les différentes caractéristiques de chaque phase et les interdépendances qui peuvent exister entre elles. Notez que le choix approprié de la variante d'une technique de conception donnée dans chaque phase est crucial, et peut avoir un effet sur les autres phases et sur l'énergie globale du système. En résumé, nous concluons que chaque phase du cycle de vie de la conception des bases de données devrait être intégrée à la conception de systèmes de bases de données.

Sommaire

Chapitre

3

Les techniques d'optimisation

Sommaire

3.1 Introduction

La validation et le déploiement des solutions d'optimisation : les recommandations obtenues d'un algorithme de résolution nécessitent une validation par le déploiement sur un environnement réel pour évaluer leur performance effective. Plusieurs outils commerciaux comme Oracle SQL Acces Advisor [1] et DB2 Design Advisor [25] et académiques comme Parinda [17] proposent ces services . Cependant, ces outils présentent des limites liées au choix des structures d'optimisation et au mode de sélection fourni.

3.2 Les techniques d'optimisation

Lors de la conception physique de la base de données, l'administrateur est amené à exécuter une série d'actions (la spécification des éléments de données, les types de données et la sélection des techniques d'optimisation) visant à traiter la complexité des requêtes et à gérer le volume important de données. Au cœur de ces actions, on trouve la sélection des techniques d'optimisation qui constitue le pilier central sur lequel repose la conception physique des bases de donnée.

Plusieurs techniques d'optimisation ont été proposées et classées en deux principales catégories (Voir Figure 1) : techniques redondantes (Fragmentation verticale, Vues matérialisées, Index…) et techniques non redondantes (Fragmentation horizontale, Traitement parallèle…).

3.2.1 Techniques d'optimisation redondantes

Les techniques redondantes sont des structures d'optimisation qui nécessitent un espace de stockage supplémentaire pour leur implémentation. Elles causent ainsi une redondance des données qui sont

présentes à la fois dans les tables et dans les structures d'optimisation [6].

Nous présentons dans ce qui suit trois techniques d'optimisation redondantes étudiées dans le cadre de ce mémoire : les index, les vues matérialisées, la fragmentation verticale.

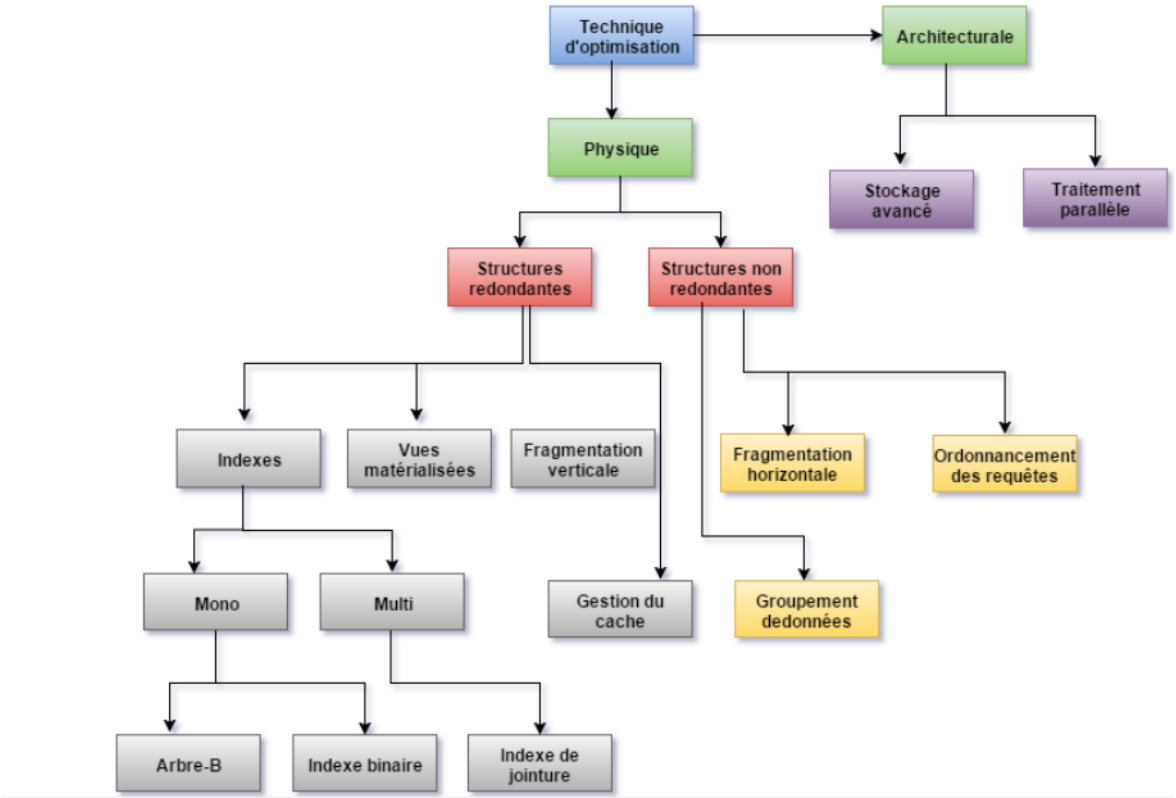


FIGURE 1 – Classification des techniques d'optimisation

3.2.1.1 Les index

Un index est une structure redondante ajoutée à la base ou l'entrepôt de données pour permettre des accès rapides aux données. Il permet à partir d'une clé d'index de trouver l'emplacement physique des tuples recherchés. Deux types d'index sont disponibles : les index mono-table (B-tree, hachage, binaires, projection, etc...) définis sur un ou plusieurs attributs de la même table, et les index multi-tables (index de jointure standards, en étoile et binaires) définis sur plusieurs tables.

3.2.1.1.1 Techniques d'indexation

Nous présentons dans ce qui suit un panorama de techniques d'indexation à savoir les index B-arbre, index de projection, index de hachage, index binaire, et les différents index de jointure.

3.2.1.1.2 Index B-arbre

L'index B-arbre est défini sur un attribut d'une table. Les nœuds de l'arbre renferment les valeurs ordonnées de l'attribut et les feuilles (niveau le plus bas) contiennent les entrées d'index ainsi qu'un pointeur vers l'emplacement physique de l'enregistrement correspondant (Voir Figure 2).

Un B-arbre offre un excellent compromis pour les opérations de recherche par clé et par intervalle, ainsi que pour les mises à jour ce qui explique son utilisation dans la plupart des SGBD [2].

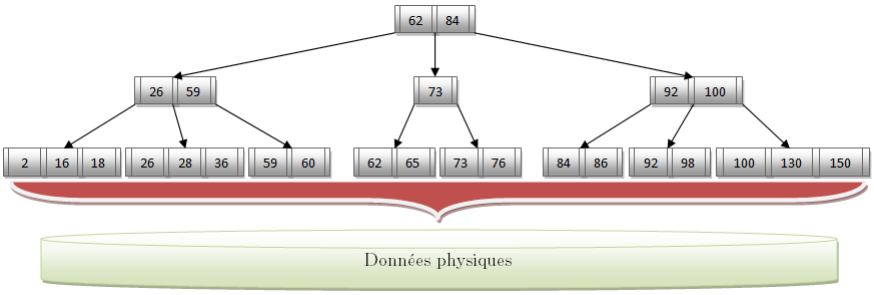


FIGURE 2 – Index B-Arbre

3.2.1.1.3 Index de projection

Un index de projection est défini sur un ou plusieurs attributs d'une table. Il consiste à stocker toutes les valeurs de ces attributs dans l'ordre de leur apparition dans la table (Voir Figure 3). Généralement, les requêtes accèdent à un sous-ensemble d'attributs d'une table. Si ces attributs sont contenus dans un index de projection, l'optimiseur ne charge que cet index pour répondre à la requête . Ce type d'index est très bénéfique pour l'optimisation des requêtes contenant la clause GROUP BY [4].

Table Client				
CID	Nom	Age	Sexe	Ville
616	Mohamed	15	M	Alger
515	Sabrina	25	F	Setif
414	Samy	33	M	Oran
313	Mebarek	50	M	Oran
212	Samia	40	F	Alger
111	Moustapha	20	M	Alger

Index de projection sur l'attribut «VILLE»
Ville
Alger
Setif
Oran
Oran
Alger
Alger

FIGURE 3 – Index de projection

3.2.1.1.4 Index de hachage

L'index de hachage repose sur l'utilisation d'une fonction de hachage. Cette fonction permet, à partir d'une valeur de clé c, de donner l'adresse f(c) d'un espace de stockage où l'élément doit être placé (Voir Figure 4). Dans ce type d'index, le choix de la fonction de hachage est très important pour garantir une bonne performance de l'index [9].

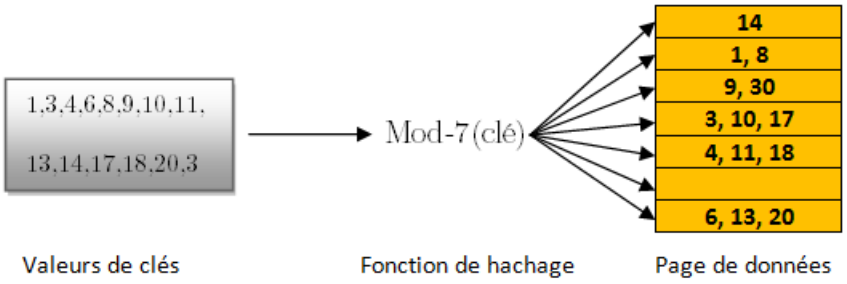


FIGURE 4 – Index de hachage

3.2.1.1.5 Index binaire (Index bitmap)

Le principe d'utilisation de l'index binaire est de définir un ensemble de vecteurs binaires dit Bitmap (contenant des valeurs 0 ou 1) pour référencer l'ensemble des n-uplets d'une table selon les valeurs de l'attribut indexé. Chaque vecteur contient autant de bits qu'il y a de n-uplets dans la table indexée (Voir Figure 5).

La construction de l'index binaire IB défini sur un attribut 'A' ayant n valeurs distinctes et appartenant à une table T composée de m instances, se fait de la manière suivante :

- Créer n vecteurs composés chacun de m entrées ;
- Pour chaque n-uplet i dans T, si i.A = vk, alors mettre 1 dans le i'éme bit du vecteur correspondant à vk, sinon, le bit est mis à 0.

Ce type d'index est très efficace pour les requêtes de type count(*) où seule la lecture de l'index suffit pour répondre à ces requêtes. L'index binaire a été considéré comme le résultat le plus important obtenu dans le cadre de l'optimisation de la couche physique des base de données [9] [2].

Table Client

CID	Nom	Age	Sexe	Ville
616	Mohamed	15	M	Alger
515	Sabrina	25	F	Setif
414	Samy	33	M	Oran
313	Mebarek	50	M	Oran
212	Samia	40	F	Alger
111	Moustapha	20	M	Alger

Index binaire sur l'attribut «VILLE»

Alger	Setif	Oran
1	0	0
0	1	0
0	0	1
0	0	1
1	0	0
1	0	0

FIGURE 5 – Index binaire

3.2.1.1.6 Index de jointure

Etant toujours présente dans les requêtes OLAP et très coûteuse en temps d'exécution, la jointure entre deux tables peut être précalculée et stockée dans un index proposé par Valduriez [24] qui matérialise les liens existant entre deux tables en utilisant une table à deux colonnes chacune représentant l'identifiant d'une table (Voir Figure 6).

Notons que la taille de l'index de jointure dépend de la sélectivité de la jointure. Si la jointure est

très sélective alors la taille de l'index est très petite [9].

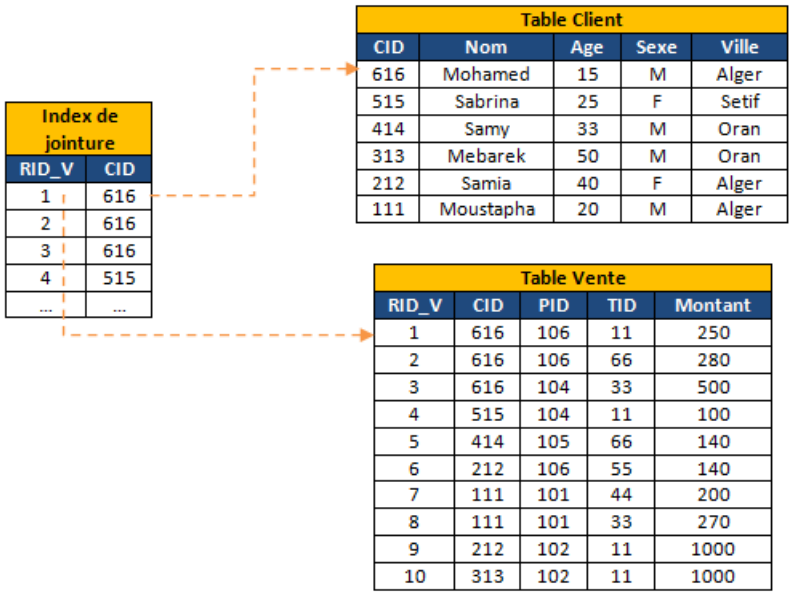


FIGURE 6 – Index de jointure

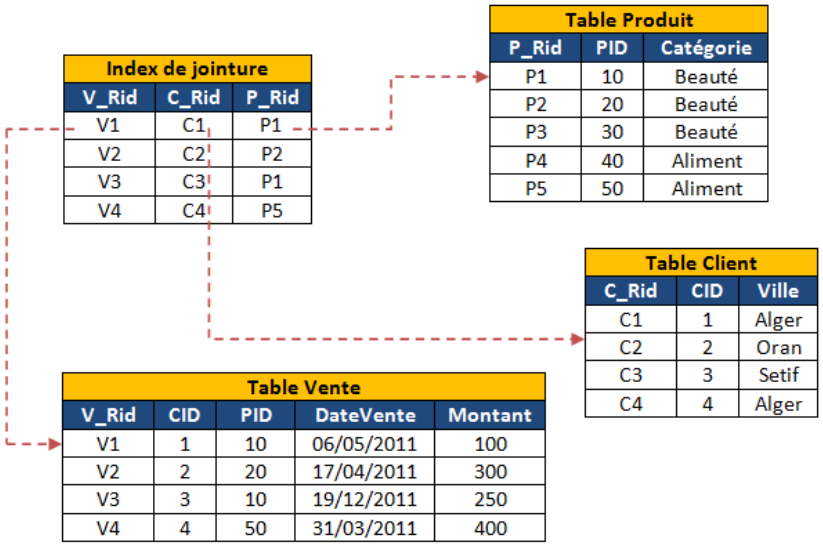
3.2.1.1.7 Index de jointure en étoile

L'index de jointure en étoile, proposé par Red Brick [11] est adapté aux requêtes définies sur les entrepôts de données modélisés en étoile. Il permet de stocker le résultat de la jointure entre la table des faits et les tables de dimensions (Voir Figure 7). Il est dit complet s'il regroupe toutes les tables de dimension, partiel sinon. Il accélère l'opération de jointure mais exige beaucoup d'espace de stockage et un coût de maintenance très élevé [2].

3.2.1.1.8 Index de jointure binaire (bitmap join index)

Comme son nom l'indique, l'index de jointure binaire (IJB) constitue une combinaison entre l'index de jointure et l'index binaire. Il a été proposé pour précalculer les jointures entre une ou plusieurs tables de dimension et la table des faits dans les entrepôts de données modélisés par un schéma en étoile [9]. L'IJB est défini sur un ou plusieurs attributs appartenant à plusieurs tables, plus précisément, il est défini sur la table des faits en utilisant des attributs appartenant à une ou plusieurs tables de dimension (Voir Figure 8).

Supposons, un attribut 'A' ayant n valeurs distinctes, et appartenant à une table de dimension D, et une table des faits F composée de m instances, la construction d'un index de jointure binaire IJB défini sur l'attribut A se fait de la manière suivante : • Créer n vecteurs composés chacun de m entrées ; • Le i'ème bit du vecteur correspondant à une valeur vk est mis à 1 si le n-uplet de rang i de la table des faits est joint avec un n-uplet de la table de dimension D tel que la valeur de A de ce n-uplet est égale à vk. Il est mis à 0 dans le cas contraire.



Index de jointure

V_Rid	C_Rid	P_Rid
V1	C1	P1
V2	C2	P2
V3	C3	P1
V4	C4	P5

Table Produit

P_Rid	PID	Catégorie
P1	10	Beauté
P2	20	Beauté
P3	30	Beauté
P4	40	Aliment
P5	50	Aliment

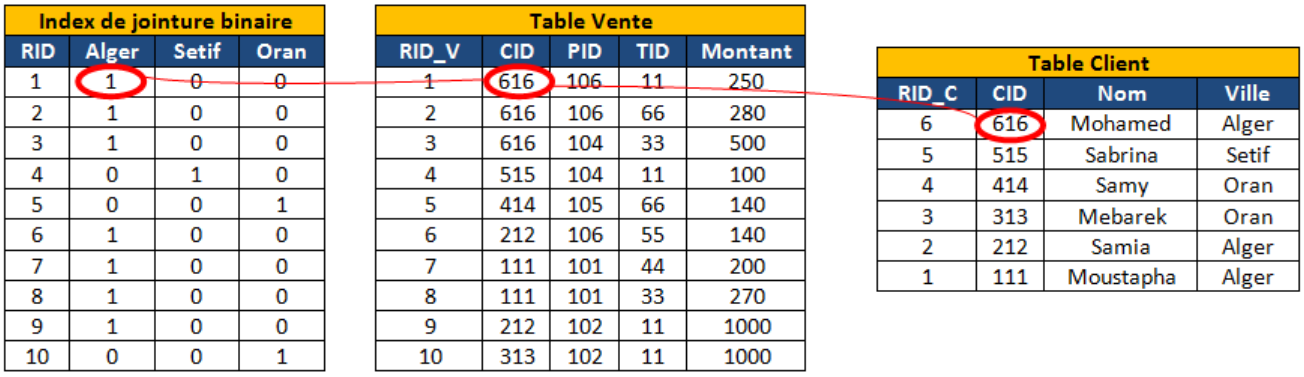
Table Client

C_Rid	CID	Ville
C1	1	Alger
C2	2	Oran
C3	3	Setif
C4	4	Alger

Table Vente

V_Rid	CID	PID	DateVente	Montant
V1	1	10	06/05/2011	100
V2	2	20	17/04/2011	300
V3	3	10	19/12/2011	250
V4	4	50	31/03/2011	400

FIGURE 7 – Index de jointure en étoile



Index de jointure binaire

RID	Alger	Setif	Oran
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	0	0	1
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0
10	0	0	1

Table Vente

RID_V	CID	PID	TID	Montant
1	616	106	11	250
2	616	106	66	280
3	616	104	33	500
4	515	104	11	100
5	414	105	66	140
6	212	106	55	140
7	111	101	44	200
8	111	101	33	270
9	212	102	11	1000
10	313	102	11	1000

Table Client

RID_C	CID	Nom	Ville
6	616	Mohamed	Alger
5	515	Sabrina	Setif
4	414	Samy	Oran
3	313	Mebarek	Oran
2	212	Samia	Alger
1	111	Moustapha	Alger

FIGURE 8 – Exemple d'index de jointure binaire IJB sur l'attribut « Ville »

Comme les index binaires classiques, les IJB sont très bénéfiques pour les requêtes de type Count(*) où la réponse à ces requêtes ne nécessite que l'accès à l'index binaire. Aucun accès aux tables n'est effectué, il suffit de calculer le nombre de 1 dans le vecteur résultat des opérations AND. L'implémentation des index de jointure binaire se fait par l'exécution de la requête de création (ORACLE) pour chaque attribut de dimension sur la table : CREATE BITMAP INDEX IJB ON Fait (Dimension.Attribut) FROM Fait, Dimension WHERE Fait.Id$_{D}imension = Dimension.Id$;

3.2.1.1.9 Index de jointure de dimension

L'index de jointure de dimensions (dimension join index) est un index bitmap proposé pour les entrepôts de données modélisés par un schéma en flocon de neige, il permet ainsi de calculer les jointures existantes entre les hiérarchies de dimension. Il repose sur le même principe que l'IJB. La seule différence

est que le remplissage des vecteurs de cet index se fait en calculant toutes les jointures nécessaires pour joindre les tables de dimension indexées quel que soit leur niveau de hiérarchie avec la table des faits [2].

3.2.1.2 Les vues matérialisées

Une vue est une requête nommée. Elle est dite matérialisée si son résultat est stocké physiquement. Les vues améliorent l'exécution des requêtes, en précalculant les opérations les plus coûteuses comme la jointure et l'agrégation et en stockant leurs résultats dans la base. En conséquence, certaines requêtes nécessitent seulement l'accès aux vues matérialisées et sont ainsi exécutées plus rapidement [9].

Les vues matérialisées peuvent être utilisées pour satisfaire plusieurs objectifs, comme l'amélioration de la performance des requêtes ou la fourniture des données dupliquées. Le concept a été largement utilisé dans l'informatique distribuée. Elles sont utilisées pour dupliquer des données au niveau des sites distribués. Cependant, la mise à jour des données implique systématiquement celle des vues matérialisées calculées à partir de ces données afin de conserver la cohérence et l'intégrité des données. Cela induit une surcharge du système liée au coût de maintenance des vues matérialisées. De plus, la matérialisation des vues requiert un espace de stockage additionnel que l'administrateur alloue à ces vues.

3.2.1.3 La fragmentation verticale

La fragmentation verticale permet de diviser une relation (table) vertialement en plusieurs sous relations (partitions), chacune va comporter un sous ensemble d'attributs de la relation initiale. Pour chaque fragment vertical ou partition, l'attribut clé est ajouté afin de pouvoir reconstituer la relation initiale par opération de jointure (Voir Figure 9). Le résultat du processus de fragmentation verticale est appelé schéma de fragmentation. Ce schéma est défini par l'ensemble des fragments ainsi que les attributs composant chaque fragment. Cette technique est redondante car elle duplique la clé primaire de la table fragmentée, plus la duplication d'autres attributs selon le besoin de fragmentation, ce qui nécessite un espace de stockage supplémentaire.

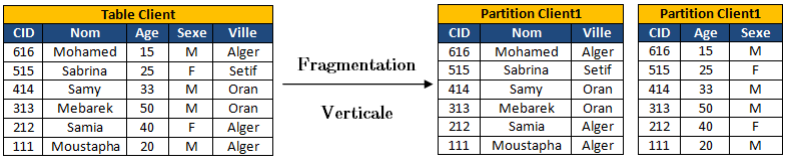


FIGURE 9 – Processus de Fragmentation verticale de la table Client

La fragmentation verticale accélère les opérations de projection. En effet, le traitement des requêtes de projection nécessite uniquement le chargement du fragment vertical dont les attributs figurent dans la requête, il n'est pas nécessaire de charger toute la table. Par contre, cette technique est non efficace pour les requêtes référençant la totalité des attributs de la table. Dans ce cas, elle requiert des jointures supplémentaires très couteuses qui accèdent à plusieurs fragments verticaux [4].

3.2.2 Techniques d'optimisation non redondantes

Contrairement aux techniques d'optimisation redondantes, les techniques non redondantes ne dupliquent pas les données ce qui ne nécessite pas un espace de stockage supplémentaire. Nous présentons dans ce qui suit deux techniques d'optimisation non redondantes à savoir la fragmentation horizontale et le traitement parallèle.

3.2.2.1 La fragmentation horizontale

La fragmentation horizontale (FH) se base sur le principe de réorganisation des données par la répartition des tuples d'une table en plusieurs sous ensembles disjoints (Voir Figure 10) appelés fragments horizontaux et pouvant être accédés séparément. Cette répartition est effectuée par une opération de restriction sur la table initiale (Ex : Client1=ville=Alger(client)) et une simple opération d'union permet de reconstituer la relation initiale [9]. Un schéma de fragmentation horizontale d'une table est le résultat du processus de fragmentation de cette table. Il contient un ensemble de fragments horizontaux, où chacun est défini par une clause de prédicats simples. Cette technique est considérée comme une technique d'optimisation non redondante du fait qu'elle ne réplique pas de données.

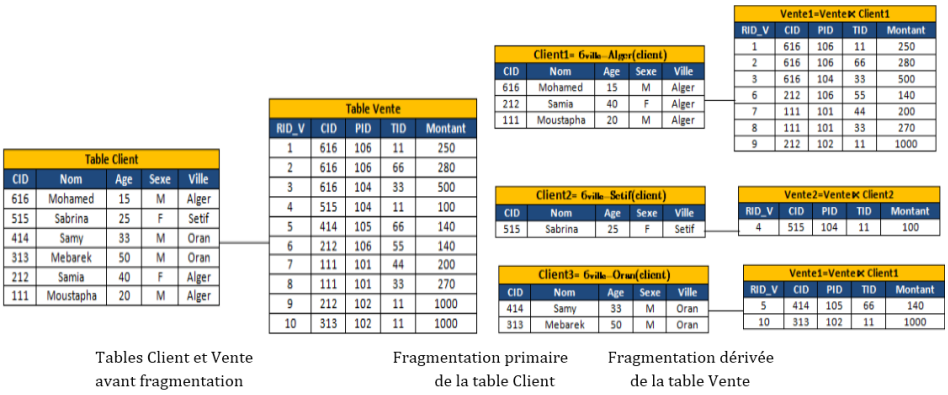


FIGURE 10 – Processus de Fragmentation horizontale de la table Client et Vente

La fragmentation horizontale se décline sous deux types : la FH primaire et la FH dérivée [4]. La fragmentation horizontale primaire d'une table se base sur les prédicats de sélection définis sur cette table, elle favorise le traitement des requêtes de restriction portant sur les attributs utilisés pour fragmenter une table ; or que la fragmentation horizontale dérivée exploite le lien existant entre deux tables pour fragmenter l'une d'entre elles en fonction des fragments de l'autre, et est utile pour le traitement des requêtes de jointure. Concrètement, la fragmentation dérivée d'une table S n'est possible que lorsqu'elle est liée avec une table T par sa clé étrangère. Une fois la table T fragmentée par la fragmentation primaire, les fragments de S sont générés par une opération de semi-jointure entre S et chaque fragment de la table T. Les deux tables seront équi-partitionnées grâce au lien père-fils.

3.2.2.1.1 Les avantages de la fragmentation horizontale

Nous citons quelques avantages da la fragmentation horizontale : • Flexibilité et Manageabilité : La fragmentation d'une table permet de donner une meilleure flexibilité pour gérer et manipuler ses partitions. Pour l'utilisateur aucune modification n'est nécessaire au niveau du code SQL car la réécriture de requêtes SQL se fait automatiquement par le système de gestion de bases de données. • Performance : La FH permet d'éviter le chargement des données non pertinentes pour répondre à une requête, ce qui augmente la performance. Elle optimise les sélections avec son mode primaire, et améliore les jointures avec le mode dérivée. • Disponibilité : Lors des opérations de maintenance de la base ou l'entrepôt de données, les partitions non concernées par ces tâches seront toujours disponible ce qui réduit l'arrêt du système. Donc la FH permet de cibler la tâche de maintenance.

3.2.2.1.2 Les modes de la fragmentation horizontale

Comme cité ci-dessus, la fragmentation horizontale s'appose sous deux catégories : • La fragmentation horizontale primaire qui comprend deux modes : le mode simple (à un seul niveau) et le mode composé (modes à deux niveaux) [9]. Un mode de fragmentation simple permet de partitionner une table selon les valeurs d'un seul attribut en un ensemble de partitions. Le mode composé permet de fragmenter une table en un ensemble de partitions en utilisant plusieurs attributs, • et la fragmentation horizontale dérivée avec son mode Référence. Nous détaillons dans ce qui suit les différents modes de fragmentation.

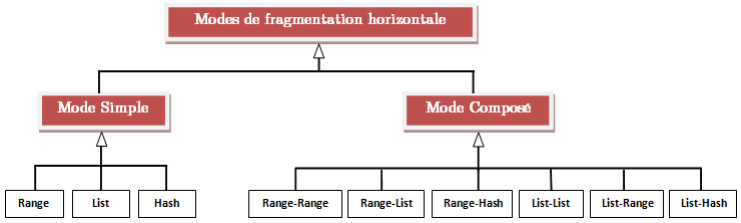


FIGURE 11 – Classification des modes de fragmentation primaire

3.2.2.1.3 Mode Simple

Un mode de fragmentation simple d'une table T est défini par : MS < C, V > où C désigne la clé de fragmentation et V un ensemble de valeurs de cette clé V = V1, V2,..., Vn. Chaque élément dans l'ensemble V est constitué d'une ou plusieurs valeurs du domaine de valeurs de la clé C. L'ensemble V peut être vu alors comme un découpage du domaine de valeurs de C en plusieurs sous-domaines sd1, sd2,..., sdn, où chaque sdi correspond à une valeur Vi. Chaque sous-domaine sdi permet de générer une partition Ti de la table T. Lorsqu'une instance I est insérée dans la table T, le système recherche le sous-domaine sdi contenant la valeur de C de cette instance et attribue cette dernière à la partition Ti définie par sdi [3]. Trois types de fragmentation peuvent réaliser le mode simple : Range, List et Hash. La différence entre ces types réside dans la façon dont l'ensemble V est déterminé.

(a) Fragmentation par intervalle (RANGE) : Avec le partitionnement par intervalles, les tuples sont

répartit en fonction de la valeur d'une colonne, par rapport à un ensemble d'intervalles, qui désigne le domaine où appartient chaque partition. Chaque intervalle est représenté par une borne inférieure et une borne supérieure. Exemple : Fragmenter la table Client sur l'attribut Age par le mode Range peut être représentée par : RANGE :<Age, [1 18[, [18 60[, [60 120]>.

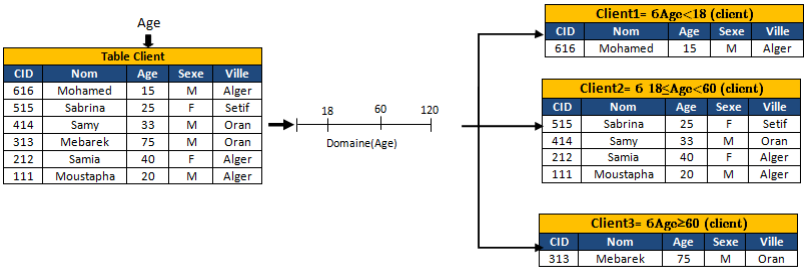


FIGURE 12 – Processus de Fragmentation par RANGE

La fragmentation par RANGE de la table Client selon ce découpage se fait en utilisant la commande SQL suivante :

```
1
2 CREATE TABLE CLIENT (
3 CID number(6), Nom varchar(30), Age number(3),
4 Sexe char(1), Ville varchar(30)
5 PARTITION BY RANGE(Age)
6 PARTITION C-Enfants VALUES LESS THAN (18) TABLESPACE TBS-Enfants,
7 PARTITION C-Adultes VALUES LESS THAN (60) TABLE SPACE TBS-Adultes,
8 PARTITION C-Retraites VALUES LESS THAN (MAXVALUE) TABLE SPACE TBS-Retraites) ;
```

Listing 3.1 – Extrait du SQL pour insérer un nouveau modèle de coût

Chaque partition est caractérisée par un nom (Ex : C-enfants), une borne supérieure (Ex : 18) et un TableSpace où elle sera stockée (Ex : TBS-Enfants). La fragmentation par Range est utilisée généralement pour l'historisation des données en utilisant comme clé de fragmentation un attribut de type Date. Le mode Range peut être bénéfique pour les requêtes d'intervalles utilisant des prédicats coïncidant avec les intervalles définissant les partitions.

(b) La fragmentation par liste (LIST) : Dans ce type de partitionnement, les partitions sont définies par des listes de valeur. Exemple : Fragmenter la table Client par le mode List en trois partitions selon l'attribut Ville : La partition1 correspond aux clients habitant Alger, la deuxième ceux habitant Oran ou Setif et la troisième aux autres clients (Voir Figure 2.18).

La fragmentation par LIST de la table Client selon ce découpage se fait en utilisant la commande SQL suivante :

```
1
2 CREATE TABLE CLIENT (
3 CID number(6), Nom varchar(30), Age number(3) ,
4 Sexe char(1), Ville varchar(30))
5       PARTITION BY LIST (Ville)
6             (PARTITION C- Alger VALUES (    Alger    ),
7               PARTITION C-Oran_Setif VALUES (    Oran    ,    Setif    ),
8               PARTITION C-Autres VALUES (DEFAULT)) ;
```

Listing 3.2 – Extrait du SQL pour insérer un nouveau modèle de coût

(c) La fragmentation par hachage (HASH) : Le mode de fragmentation par hachage utilise une fonction de hachage fournie par le système pour fragmenter une table. Pour utiliser ce mode, il suffit de

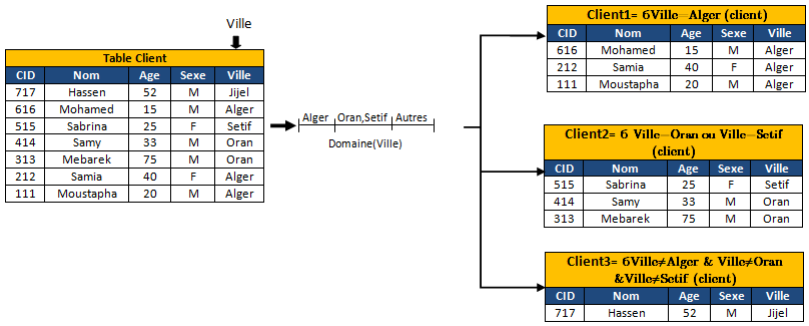


FIGURE 13 – Processus de Fragmentation par List

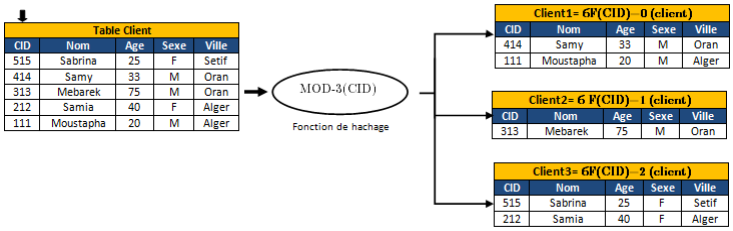


FIGURE 14 – Processus de Fragmentation par Hash

spécifier au système la clé de fragmentation ainsi que le nombre de partitions voulues (les éléments de l'ensemble V sont générés par la fonction de hachage). Ce mode est donc défini par < C, n > où C est la clé de fragmentation et n est le nombre de partitions voulues. Exemple : Fragmenter la table Client sur l'attribut CID par le mode Hash en trois partitions (Voir Figure 14).

La fragmentation par HASH de la table Client selon ce découpage se fait en utilisant la commande SQL suivante :

```
10
11 language=sql]
12 CREATE TABLE CLIENT (
13 CID number(6), Nom varchar(30), Age number(3),
14 Sexe char(1), Ville varchar(30))
15         PARTITION BY HASH (CID)
16         PARTITION 3 STORE IN (TBS1, TBS2, TBS4);
```

Listing 3.3 – Extrait du SQL pour insérer un nouveau modèle de coût

3.2.2.1.4 Mode Composé

Le mode composé permet de combiner deux modes simples de fragmentation MS1 et MS2 (Voir Figure 15). Le premier mode simple MS1 est utilisé pour fragmenter la table en un ensemble de partitions (premier niveau de fragmentation). Le deuxième mode simple MS2 est utilisé pour fragmenter chaque partition obtenue par MS1 en un ensemble de sous-partions (deuxième niveau de fragmentation). Notons que la clé de fragmentation utilisée dans MS1 doit être différente à celle utilisée dans MS2. En combinant tous les modes simples, plusieurs modes composites peuvent être définis : Range-Range, Range-List, Range-Hash, Hash-Hash, Hash-List, etc...

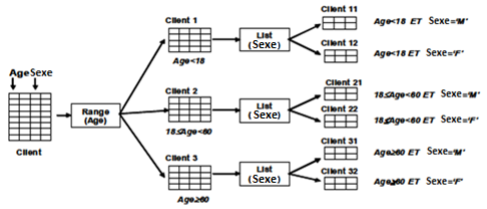


FIGURE 15 – Exemple du mode composé RANGE-LIST

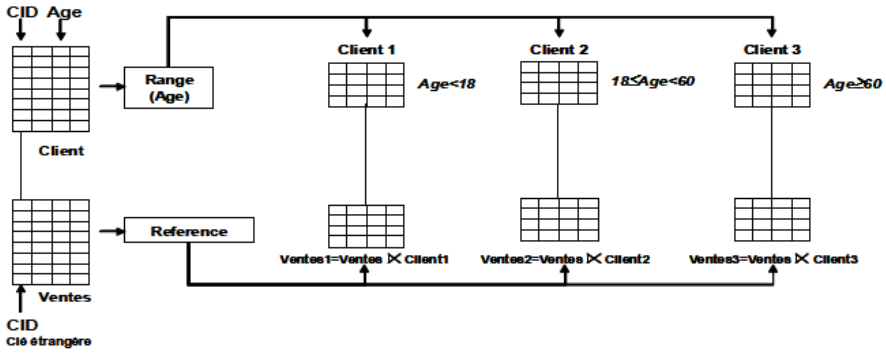


FIGURE 16 – Mode de fragmentation par référence

La fragmentation composite pour l'exemple de la Figure 15 peut être matérialisée par la commande SQL suivante :

```
18
19 CREATE TABLE CLIENT (
20 CID number(6), Nom varchar(30), Age number(3),
21 Sexe char(1), Ville varchar(30))
22        PARTITION BY HASH (CID)
23        PARTITION 3 STORE IN (TBS1, TBS2, TBS4);
```

Listing 3.4 – Extrait du SQL pour insérer un nouveau modèle de coût

3.2.2.1.5 Mode Référence

La fragmentation par référence permet de fragmenter une table S selon les fragments d'une autre table T en utilisant la relation père-fils entre les deux tables référencée par la clé étrangère (principe de la fragmentation horizontale dérivée). Exemple : Soit la table Client fragmentée par le mode Range en trois fragments. La fragmentation par référence de la table Ventes permet de créer trois partitions chacune liée à une partition de la table Client (Voir Figure 16).

Le code suivant permet de fragmenter la table Ventes par référence :

```
25 CREATE TABLE VENTES (
26 CID number(6), Date DATE , Montant Number(8,2)
27          CONSTRAINT Client_fk FOREIGN KEY (CID) REFERENCES Client(CID))
28          PARTITION BY REFERENCE(Client_fk);
```

Listing 3.5 – Extrait du SQL pour insérer un nouveau modèle de coût

3.2.2.1.6 Règles de correction de la fragmentation horizontale

Pour considérer qu'un processus de fragmentation horizontale d'une table T en N fragments T1,T2,•••,Tn soit valide, il doit vérifier trois règles de correction : la complétude, la disjonction et la reconstruction. (a) Complétude : Toute instance de T doit être présente dans au moins un fragment. La complétude garantit qu'aucune instance ne sera perdue après la fragmentation. (b) Disjonction : Toute instance appartenant à un fragment Ti ne doit pas appartenir à un autre fragment. (c) Reconstruction : Une fois la fragmentation effectuée, il doit être toujours possible de reconstruire la table d'origine à partir de ses fragments.

3.2.2.2 Le traitement parallèle

Le traitement parallèle des requêtes consiste à répartir leur déroulement dans l'espace (nœuds ou processeurs d'exécution) et dans le temps (ordonnancement) par une utilisation judicieuse des ressources disponible [18]. On cherche ainsi à réduire le temps de réponse moyen (satisfaction des utilisateurs) et le temps de réponse de chaque requête prise indépendamment (satisfaction de chaque utilisateur en particulier). Afin de permettre l'exploitation optimale des ressources, l'utilisation de système de gestion de bases de données (SGBD) parallèle s'avère primordiale. Ce type de SGBD fonctionne sur plusieurs processeurs et disques, reliés par des réseaux à hauts débits, conçu pour l'exécution parallèle d'opérations telles que les opérations d'algèbre relationnelle, et doit assurer des fonctions telles que les fonctions d'agrégation, les fonctions de gestion et contrôle de données et de transactions réparties… Il existe deux principaux modèles d'exécution : Le modèle d'exécution fragmenté, et le modèle d'exécution non fragmenté.

Dans le premier modèle d'exécution, l'ensemble des tables sont fragmentées, ainsi chaque processeur n'accède qu'à un sous ensemble de données. Cette approche est la seule possible sur une architecture « Shared-Nothing » (Sans partage), sans quoi les accès aux données, souvent distants, entrainerait des surcoûts de communication prohibitifs [18].

Le modèle d'exécution non fragmenté est uniquement applicable sur des architectures à mémoire partagée . Chaque processeur n'accède qu'à un sous ensemble de tables. Ce modèle est considéré comme une simple parallélisation d'une exécution centralisée (en mono-processeur).

3.2.2.2.1 Formes de parallélisme

Il existe différentes formes de parallélisme : le parallélisme inter-requêtes, et le parallélisme intra-requêtes [7]. Une présentation des différentes formes de parallélisme fera l'objet de la suite de cette section.

3.2.2.2.2 Parallélisme inter-requêtes

Ce type de parallélisme vise à partager les ressources de façon équitable et efficace entre les requêtes soumissent par les utilisateurs (Voir Figure 19). En d'autres termes, il permet d'exécuter plusieurs requêtes en parallèle.

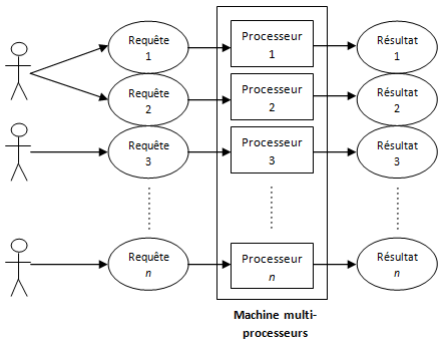


FIGURE 17 – Parallélisme inter-requêtes

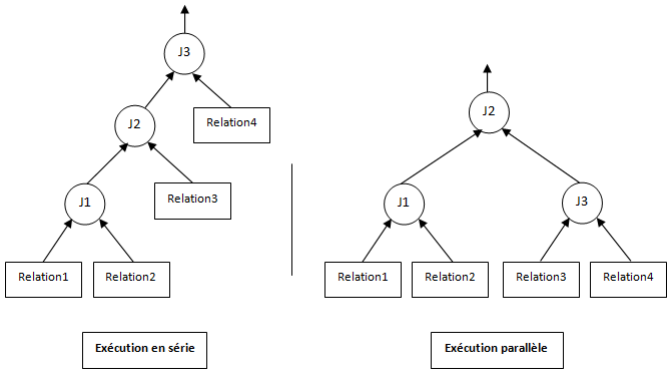


FIGURE 18 – Parallélisme intra-requêtes

3.2.2.2.3 Parallélisme intra-requête

Une requête relationnelle, sous sa forme algébrique, est formée par un graphe d'opérateurs s'appliquant sur de grands ensembles de données ou relations. Ce type de parallélisme permet de fragmenter le code séquentiel d'une requête selon ses opérateurs pour constituer plusieurs fragments ou tâches pouvant s'exécuter simultanément (en parallèle) et pouvant être combinés ou non pour améliorer le temps de réponse de la requête, puis de rassembler les sous-résultats obtenus. Le gain en performance dans ce type de parallélisme est facteur d'autres paramètres notamment la taille des relations et leur distribution [18].

Dans l'exemple de la (Figure 18) les trois jointures J1, J2, J3 au lieu d'être exécutées séquentiellement (arbre de gauche), il est possible d'effectuer parallèlement les deux jointures J1 et J3, puis d'effectuer ensuite la jointure J2 (arbre de droite). En exécutant en parallèle plusieurs opérateurs du graphe de la requête, il est possible de générer du parallélisme de type inter-opérateur. En fragmentant les opérandes d'un seul opérateur, il est possible de le décomposer en sous-opérateurs identiques conduisant à générer du parallélisme de type intra-opérateur [8].

(a) Parallélisme inter-opérateur Le parallélisme inter-opérateur consiste à exécuter en parallèle des opérateurs d'une même requête sur différents processeurs (Voir Figure 19), ce qui permet de faire travailler plusieurs nœuds (processeurs), de limiter le temps d'horloge, et de limiter le volume de données transférées.

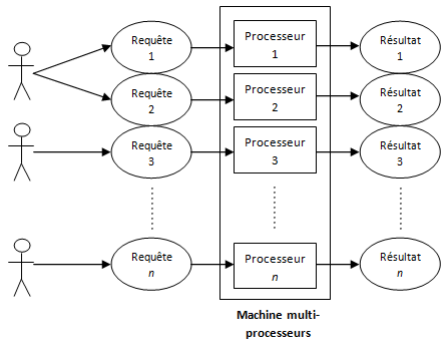


FIGURE 19 – Parallélisme inter-opérateurs

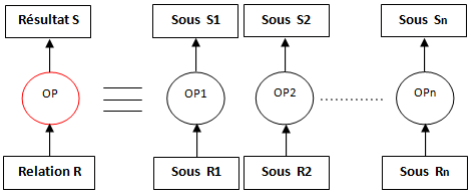


FIGURE 20 – Parallélisme intra-opérateurs

(b) Parallélisme intra-opérateur Dans ce type de parallélisme, chaque opérateur est décomposé en un ensemble de sous-opérateurs, chacun s'exécutant sur une partition de la relation (Table). Cette décomposition est rendue possible grâce à la fragmentation des relations de la base ou l'entrepôt de données [8]. Une même opération peut alors être traitée par plusieurs processeurs [18] (Voir Figure 20).

3.3 Conclusion

Nous avons présenté dans ce chapitre, la classifiant et les différentes techniques d'optimisations des requêtes définies dans le contexte de la base de données,à savoir : les index, les vues matérialisées, et la fragmentation verticale comme techniques redondantes, ainsi que la fragmentation horizontale primaire et dérivée, et le traitement parallèle comme techniques non redondantes.

Troisième partie

Notre démarche adoptée

Chapitre

4

Présentation de notre approche

Sommaire

4.1 Introduction

Comme nous avons vu dans les deux chapitre précedens, le principal objectif lors du traitement de requête par un SGBD est la minimisation du temps de réponse des requêtes, à l'aide de techniques sophistiquées, qui comprennent des algorithmes pour sélectionner et exécuter un plan optimal pour une requête.

Dans cette section, nous présentons notre méthodologie pour créer un *advisors* qui recommande la configuration physique d'une requête SQL. Nous adoptons une approche basée sur les modèles pour trouver les paramètres clés ayant un impact sur la conception physique de base de données.

4.2 Processus conceptuel de la conceptions physique des BDs

Une conception physique dans une base de données est un ensemble de structures de stockage, qui sont utilisés pour accélérer les requêtes futures à mesure qu'ils arrivent. Le type de structures de stockage utilisées dépend souvent de l'architecture de base de données spécifique. La plupart des bases de données utilisent à la fois des vues matérialisées et des indexes dans leur conception physique.

La (Figure 4.1) donne un aperçu général du processus de conception physique. Il montre trois phases principales. La première phase est consacrée aux DBAs, où les caractéristiques du système et les besoins de DBA sont définis. Notez que les caractéristiques du système peuvent également être liées aux phases de conception précédentes du cycle de vie (c'est-à-dire des phases conceptuelles et/ou logiques de la base de données). Une fois que le DBA spécifie ses besoins, celui-ci peut être évalué par un ou plusieurs advisor. De toute évidence, les outils utilisés doivent supporter comme entrée les spécificités du système sous la conception. Bien que certains outils d'aide en des boîtes blanches fournissent les algorithmes et les modèles de coûts qui sont derrière leurs recommandations, l'étape consistant à choisir des Advisors adaptés n'est pas automatique et peut être laborieuse et susceptible d'erreurs, en raison de la variété des paramètres qui doivent être pris en compte et leur impact sur la recommandation, sa qualité et sa justesse. L'idée sous-jacente de cette orientation est de proposer une approche qui met en majuscule la recherche sur la conception physique et la transmission d'idées, de la conception basée sur le savoir au processus modèle

En raison de la complexité de la conception physique, plusieurs efforts de l'industrie et du milieu universitaire ont été menés et des outils proposés, également appelés *advisors* pour aider le DBA pendant leurs tâches dans le choix de sélection des $\mathcal{SP}$s.

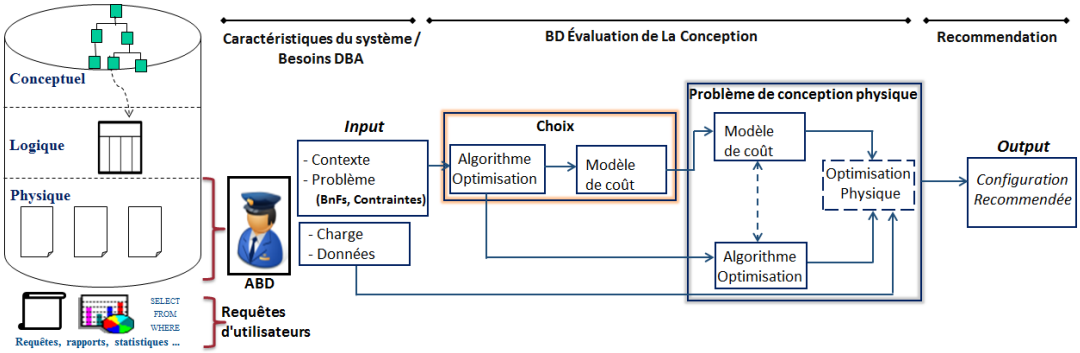


FIGURE 4.1 – Processus conceptuel de la conceptions physique des BDs

4.3 Formalisation de problème de séléction des structures physiques

L'administrateur de base de données ne peut pas matérialiser toutes les structures physiques possibles, car il est limité par certaines ressources comme, l'espace disque et le temps de calcul, ceci est connu comme le problème de sélection des structures physique PCP. Le PCP consiste à trouver un ensemble approprié de structures satisfaisant un ensemble donnée de BnF tels que la *performance* des requêtes, la *fiabilité*, *l'utilisabilité*,...etc. Les structures sélectionnées doivent répondre à un ensemble de contraintes telles que le coût de stockage, les coûts de maintenance, etc.

Le PCP est connu pour être un problème *NP-complet* [10] en raison du fait que l'espace de solution s'accroît de façon exponentielle lorsque la taille du problème augmente. La formalisation générale de PCP est défini comme suit : Étant donné :

- **Entrées** :
  - un schéma d'une base de données : BD ; une charge de requêtes : W=$\{Q_1,Q_2,..,Q_n\}$;

- un ensemble de contraintes : C={C_1,C_2,..,C_p}
- un ensemble de besoins non-fonctionnelles : BNF={bnf_1,bnf_2,...,bnf_k};

— **Sorties** :

- sélectionner un ensemble de Index : SO ={so_1,$so2$,..;so_p} satisfaisant des BnF et en respectant l'ensemble des contraintes C.
- estimation de BnF (exp. E/S) quand on exécutes les requêtes.

Plusieurs instanciations de cette formalisation existent : (i) le BNF , (ii) les contraintes C, et (iii) les algorithmes utilisés pour résoudre le PCP.

FIGURE 4.2 – Classification des travaux des advisors pour les Index (Ix)

Ces dimensions permettent de classer les solutions existant dans la synthése bibliographique. La (Figure 4.3) illustre notre cube.

4.4 Vue d'ensemble de notre approche

Les principales étapes pour développer notre *advisor* sont : (i) l'analyse de Domaine, (ii) la modélisation de domaine, (iii) la mise en œuvre du domaine comme illustre dans la figure suivant : (Figure 4.4)

- **Analyse de domaine** : Pour identifier les principales dimensions d'un advisors.
- **Modélisation de domaine** : Pour lier les dimensions identifiés entre eux.
- **Usage** : Pour Manipuler les entités identifiées on se basant sur une API (Application Programming Interface) .

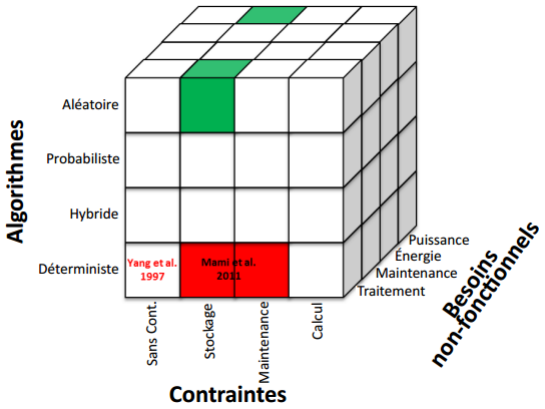


FIGURE 4.3 – Méthodes de résolutions du PCP

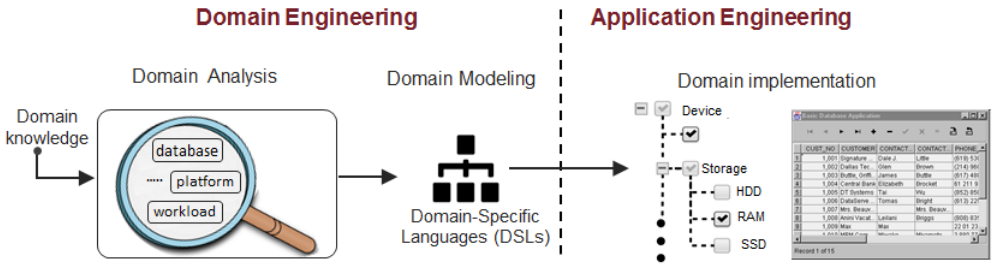


FIGURE 4.4 – notre approche

4.4.1 Les étapes de l'approche

Dans cette section nous allons présenter les étapes de notre approche proposée :

4.4.1.1 L'analyse de Domaine

Pour analyser le domaine nous avons examiné les articles issus des principales conférences de base de donnée (VLDB , SIGMOD, ICDE) à partir de la période de 2009-2016. Nous trouvons environs 8 articles traitons les différents Advisors et les différents composant des Advisors.

Nous comparons les Advisors proposés dans la littérature et nous avons extraire les différents composantes qui représentent les briques de base utilisées pour développer un tel advisor :

- Charge des requêtes
- les besoins non fonctionnelle
- les paramètres matériel (hardware)
- les paramètres logiciels (software)
- les algorithmes de sélection
- les modèles de coût

Nous avons analysé la relation entre ces outils, l'exigence et l'évolution de la technologie de la base de données. Après notre synthèse on a constaté que la majorité des travaux proposent des outils automa-

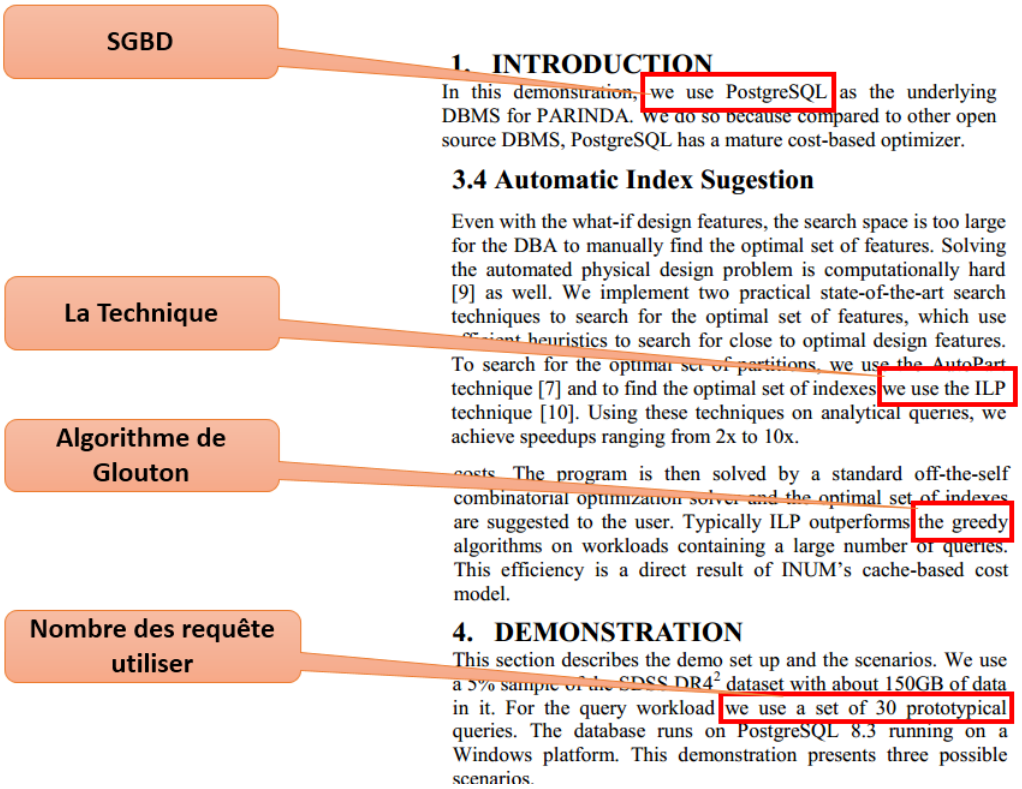


FIGURE 4.5 – Les composants de PARINDA

tiques pour la phase d'administration et de tuning de la conception physique avec l'objectif de réduire le coût d'administration (intervention humaine). Ces outils permettent d'automatiser l'ensemble des techniques d'optimisation. Ils permettent d'analyser complètement la charge et propose des recommandations pour créer des nouveaux structures physique si nécessaire, de supprimer les structures inutilisés comme par exemple la création des vues matérialisées, etc.

Peu de travaux sont orientés vers une optique semi-automatique où l'administrateur (DBA) peut interagir avec l'Advisors et injecter ces préférences souhaitées. La (Figure 4.6) illustre trois mode de sélection : (i) Automatique (ii) semi automatique et (iii) Interactive.

4.4.1.2 La modélisation de Domaine

1. **La méta-modélisation** : La méta-modélisation est le processus de définition du méta modèle d'un langage de modélisation (voir Figure 4.7). Le méta modélisme vise à modéliser la langue, ce qui permet d'exprimer le système conçu. Un langage de modélisation est tout langage artificiel qui peut être utilisé pour exprimer des informations de systèmes dans une structure. Les langages de modélisation peuvent être graphiques ou textuels (des formes nommées qui représentent des concepts, et des liens reliant les formulaires pour exprimer les relations). Les langages de modélisation textuelle utilisent généralement des grammaires normalisées pour faire des expressions interprétables par ordinateur. Exemples d'un langage de modélisation graphique et de son langage de modélisation textuel correspondant sont UML et XMI.

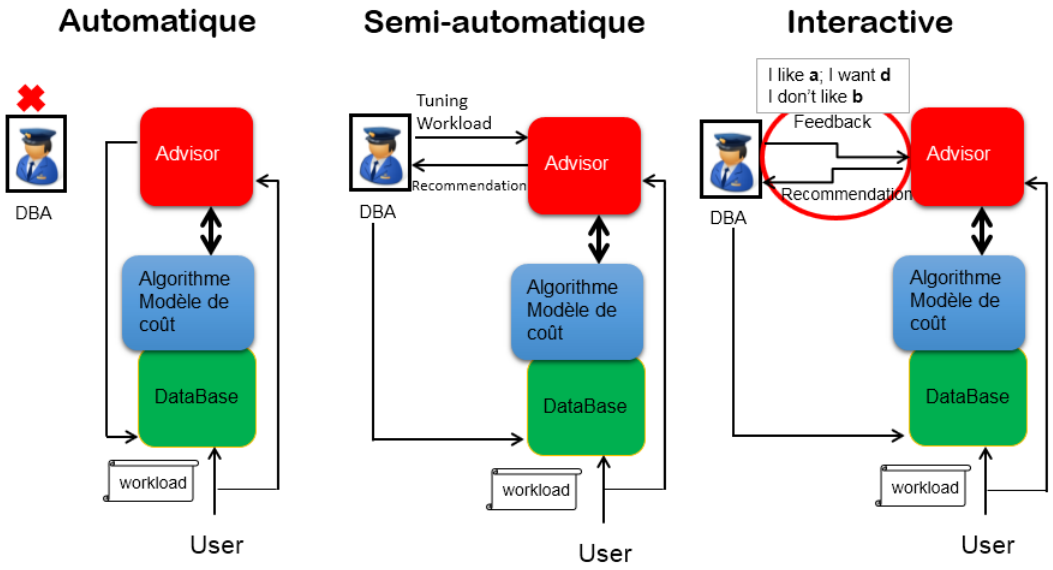


FIGURE 4.6 – Classification des Advisors

La notion de méta modèle est largement utilisée comme langue de description dans différents domaines. Ensuite, de nombreux méta-modèles ont émergé pour fournir leurs caractéristiques dans un domaine particulier (services Web, bases de données, développement de logiciels, etc.). Pour faire face à l'émergence incompatible de méta-modèles, *l'Object Management Group* (OMG) a proposé un cadre générique prenant en charge différents méta modèles. Étant donné que l'objectif est de freiner le nombre de niveaux d'abstraction, la solution était de fournir un langage commun définissant tous les méta-modèles. En conséquence, l'OMG a fourni le MOF en tant que méta-méta-modèle. Cette dernière est la pierre angulaire de MDA. Par conséquent, l'approche MDA représente l'architecture à quatre couches (voir Figure 4.7)

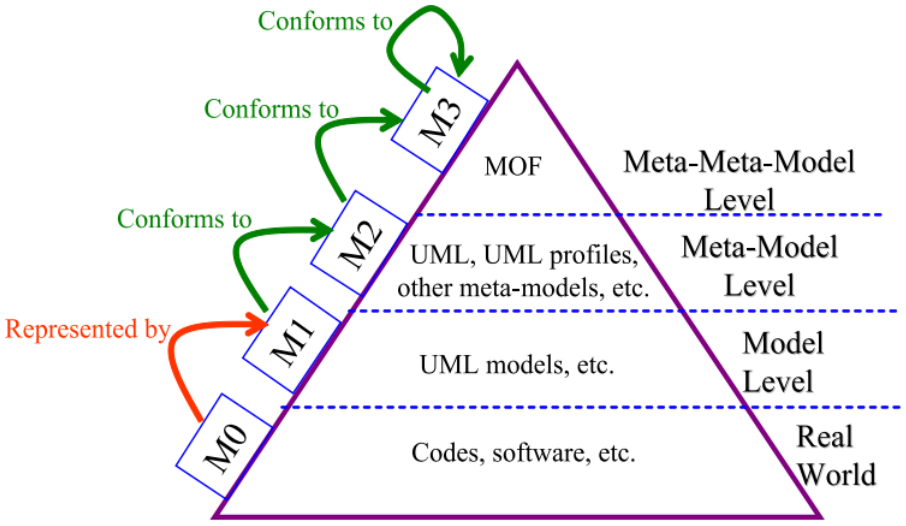


FIGURE 4.7 – L'architecture à quatre couches de MDA

— **Méta-méta-modèle (*M3*) :** un méta-méta-modèle est un modèle qui décrit des langages de méta-modélisation. Autrement dit, le méta-méta-modèle contient les éléments de modélisation nécessaires à la définition des langages de modélisation et il a également la capacité de se décrire lui-même.
— **Méta-modèle (*M2*) :**selon la définition du MOF (Meta Object Facility), un méta-modèle définit la structure que doit avoir tout modèle conforme à ce méta-modèle. On dit qu'un modèle est conforme à un méta-modèle si l'ensemble des éléments qui constituent le modèle est défini par le méta-modèle. Par exemple, le méta-modèle UML définit que les modèles UML contiennent des packages, leurs packages des classes, leurs classes des attributs et des opérations, etc.
— **Modèle (*M1*) :** une abstraction d'un système conçu sous la forme d'un ensemble de faits construits selon une intention particulière. Il doit pouvoir être utilisé pour répondre à des questions sur le système étudié.
— **Monde réel (*M0*) :** contient des objets concrets représentés par des modèles de niveau M1, par exemple, des codes exécutables correspondant aux modèles UML.

4.4.1.3 Les éléments corps

Cette section est consacrée à présenter notre contribution basée sur un modèle. La (Figure 4.8) représente les éléments de base du méta-modèle, dont son élément racine est la classe `Advisor` (c'est-à-dire que l'instanciation commence à partir de cette classe). Chaque instance de l'Advisor est composée d'un contexte (classe `Context` d'instance), d'un algorithme de sélection (classe `Algorithme` d'instance), d'un modèle de coût (classe `CostModel` d'instance) et des hypothèse (classe `hypo`

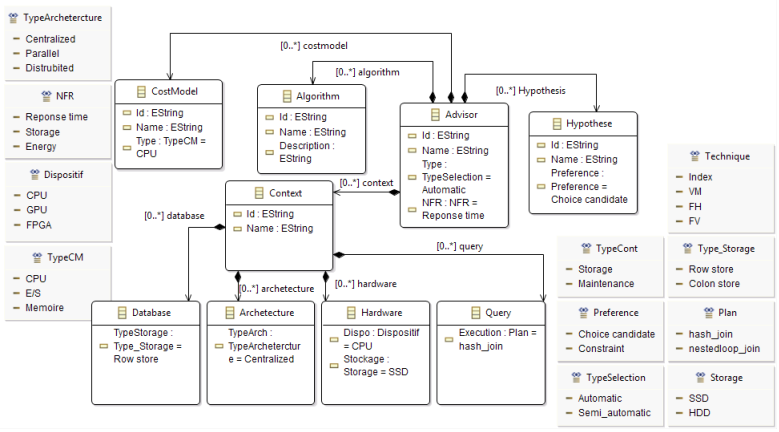


FIGURE 4.8 – Méta-modèle de l'Advisor

Chaque algorithme possède un identifiant, nom (hérité d'algorithme) et chaque algorithme possède un ou plusieurs critères d'arrêt, de l'initialisation et des solutions (voir Figure 4.9). Il existe quatre types spécifiques d'algorithme : algorithmes *déterministes*, algorithmes *randomisés*, algorithmes *hybrides*, algorithmes *aléatoires*, algorithmes *hybrides* et *programmation dynamique*. La (Figure 4.10) représente l'élément classe `CostModel`. Chaque instance `CostModel` est composée d'une métrique (classe `Metric` d'instance), d'un contexte (classe de `contexte` d'instance) et d'une fonction de coût (classe de `CostFunction` d'instance). Il se caractérise également par un nom. Chaque instance de la classe `CostModel` possède également au moins un type de coût. Grâce à

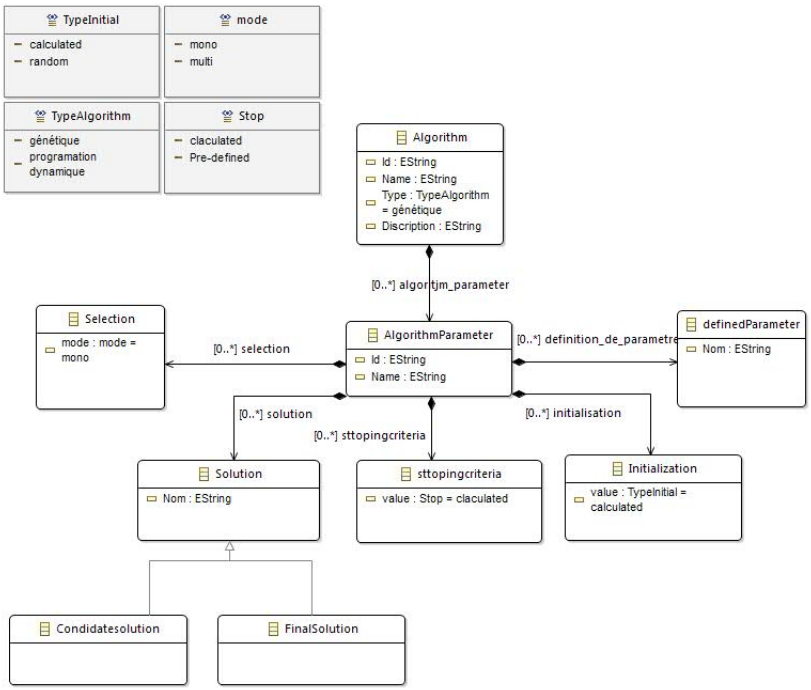


FIGURE 4.9 – Meta-modéle de l'algorithme de sélection

la classe `CostFunction`, la formule mathématique d'un modèle de coût est soutenue de façon structurée. La classe `CostFunction` comprend deux parties d'opérandes : coûts logiques et coûts physiques. En fait, un opérande peut être une valeur réelle donnée (c'est-à-dire une instance de la classe `ConstantValue`) ou dérivé d'autres paramètres de contexte. Dans ce cas, l'opérande est représenté par une instance de classe `CalculatedValue`.

Chaque contexte (instance de classe de `contexte`) d'un modèle de coût donné est décrit par un ensemble de paramètres de système de base de données. Ces paramètres sont liés à différentes catégories. La méta-modélisation de ces catégories de paramètres et leurs attributs entraînent de nombreuses classes et énumérations. En raison du manque d'espace, la (Figure 4.11) représente le méta-modèles qui correspondent aux paramètres contextuels. Avec la classification donnée, nous soutenons que tous les paramètres de contexte considérés dans la littérature relèvent de l'une de ces catégorisations. Chaque catégorie est représentée par une classe. Ainsi, les classes de paramètres appartiennent à la même catégorie hérité de cette classe de catégorie racine.

Paramètres de la base de données : les éléments de cette catégorie sont liés à la base de données et aux fonctionnalités différentes qui doivent être fournies par le système de gestion de base de données (SGBD). Grâce à cette catégorie, nous précisons les paramètres de contexte des systèmes de stockage (par exemple relationnel ou non relationnel), la gestion des navires (par exemple, la taille du pool de sauvegarde) et le schéma de la base de données (par exemple, tables / colonnes, table de partitionnement). (Voir Figure 4.12).

Paramètres matériels : en général, les paramètres du contexte matériel des caractéristiques du périphérique, tels que le périphérique de traitement (p. Ex. CPU, GPU des unités de traitement graphiques, etc.), un périphérique de stockage différent (p. Ex. Mémoire principale, disque dur

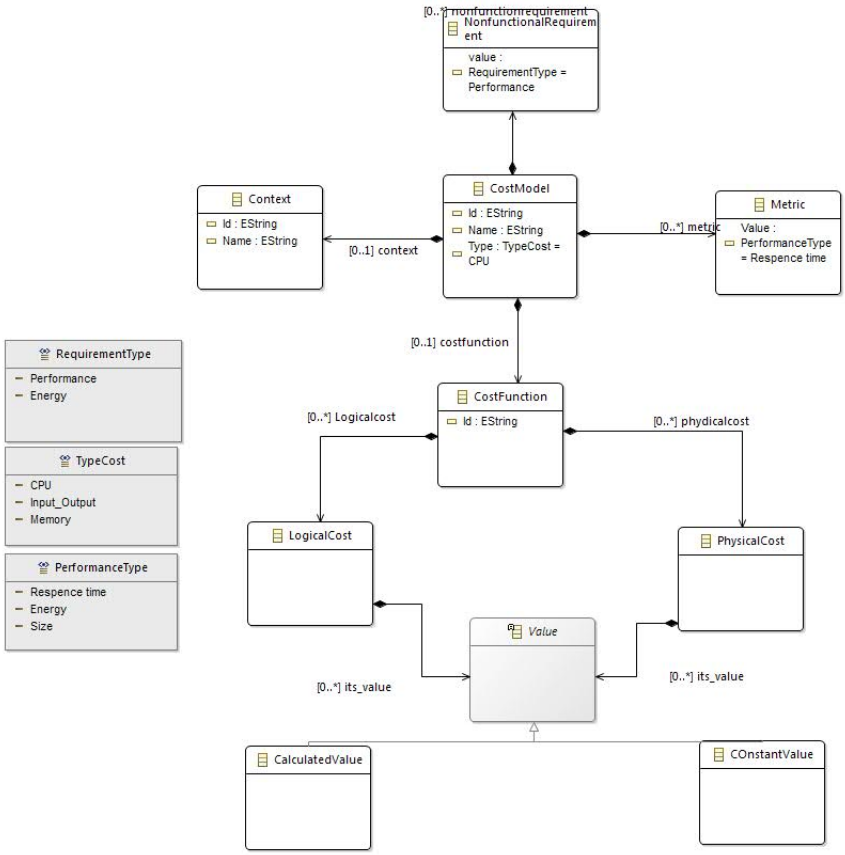


FIGURE 4.10 – Méta-modèle du modèle de coût

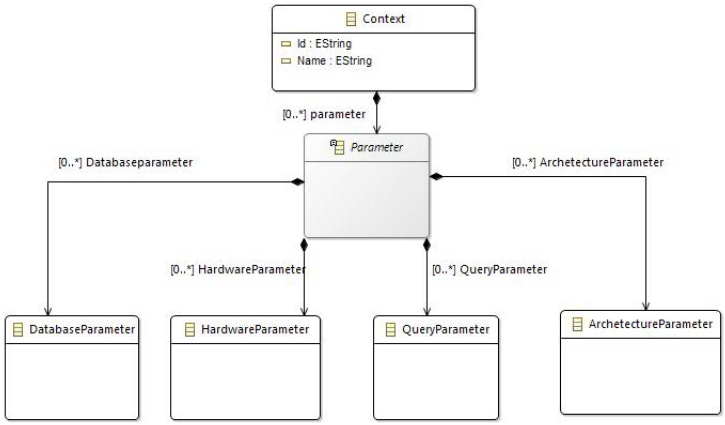


FIGURE 4.11 – Méta-modèle du Contexte

SSD ou disque dur HDD) et dispositif de communication.

Paramètres de requête : les paramètres de requête signifient des concepts utilisés pour effectuer un ensemble d'opérations d'algèbres (join, select, etc.). Le fonctionnement peut être une fonction unaire ou binaire. Le résultat de la requête peut être restreint par un ensemble de prédicats. Ces opérateurs devraient effectuer le plus rapidement possible en exploitant une méthode

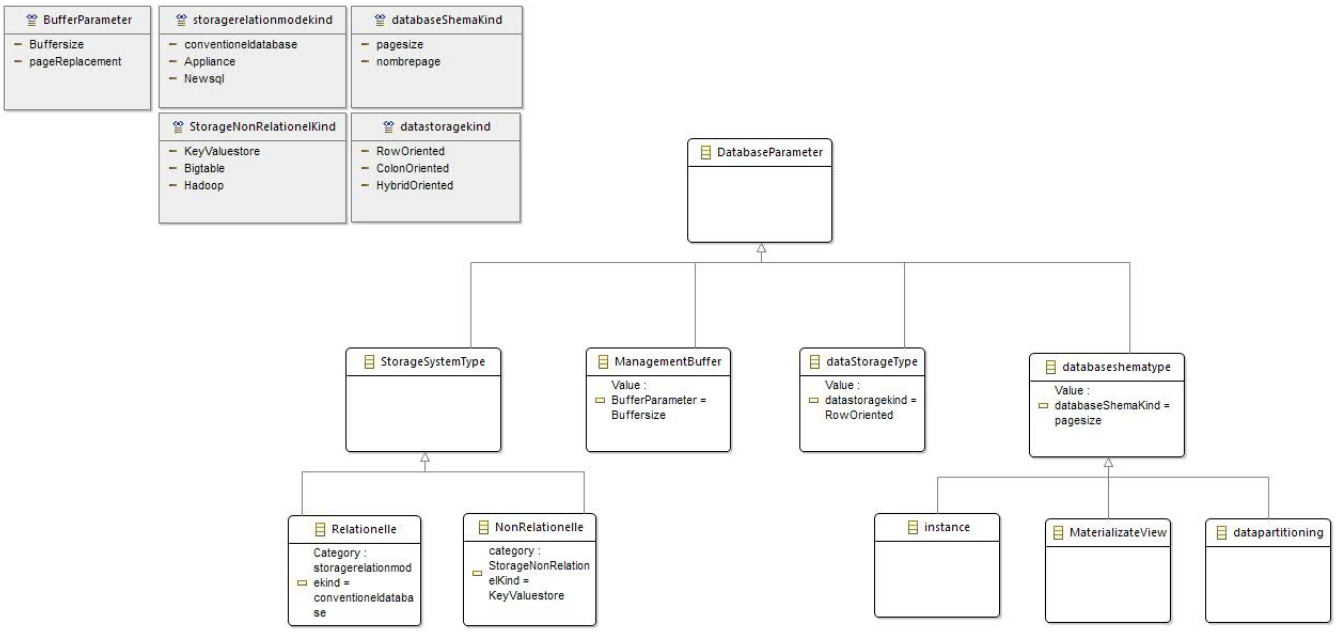


FIGURE 4.12 – Méta-modèle de la base de donnée

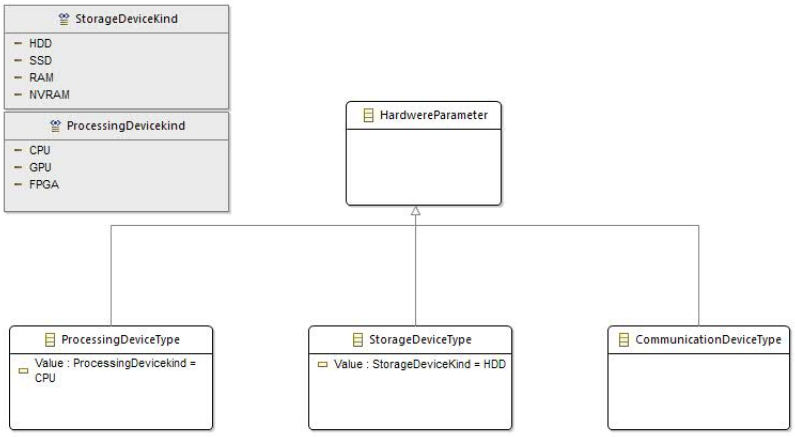


FIGURE 4.13 – Méta-modèle des paramètres matérielles

d'accès sous-jacente (par exemple, des méthodes d'index ou des méthodes d'accès en mémoire [manegold2000optimizing].

Paramètres d'architecture : les éléments de cette catégorie contiennent l'architecture de déploiement telles que : systèmes de base de données distribués ou parallèles, clusters de bases de données ou les Cloud. Ces paramètres de catégorie alimentent le contexte du modèle de coût sur le type de l'architecture du système (par exemple, la mémoire partagée, le disque partagé, etc.) et leurs paramètres comme le nombre de nœuds dans un environnement parallèle. La (Figure 4.16) représente l'élément classe Hypothèse qui contiennent des préférences qui possède un Id, Name et un type qui s'applique soit a priori soit a posteriori soit en mode interactive. Chaque préférence est décrite par une table, des attributs, est des structures physiques.

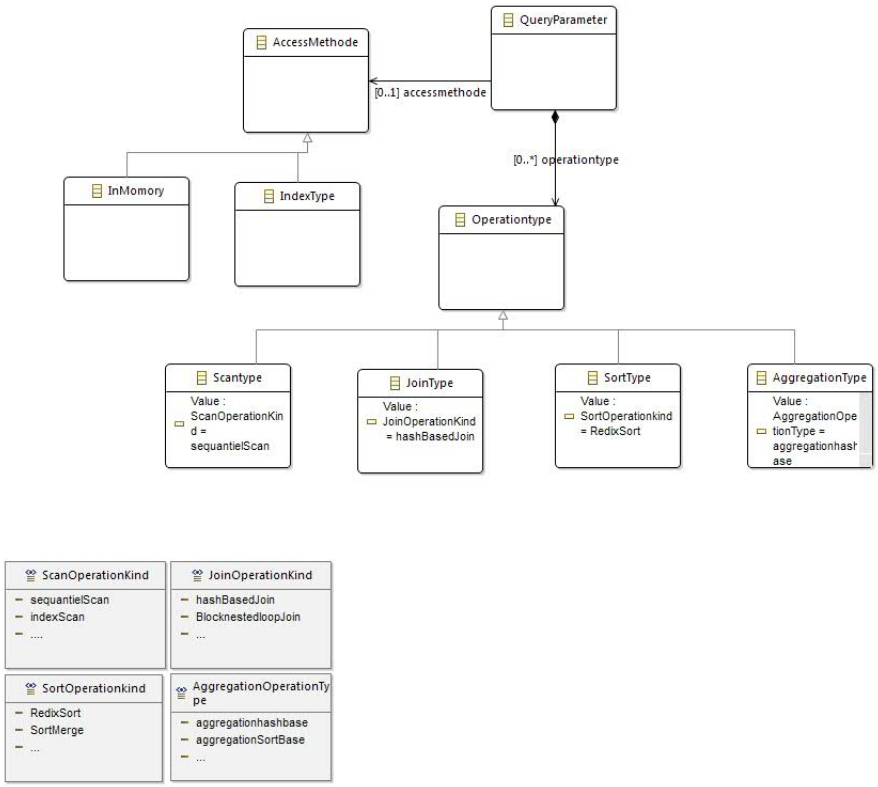


FIGURE 4.14 – Méta-modèle des requêtes

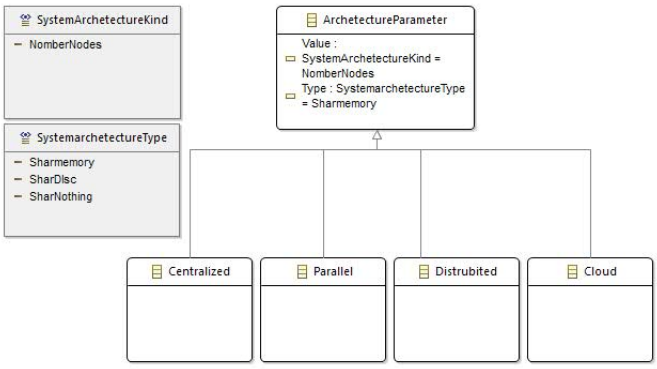


FIGURE 4.15 – Méta-modèle de paramètres d'architecture

4.4.1.4 Exemple d'instanciation

Pour valider nos contributions, nous proposons de dérouler la méthode de conception proposée en instanciant le Framework <A, CM, Hyp, C> :

— **A** : L'algorithme de sélection ;
— **CM** : Le modèle de coût utilisé ;
— **Hyp** : Les Hypothèses considérées ;
— **C** : Le Contexte qui représente l'ensemble de paramètres.

Nous présentons dans cette section une instanciation du Framework proposé pour l'exemple

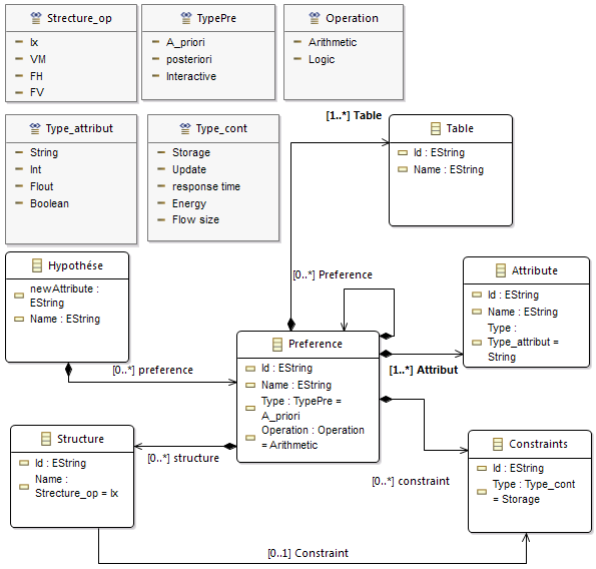


FIGURE 4.16 – Méta-modèle de préférence

présenté dans la section ??. Cette instanciation permet d'une part de tester et de valider notre approche. D'autre part, elle permet de fournir un moyen de construire un advisor de base de données à partir des composantes qui deviennent des briques de bases pour construire cet advisors. Pour illustrer nos propos, nous avons instancié le méta-modèle comme présenté dans la(Figure 4.17). La (Figure 4.17) illustre l'ensemble des concepts de la spécification présentés dans l'exemple précédent.

4.5 Cas d'étude : Les indexes

Cas d'étude : Les index

Définition 1

Un index d'une manière globale est une liste ordonnée.

Définition 2

Un index est un élément qui permet d'optimiser les requêtes pour chercher et acquérir les informations sur la base de donnée, de la même manière qu'un index sur un livre.

Les $\mathcal{IX}$ sont de petites tailles par rapport à la table réelle et sont constituées de données importantes. Par conséquent, une exécution plus rapide des requêtes est possible. Lorsque les requêtes sont générées dans le système, les $\mathcal{IX}$ sont accédées en premier. Si les résultats ne sont pas trouvés dans les $\mathcal{IX}$, les tables réelles du système sont recherchées.

Exemple 1. *Pour créer un index qui comptabilise la somme des ventes de chaque client par jour de notre base de données de vente, la requête SQL s'écrit comme suit :*

```
SQL : CREATE INDEX BITMAP ON Ouvrier(Salaire)
TABLESPACE Sal_espace
```

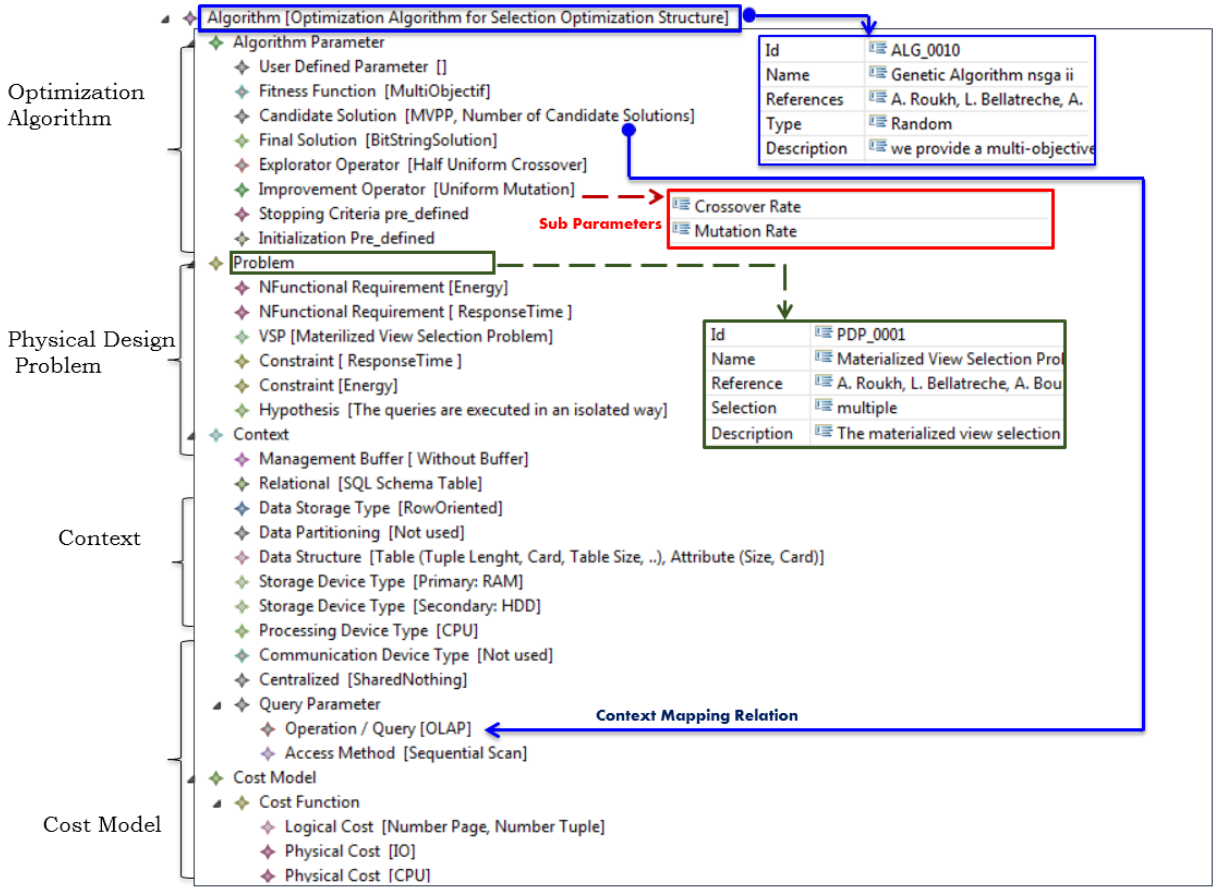


FIGURE 4.17 – Exemple d'instanciation de l'exemple dans ??

Si la relation Commandes contient des dizaines de millions de tuples, la différence entre le calcul de l'agrégation à partir des données de base et à l'aide de l'index peut être de plusieurs ordres de grandeur.

Les défis à relever pour exploiter les indexes sont : (1) identifier les bonnes indexes, (2) exploiter les $\mathcal{IX}$ pour répondre aux requêtes, et (3) mettre à jour efficacement les $\mathcal{IX}$ lors du changement des données de base [Chaudhuri97b].

4.5.1 Problème de sélection des $\mathcal{IX}$

L'administrateur de base de données ne peut pas matérialiser toutes les indexes possibles, car il est limité par certaines ressources comme, l'espace disque et le temps de calcul, ceci est connu comme le problème de sélection des indexes.

Le $\mathcal{PSI}$ consiste à trouver un ensemble approprié d'indexes satisfaisant un ensemble donné de $\mathcal{BNF}$ tels que la performance des requêtes, la fiabilité, l'utilisabilité, etc. Les IX sélectionnées doivent répondre à un ensemble de contraintes telles que le coût de stockage, les coûts de maintenance, etc. Le $\mathcal{PSI}$ est connu pour être un problème NP-complet en raison du fait que l'espace de solution s'accroît de façon exponentielle lorsque la taille du problème augmente. La

formalisation générale de $\mathcal{PSI}$ est défini comme suit : Étant donné :

— un schéma d'une base de données : DB ;
— une charge de requêtes : $\mathcal{W} = \{Q_1, Q_2, \cdots, Q_n\}$;
— un ensemble de contraintes : $\mathcal{C} = \{C_1, C_2, \cdots, C_p\}$;
— un ensemble de besoins non-fonctionnelles : $\mathcal{BNF} = \{nfr_1, \cdots, nfr_k\}$.

Le $\mathcal{PSI}$ consiste à sélectionner un ensemble d'indexes : $IX = \{I_1, I_2, \cdots, I_m\}$ satisfaisant des $\mathcal{BNF}$ et en respectant l'ensemble des contraintes $\mathcal{C}$.

Plusieurs instanciations de cette formalisation existent. Nous proposons de les représenter par un *cube* ayant trois dimensions principales : (i) le $\mathcal{BNF}$, (ii) les contraintes $\mathcal{C}$, et (iii) les algorithmes utilisés pour résoudre le $\mathcal{PSI}$. Ces dimensions permettent de classer les solutions existant dans la synthése bibliographique.

En ce qui concerne les $\mathcal{BNF}$ que les $\mathcal{IX}$ sélectionnés doivent satisfaire, elles couvrent plusieurs objectifs, tels que : la minimisation du temps de réponse des requêtes, la minimisation du temps de réponse et du coût de maintenance. Ces objectifs peuvent être combinés pour donner lieu à une formalisation *multi-objectif* du $\mathcal{PSI}$ comme dans où le temps de réponse et le coût de maintenance ont été considérés.

Les contraintes : Les premières études sur le $\mathcal{PSI}$ supposent l'absence de contraintes. Par la suite et en raison de la grande taille de stockage des $\mathcal{IX}$ sélectionnés, l'espace de stockage devient la principale contrainte que le processus de sélection doit intégrer. Ensuite, le coût de maintenance devient un paramètre important car les $\mathcal{IX}$ sélectionnés doivent être mises à jour une fois les tables de base modifiées. Certaines études ont considéré les deux contraintes. Récemment, des études traitent le cas du déploiement des $\mathcal{IX}$ dans un environnement distribué avec le coût de communication réseaux comme contrainte, ou encore un environnement cloud sous la contrainte de coût monétaire.

Algorithmes de résolution : La résolution de $\mathcal{PSI}$ est généralement réalisée soit par des algorithmes simples ou avancés qui couvrent des algorithmes *déterministes* , algorithmes *aléatoires*, algorithmes *hybrides* , et la *programmation par contraintes*.

Type de sélection : Il existe deux approches pour la sélection des indexes : *statique* et *dynamique*. Une approche de sélection d'index statique suppose que la charge de requêtes est fixe, et choisit ensuite l'ensemble d' indexes à matérialiser. Alors que, dans une approche de sélection d'index dynamique, la sélection est appliquée lorsqu'une requête arrive. Par conséquent, la charge de requêtes est construite progressivement et change avec le temps.

4.6 Architecture de notre advisors

Un advisor prend comme entrée la charge de requêtes, la conception physique d'origine et les contraintes définies par le DBA telles que l'espace maximum pris par les colonnes répliquées dans les partitions. La sortie est composée d'une configuration physique qui améliore de manière optimale le temps d'exécution de cette charge de requêtes et des nouvelles requêtes réécrites sur cette nouvelle configuration physique. La (Figure 4.18) illustre une vue des principaux composants d'un *advisor* qui sont :

— **Le composant d'entré :** l'entré d'un advisor des $\mathcal{SP}$s est donnée comme une charge de

requêtes et un schéma logique de la conception originale et les contraintes. Il produit une nouvelle conception physique et estime l'avantage d'utiliser cette nouvelle conception.

- **Module de réécriture des requêtes :** Cette fonctionnalité de réécriture de la charge de requêtes sur la configuration des $\mathcal{SP}$s sélectionnés.
- **Module de génération des configurations initiales :** génère l'espace de solution initiale à partir de la charge de requêtes et le schéma logique,
- **Collecteur de statistiques** : extrait les statistiques collectées sur la base de données. Cela est possible grâce à Statistics API fournit par les SGBD. Cette API permet d'accéder aux métadonnées d'une BD et de récupérer les différentes statistiques : cardinalités des tables, taille moyenne des n-uplets, facteurs de sélectivité des prédicats, etc.
- **Modèles des coûts des structures physiques** Les configurations candidates sont évaluées en utilisant le module d'évaluation des coûts ou le moule *what-if*. Les statistiques retournées par Statistics API sont utilisées par le *modèle de coût* pour estimer la qualité des différentes structures d'optimisation sélectionnées.
- **Algorithme d'énumération des configurations physiques :** Le but de cette étape est de sélectionner une meilleure configuration pour la charge des requêtes. La configuration générant le minimum de coût, sera choisie comme la meilleure configuration pour les requêtes.
- **Module de génération des $\mathcal{SP}$s :** génère les recommandations sous forme des scripts de création des $\mathcal{SP}$s structures physiques (indexes, partitionnement, vue matérialisée) basé sur les paramètres relatifs à chaque requête ainsi que les tables et les attributs pris en compte .
- **Module d'implémentation :** le Module d'implémentation des structures physiques basé sur l'API *what-if* afin de matérialiser un ensemble des structures physiques (indexes, vues matérialisées, partitionnement) sous une contrainte du coût de maintenance.
- **Les composants *what-if*** sont utilisés pour simuler des $\mathcal{SP}$s comme des partitions physiques, des index et la présence ou l'absence des méthodes de jointure. On distingue quatre composants utilisant les composants *what-if* : le composant d'indexation automatique, le composant de partitionnement automatique, le composant de création automatique des vues matérialisées.

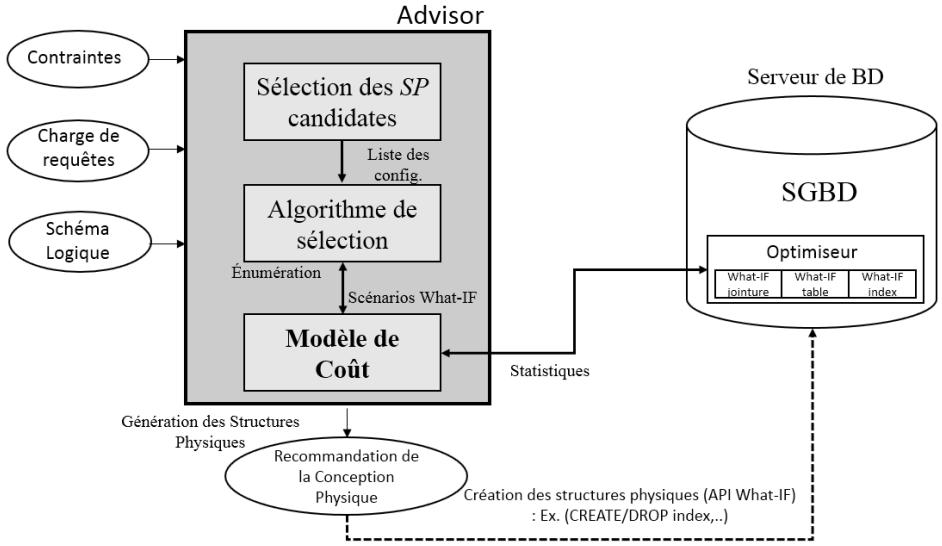


Figure 4.18 – Composantes d'un advisor des structures physiques

4.6.0.1 Algorithmes d'optimisation et les modèles de coûts

De nombreux problèmes de conception physique (CP) peuvent être pris en compte au cours de cette phase [3], par exemple le BDA (Database Administrator) a des tâches comme le choix ou le développement des algorithmes de sélection pour les structures physiques d'optimisation tels que les vues matérialisées PSV, les indexes PSI, le partitionnement PSPH, Buffer Management BMP, Query Scheduling QSP[14], La planification conjointe des requêtes et la gestion des tampons (QSBM), Taille du système.

La recherche d'un ensemble approprié de structures physique ($\mathcal{SP}$) doit satisfaire un ensemble donné de BnF tel que la performance d'interrogation, la fiabilité, la facilité d'utilisation, etc. Le problème de conception physique doit répondre à un ensemble de contraintes telles que le coût de stockage, le coût de maintenance caractérisé, etc. problème NP complet en raison du fait que l'espace de solution croît de façon exponentielle que la taille du problème augmente. Le problème de la conceptions physique de base de données, qui a donné un problème lié à un ensemble de contraintes, un ensemble de BnF qui représente la fonction objectif, un algorithme de sélection, des paramètres de contexte qui incluent la base de données de schéma de base de données, une charge de travail de requête et les paramètres pour l'environnement d'exécution telque la configuration du dispositif matériel. Les structures physiques sont recommandées par un algorithme de sélection, de sorte que toutes ces contraintes sont celles qui satisfont un BnF. Les algorithmes de sélection utilisés regroupent les algorithmes déterministes, des algorithmes aléatoires, etc.

Ces algorithmes sont conduits par un modèle de coût (MdC) pour quantifier la solution proposée du problème d'optimisation dans la conception physique. Ce problème de conception physique est illustré à la(Figure 4.19).

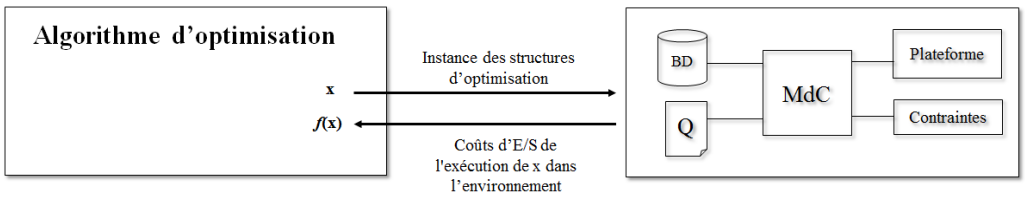


FIGURE 4.19 – L'utilisation du MdC dans les algorithmes d'optimisation

4.7 Algorithme Glouton

La sélection d'une configuration d'index mono-attributs se déroule en trois étapes :(1) l'identification des attributs indexables, (2) l'initialisation de la configuration et (3) l'enrichissement de la configuration actuelle par l'ajout de nouveaux index. Dans la première étape, l'ensemble des requêtes est analysé afin d'extraire les attributs indexables. Ces attributs sont les attributs sur lesquels un prédicat de sélection est défini dans la charge de requêtes. Les attributs indexables candidats sont choisis parmi les attributs indexables de faible et de moyenne cardinalité. La deuxième et la troisième étapes sont réalisées au sein d'un algorithme glouton que nous avons proposé. Cet algorithme est basé sur le modèle de coût que nous avons présenté dans la Section 4.8.

L'algorithme commence par une configuration composée d'un index mono-attribut défini sur l'attribut ayant une cardinalité minimum noté BJImin. La configuration initiale est améliorée itérativement par l'ajout d'index définis sur d'autres attributs non encore indexés. L'algorithme s'arrête lorsque deux conditions sont satisfaites : aucune amélioration n'est possible et l'espace de stockage est consommé. Notre algorithme glouton est décrit dans(Algorithme 4.1).

Algorithme 4.1 Algorithme glouton pour la sélection d'index

Entrée : — IDX_j : Index défini sur l'attribut A_j.
— $Taille(IDX_j)$: coût de stockage de IDX_j
— $C[Q, \phi]$: coût d'exécution de Q avant indexation. [Q, C] : coût d'exécution de Q en utilisant la configuration C.
— Charge Q, Ensemble d'attributs indexables $IDXASET$, Espace de stockage S.
Sortie : Configuration finale C_f

$C_f \leftarrow \text{IDX}_{min}$;
S $\leftarrow S-$Taille(IDX$_{min}$);
IDXSET $\leftarrow IDXSET-\text{A}_{min}$; ▷ A_{min} est l'attribut utilisé pour générer IDX_{min}
tant que Taille (C_f)≤ S **faire**
  pour (chaque $A_j \in$ IDXSET) **faire**
    si ((C[Q', ($C_f \cup IDX_j$))] < C[Q',∅] **ET** (($Taille$(C$_f \cup$ IDXj) ≤ S))) **alors**
      $C_f \leftarrow \text{C}_f \cup IDX_j$;
      Taille(C_f) ← $Taille$(C$_f$) + Taille(IDX_j) ;
      IDXSET ← IDXSET - A_j ;
    fin si
  fin pour
fin tant que

4.8 Modèles de coût pour les index

Le modèle de coût que nous avons défini permet d'estimer le nombre d'E/S nécessaires (exprimé en nombre de pages chargées) pour exécuter une requête.

4.8.1 Coût d'accès aux données par IJB

En résumé, le coût d'exécution d'une requête qui accède aux données par IJB noté I est
Cost(Q,I)=log | A | $-1 + \frac{|A|}{(m-1)} + d\frac{\|F\|}{8PS} + \| F \| (1 - e^{-Nr/\|F\|})$
avec Nr= d | F |$_{\overline{|A|}}$

4.8.2 Tous les résultats intermédiaires tiennent en mémoire

On désigne par Cost le cout de la requête suivant une méthode d'accès (IJB), par différente algorithme de jointure dans les supports de stockage SD =(SSD,HHD).
L'optimiseur effectue la première jointure ensuite charge les tables de dimension une à une pour effectuer une jointure par hachage de ces dernières avec les résultats intermédiaires. Le coût peut être calculé par la formule suivante :
cost=3 ×(| F | + | D_1 |) + $\sum_{i=2}^{k}$ 3× | D_i |

Tableau 4.1 – Paramètres utilisés dans les formules de coût

Paramétré	Signification
F	la table des faits
$\mid R \mid$	nombre de pages nécessaires pour stocker la table R
$\parallel R \parallel$	nombre de tuples de la table R
A	attribut d'indexation
$\mid A \mid$	cardinalité de l'attribut A
PS	taille en octet de la page système
W(X)	taille en octet d'un tuple de la table T ou de l'attribut X
d	Nombre de bitmap utilisé pour une requête
m	Ordre d'un B-arbre
Nr	le nombre de tuples fait accédés par d bitmap

4.9 Conclusion

Comme nous avons vu au cours de cette section, le principal objectif lors du traitement de requête par un SGBD est la minimisation du temps de réponse des requêtes, à l'aide de techniques sophistiquées, qui comprennent des algorithmes pour sélectionner et exécuter un plan optimal pour une requête.

Afin d'intégrer les préférences de DBA dans une base de données, il faut, en premier lieu, l'étudier pour identifier ses composantes qui peuvent avoir un impact sur les performances du système. Dans ce chapitre nous avons décrit les trois types des *advisors* : en ligne, hors ligne et interactifs. Nous avons identifié l'opportunité d'intégrer les préférences dans la phase de conception physique, plus précisément, dans le choix d'une structure d'optimisation. Nous nous sommes focalisés sur la technique redondante qui est les indexes, et nous avons proposé de reformuler le problème de sélection d'index en un problème d'optimisation ou il faut minimiser le temps en respectant certaines contraintes.

Les besoins et spécifications exprimés dans ce chapitre représentent la base sur laquelle notre solution est fondée. En effet, le résultat de l'étape d'analyse et de modélisation de notre domaine influe directement sur les résultats des étapes qui restent. Nous présentons dans le chapitre suivant la solution que nous avons proposée pour répondre aux besoins de DBA ainsi que la conception de cette solution.

Sommaire

Chapitre

5

L'implémentation et la mise en œuvre de notre application

Sommaire

5.1 Introduction

Ce chapitre a pour objectif de visualiser et d'apprécier les résultats des suggestions faites lors de l'étude de l'existant et des souhaits émis lors de l'étude des besoins, et également, elle rend tangible la conception et cela en procédant comme suit :

- Présentation des différents outils (technologiques et logiciels) utilisés pour le Développement de notre système.
- Présentation de la conception de notre outil, afin de figurer le travail fait et D'illustrer les principales fonctionnalités réalisées.

5.2 Présentation des technologies de développement utilisées

Nous présentons dans cette section les outils utilisés pour le développement de notre application (ORACLE, IDE Netbeans, Navicat for ORACLE, Enterprise Architect, Ecore . . . etc).

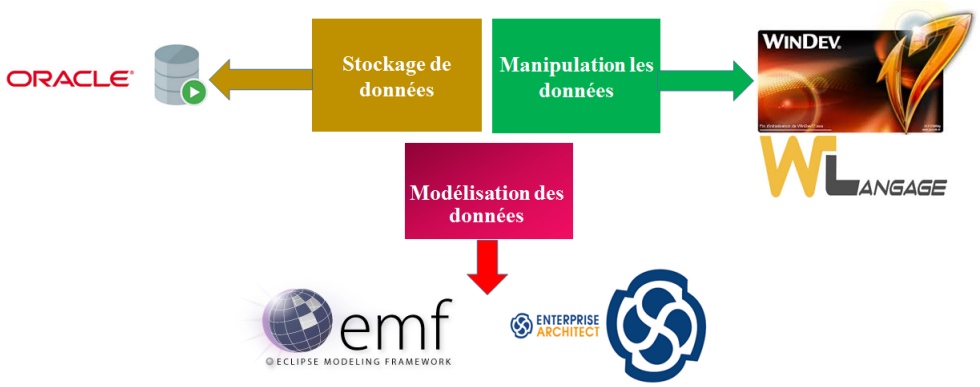


FIGURE 5.1 – La technologie de développement utiliser

5.2.1 Présentation du langage WL

Le WLangage est un langage de programmation de 5eme génération.(L5G) a la fois simple et très puissant Inclus dans les outils de développement WinDev, WebDev et WinDev Mobile, il est propriétaire et ne peut être manipulé qu'avec les outils PC SOFT. Le WLangage basé sur le concept de résolution de problèmes en utilisant des contraintes données au programme, plutôt que d'utiliser un algorithme écrit par un programmeur (3e génération) ou une spécification formelle de représentation en tables et interrogations SQL (4e génération). La plupart des langages fonctionnant par contraintes ou par programmation logique ainsi que quelques langages déclaratifs sont des langages de cinquième génération. Le WLangage peut également s'appuyer sur le framework Java pour une partie de ses fonctionnalités, ce qui permet une indépendance relative et limitée du fichier exécutable par rapport au système d'exploitation cible. Il en va de même dans WebDev, où le WLangage peut s'appuyer sur le framework PHP, sans toutefois permettre d'utiliser toutes les possibilités de ce dernier.

5.2.2 Eclipse Modeling Framework EMF

Le projet EMF est un cadre de modélisation pour la construction d'outils et d'autres applications basées sur un modèle de données structuré. À partir d'une spécification de modèle décrite dans XMI, EMF fournit des outils et un support d'exécution pour produire un ensemble de classes Java pour le modèle, ainsi qu'un ensemble de classes d'adaptateurs qui permettent l'affichage et l'édition par commande du modèle et un éditeur de base. EMF inclure un méta-modèle Ecore pour décrire les modèles et le support d'exécution pour les modèles.

5.2.3 ORACLE 11.g

Pour la gestion de la base de données de notre application on a choisi ORACLE 11.g. c'est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, Il fonctionne sur de nombreux systèmes d'exploitation différents incluant Linux, Mac OS X, Windows . Oracle a été édité par la société du même nom (Oracle Corporation - http ://www.oracle.com), leader mondial des bases de données. La société Oracle Corporation a été créée en 1977 par Lawrence Ellison, Bob Miner, et Ed Oates. Elle s'appelle alors Relational Software Incorporated (RSI) et commercialise un

Système de Gestion de Bases de données relationnelles (SGBDR ou RDBMS pour Relational Database Management System) nommé Oracle.

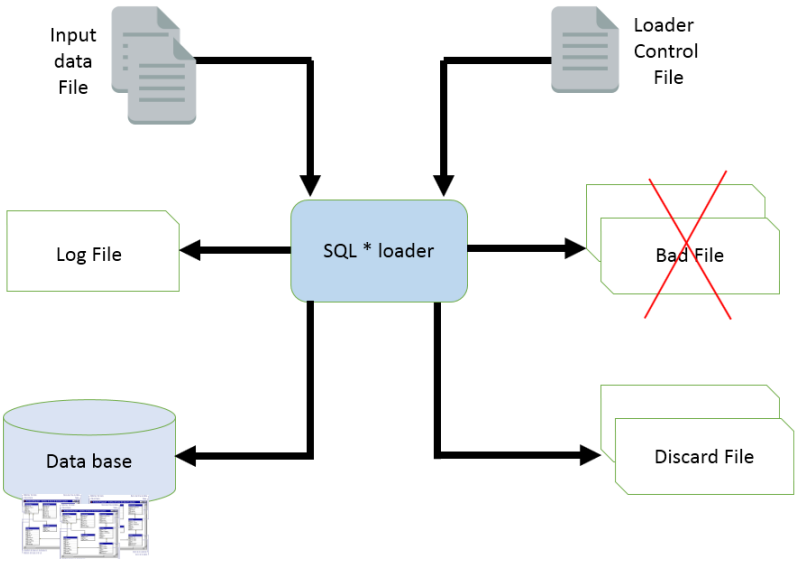


FIGURE 5.2 – Mécanisme de SQL Loader

5.2.4 Navicat for ORACLE

Navicat For ORACLE est un outil pour gérer les bases de données ORACLE, et pour convertir des bases en XML, CSV, MS (Excel et Access), et autres formats, en des bases ORACLE. Les autres fonctions majeures du programme est l'import et l'export, un support Unicode, le tunnel HTTP/SSH, la synchronisation des données, le transfert des données, la mise en place d'un questionnement visuel, et la construction d'un rapport visuel, et bien d'autres encore. De plus, les fonctions de navigation vous servent à l'importation de données d'ODBC. Ce programme est un software et a donc une utilisation limitée dans le temps.

5.2.5 Entreprise Architect

est un logiciel de modélisation et de conception UML, édité par la société australienne Sparx Systèmes. Couvrant, par ses fonctionnalités, l'ensemble des étapes du cycle de conception d'application, il est l'un des logiciels de conception et de modélisation les plus reconnus.

5.2.6 Présentation de SQL Loader

Une séance typique de SQL Loader prend en entrée un fichier de contrôle, qui contrôle le comportement de SQL Loader, et un ou plusieurs fichiers de données. La sortie de SQL * Loader est une base de données Oracle (où les données sont chargées), un fichier journal, un mauvais fichier, et potentiellement, un fichier de défausse.

En ce qui concerne cette étape (chargement de tables) on a pris le chemin de DATAFILE qui contient les enregistrements des tables pour les mettre dans le Control file (le script pour extraire les DATAFILE dans les tables) puis on exécute le SQL Loader à partir de l'invite de commande et on s'authentifier en

utilisant notre compte ORACL et on finit par mettre le chemin du Control File. La Figure 5.3 et Figure 5.4 illustre le mécanisme expliqué par avant sous exemple de la table «DIM_DATE» :

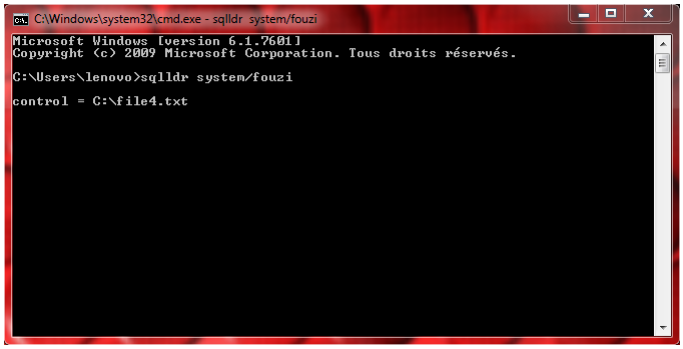


FIGURE 5.3 – Exemple de chargement de table « DIM_DATE »

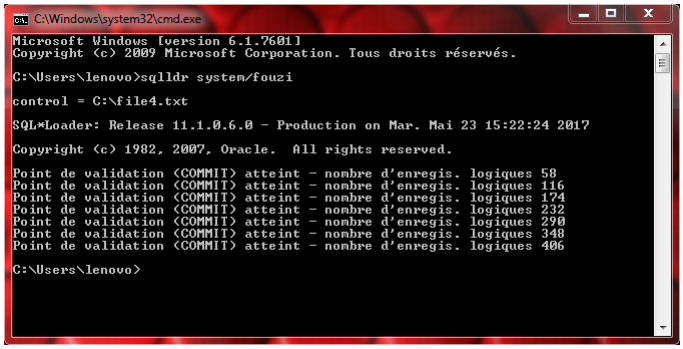


FIGURE 5.4 – Chargement de la Table Dim_{Date}

5.3 Présentation de notre outil

5.3.1 Objectifs

Notre outil gère principalement la technique d'optimisation des index est se focalise sur l'algorithme de Glouton. Les objectifs principaux sont :

- Permettre la visualisation de l'état courant de la base de données : la structure de la BD (schéma, attributs, taille de chaque table, définition de chaque attribut, etc.) et la charge définie sur la BD (le nombre de sélections, le nombre de jointures, attributs de sélection, les facteurs de sélectivité de chaque prédicat, etc.)
- Évaluer le coût E/S d'une charge de requêtes (un tableau affichant les résultats calculés) et faire la comparaison entre l'optimisation avec et sans Index.

5.3.2 Conception de l'outil

Dans cette section nous allons consacrer une première partie pour présenter quelques diagrammes UML, ils ont été élaborés en fonction d'avancement et de développement de notre projet. Et en ce qui concerne

la deuxième partie, elle se base sur la présentation de l'application via des prises d'écrans, afin de figurer le travail fait et d'illustrer les grandes et principales fonctionnalités réalisées.

5.3.2.1 La modélisation du système

Parmi les diagrammes UML largement connus par les informaticiens, on cite :

- le diagramme de cas d'utilisation : il permet de recueillir, d'analyser et d'organiser les besoins. Avec lui débute l'étape d'analyse de notre système.
- Diagramme de séquence : il permet de représenter des interactions entre objets et acteurs, selon un point de vue temporel avec une chronologie des envois de messages, c'est un type de diagramme d'interaction.
- Diagramme de déploiement : il permet de représenter l'architecture physique d'un système. Ils montrent la distribution des composants logiciels sur la base d'unités d'exécution.

Le seul acteur dans notre mécanisme est l'administrateur de la base de Données (DBA)

(a) **Diagramme des cas d'utilisation** : Un cas d'utilisation est une manière spécifique d'exprimer l'utilisation ou les services d'un système. Afin d'exprimer l'utilisation de notre système, nous avons modélisé les fonctionnalités fournies par notre outil ainsi leurs dépendances entre eux par un diagramme de cas d'utilisation de standard UML

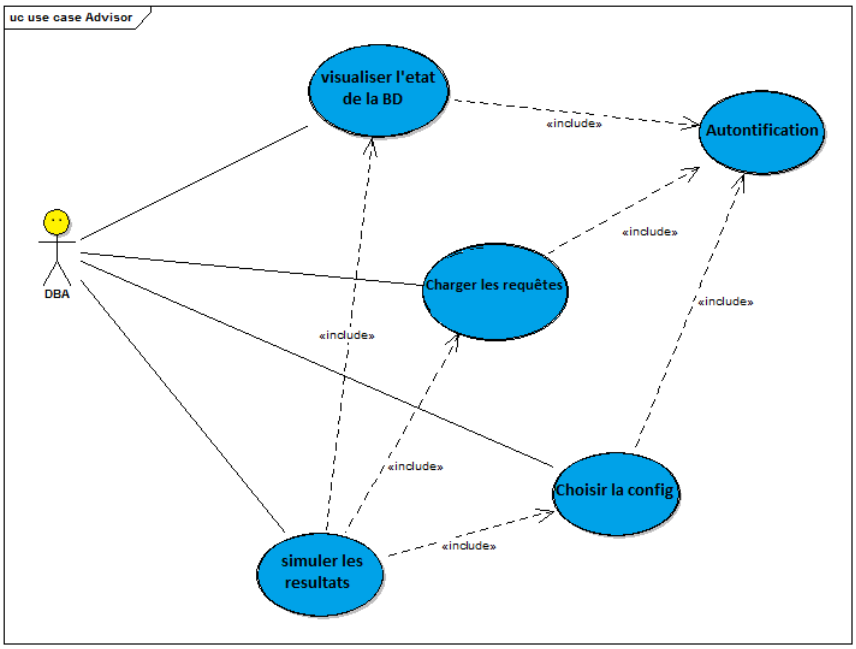


FIGURE 5.5 – Diagramme de cas d'utilisation

(b) **Diagramme de séquence** : Une séquence est une scénarisation théorique d'un cas d'utilisation précis en impliquant toutes les possibilités auxquelles ce dernier peut être confronté. Dans cette partie, on va présenter deux diagrammes de séquence comme suit :

- ***Visualiser la base de données*** : Le système vérifie ces paramètres au près du SGBD, puis se connecte et charge la liste des bases de données existants dans le compte du User et

l'affiche. Au final, Après le choix de BD, le système extrait, à partir de la méta-donnée, les informations pertinentes des tables de cette BD et ces attribut ensuite les affiches.

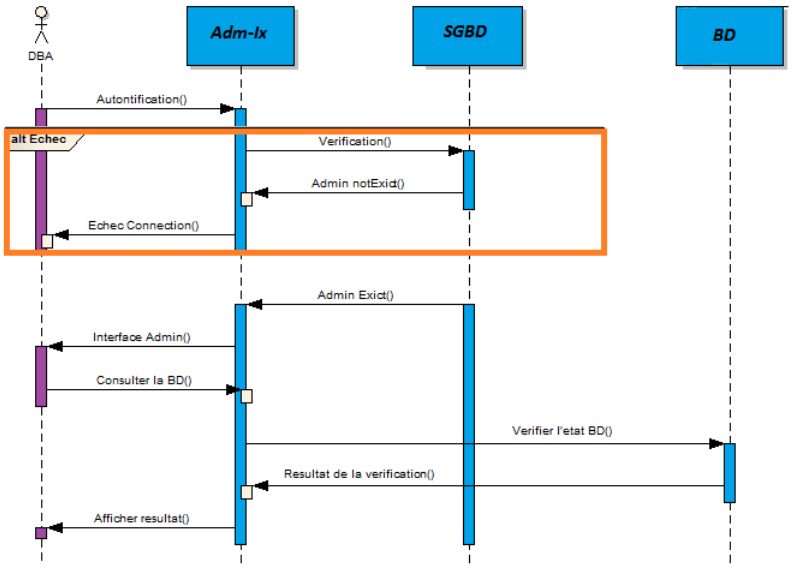


FIGURE 5.6 – Diagramme de séquence : visualisation de BD

— ***Chargement des requêtes*** : Après avoir chargé la base de données. L'administrateur doit fournir une liste de requêtes qui sera utilisée pour optimiser la BD. Ceci est réalisé soit en chargeant le fichier contenant ces requêtes, soit en ajoutant les requêtes une par une. Dans le premier cas c'est le système qui va extraire ces requêtes à partir du chemin spécifié par l'administrateur. Toutes les requêtes introduites seront analysées syntaxiquement par le système. Ce dernier vérifie si les requêtes, respectent la syntaxe d'une requête SQL de sélection, et référencent des objets de la base de données. L'état de chaque requête (correcte ou non correcte) sera affiché au final.

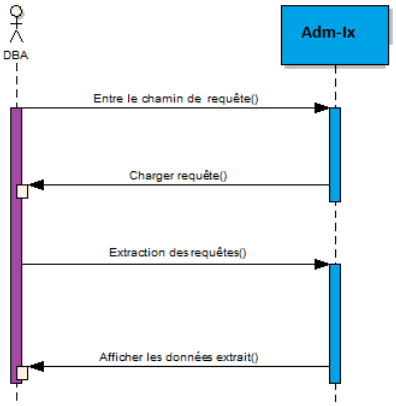


FIGURE 5.7 – Diagramme de séquence : chargement des requêtes

(c) **Diagramme de déploiement** : Nous présentons dans cette section un diagramme de déploiement proposé par UML (Voir Figure ci-dessous). Un diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux.

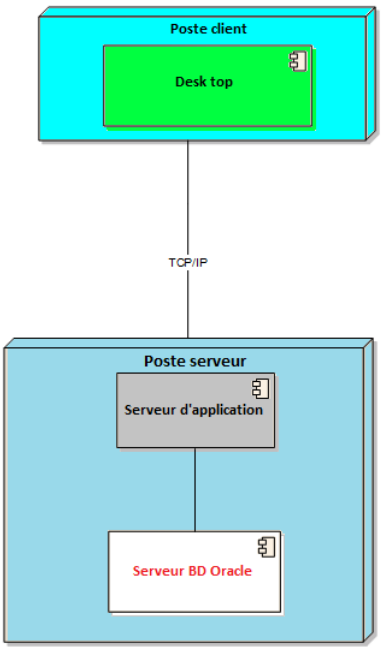


FIGURE 5.8 – Diagramme de déploiement

Passant maintenant à la deuxième partie, concernant la présentation du prototype réalisé nous montrons quelques fonctionnalités en prises d'écrans.

5.3.2.2 Présentation des interfaces

(a) ***Architecture de notre outil*** : Dans cette section nous allons présenter l'architecture globale du fonctionnement de notre outil figure Figure 5.9. Cette figure montre que le DBA commence par visualiser l'état actuel de la base de donnée selon trois types d'informations : informations sur la BD, sur les requêtes et sur le système physique. Les informations concernant la base peuvent être obtenues automatiquement en accédant aux catalogues de la base de données. Après la visualisation, le DBA peut personnaliser la configuration à évaluer :

(b) ***Détail de l'application*** :

— *Interface d'authentification Au SGBD* : Avant d'accéder à l'application, l'utilisateur doit d'abord s'authentifier en utilisant les paramètres de connexion Au SGBD (Oracle) qui lui ont été donnés (login, mot de passe).

— *Visualiser une base de données* : Après l'authentification au SGBD Oracle et récupère certaines informations tels que : le schéma de la BD, les tables et leurs détailles, et leur attributs.

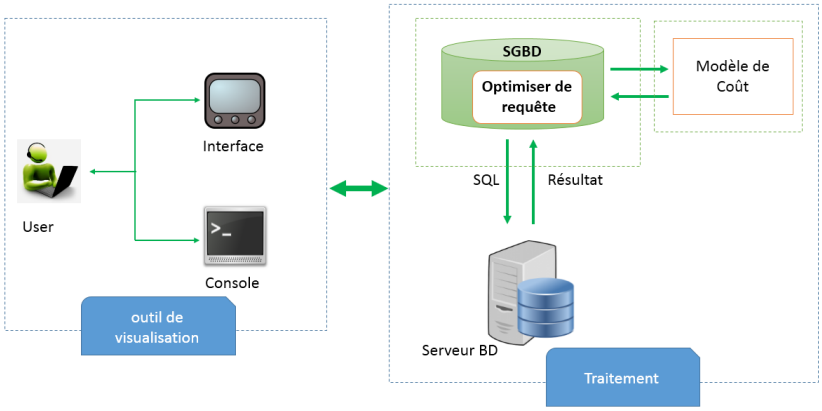


FIGURE 5.9 – L'architecture globale de notre l'outil

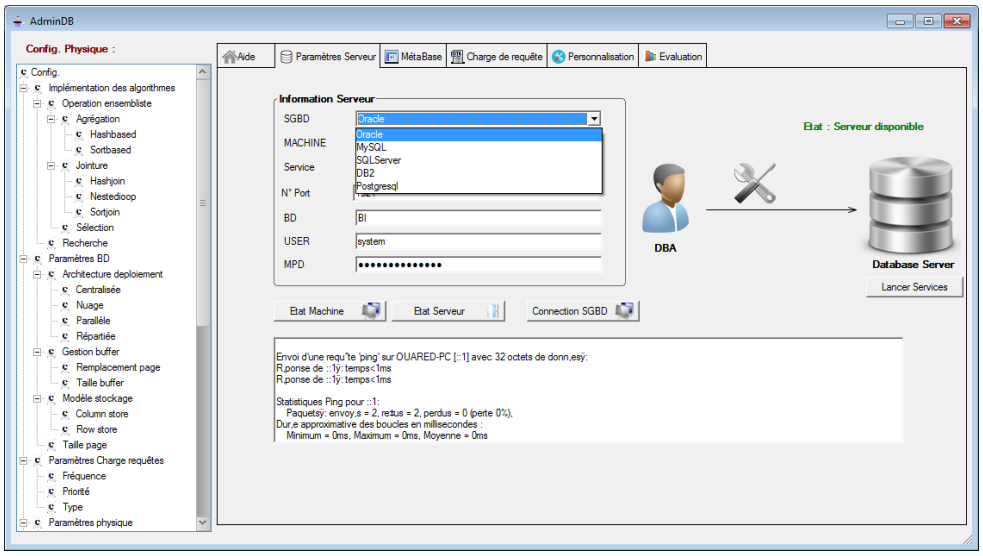


FIGURE 5.10 – Interface d'authentification.

```
30 SELECT SEGMENT_NAME, BYTES/1024
31                     FROM dba_segments
32                     WHERE SEGMENT_TYPE='TABLE' AND OWNER= 'BI'
33                     AND SEGMENT_NAME NOT LIKE '%$%'
34                     ORDER BY SEGMENT_NAME
```

Listing 5.1 – Exemple : Script d'extraction des informations des tables

```
35
36 TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH
37                     FROM ALL_TAB_COLUMNS
38                     WHERE table_name ='CUSTOMER'
39                     or table_name ='PART'
40                     or table_name ='DIM_DATE'
41                     or table_name ='LINEORDER'
```

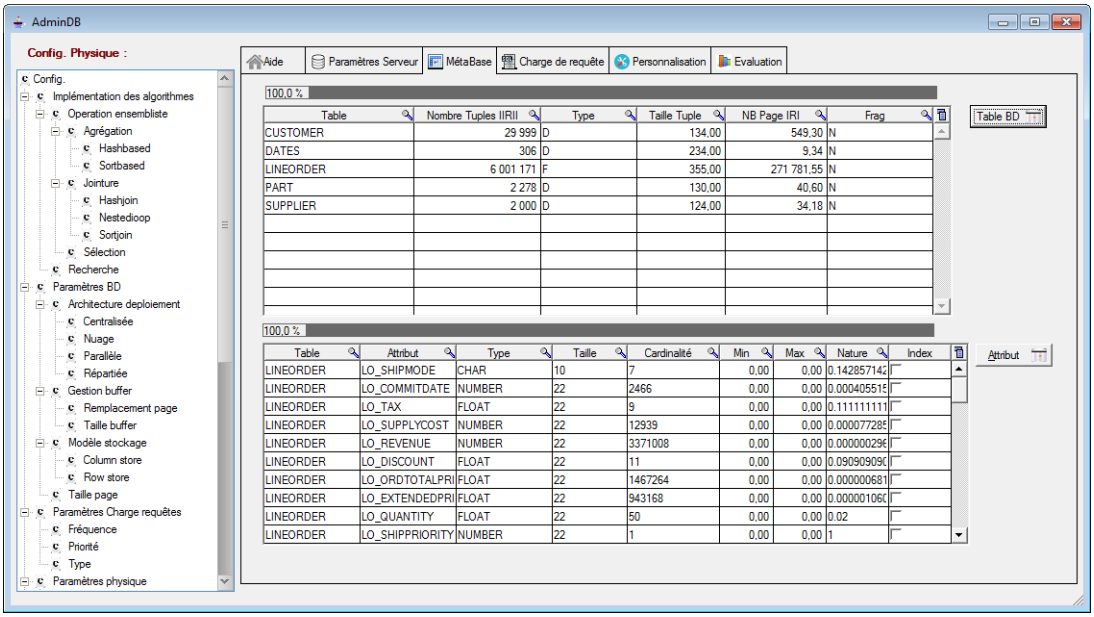


FIGURE 5.11 – Interface de chargement de méta-base.

```
42                or table_name ='SUPPLIER' ORDER BY COLUMN_NAME
```

Listing 5.2 – Exemple : Script d'extraction des informations des attributs

— *Chargement de requêtes* : Le DBA permet également de spécifier la charge de requêtes à estimer, à partir d'un emplacement sur disque. Après le chargement le DBA procède à l'identification de la configuration à estimer (ces préférences) Ceci est effectué manuellement ou automatiquement par la configuration par défaut de l'outil.

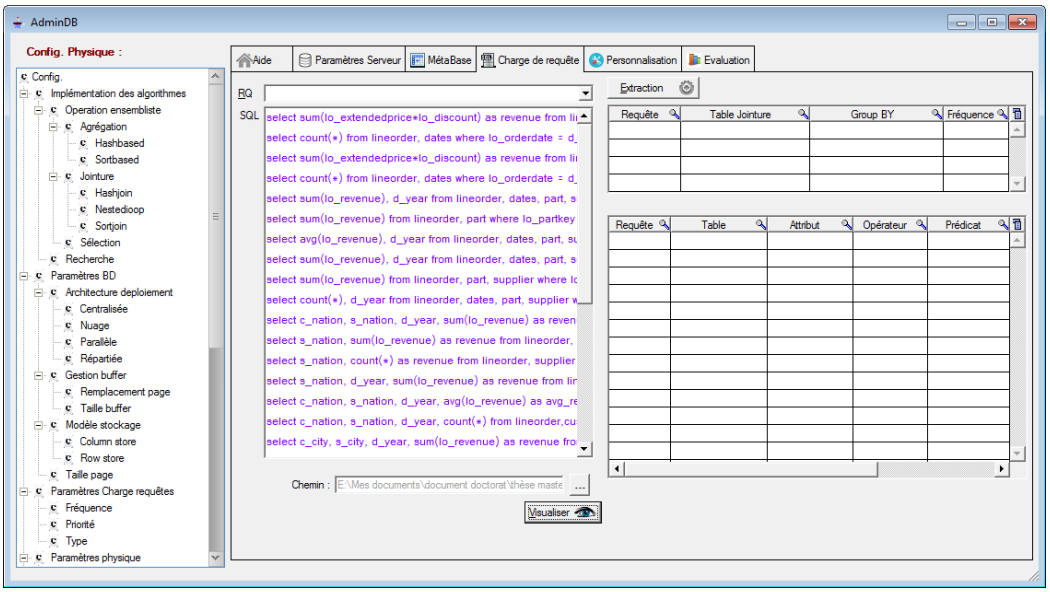


FIGURE 5.12 – l'interface de chargement de requêtes

— *Configuration des paramètres* : Après le chargement de la Méta-base et de la charge de requêtes, le DBA procède à l'identification de la configuration à estimer à savoir : le facteur du modèle de cout, le mode de sélection (manuel automatique) des 12 attributs.

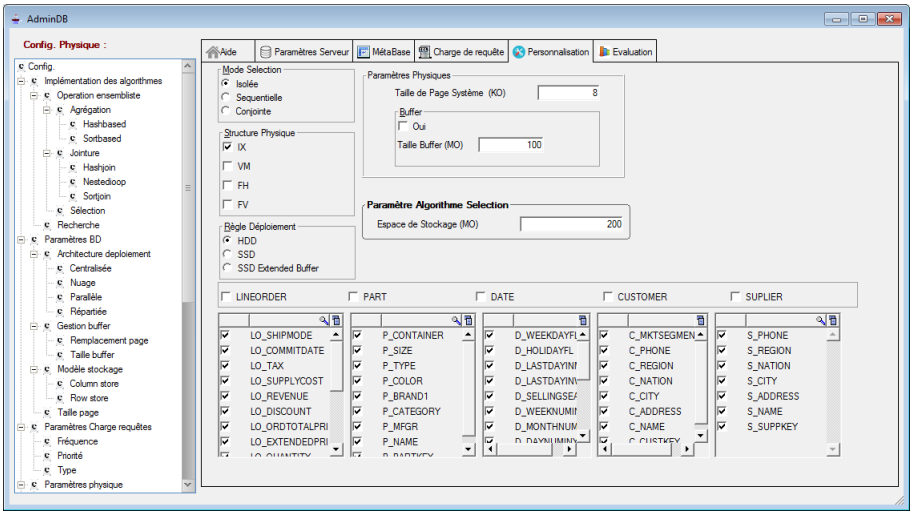


FIGURE 5.13 – Interface de configuration des paramètres

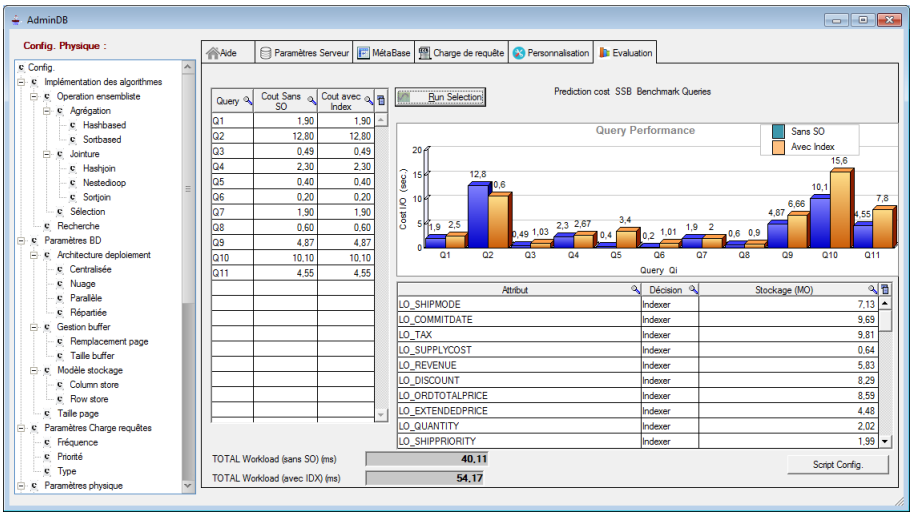


FIGURE 5.14 – Interface de visualisation de résultat

— *Visualisation de résultats* : L'outil permet de visualiser les résultats, il montre le coût en terme d'entre sortie disque.

5.4 Conclusion

Dans ce chapitre, nous avons présenté notre outil développé, ce dernier permettant d'assister le DBA dans leur tâche pour simuler le comportement de la requête selon différentes configuration. Il offre des interfaces conviviales et simples pour permettre aux administrateurs de naviguer et de visualiser l'état

en cour de BD. Une autre caractéristique de l'outil est sa capacité à se connecter aux services oracle pour visualiser les paramètres physiques et réaliser une analyse syntaxique de la charge de requêtes SQL.

Quatrième partie

Conclusion et perspectives

Conclusion

"To accomplish great things, we must not only act, but also dream;
not only plan, but also believe".
Anatole France (1844-1924)

5.5 Conclusion

L'intérêt majeur de ce mémoire est d'assisté le DBA pour effectuer ces tâche. Nous avons effectué une recherche d'informations pour se familiariser avec le sujet, ce qui nous a permis de produire une synthèse bibliographique sur le cœur de ce travail. Nous avons présenté une nouvelle approche basée sur l'ingénierie dirigée par les modèles afin de faciliter et d'aider l'administrateur de la base de donnée DBA dans les tâches de conception physique et de tuning. Notre outil permet à l'administrateur d'effectuer plusieurs choix, comme le choix de ces préférence, les objets de base de donnée (tables, attributs) participant dans le processus d'optimisation, la charge de requêtes à optimiser, etc. Si l'administrateur n'est pas satisfait de ces recommandations, il peut revenir en arrière pour revoir ses choix, par exemple éliminer ou ajouter un attribut candidat. Le travail présenté de ce rapport s'articule dans trois points essentiels : l'analyse de domaine, la modélisation de domaine par des Meta-modèles et implémenter une instance de ce modèle pour traiter le problème de sélection des indexes. Notre outil prototype présente les fonctionnalités suivantes :

- Interface d'authentification.
- Interface de méta base.
- Interface des préférences.
- Interface des résultats.

Il est souhaitable de fournir les indications aux concepteurs et programmeurs d'un système d'interrogation particulier, ou, encore mieux, d'insérer des contrôles, afin de prévenir autant que possibles les erreurs de l'instanciation. Notre travail ouvre plusieurs perspectives, nous pouvons citer :

- Ajouter d'autres préférences.
- Implémenter d'autre Algorithme de sélection.
- Enrichir l'outil par d'autres techniques d'optimisation, comme VM, FH, et FV.
- Détailler les dimensions identifiées (Dispositif de traitement : GPU, FPGA).
- Généraliser l'application sur d'autre SGBD

.

Bibliographie

[1] S Agrawal. « Automatic sql tuning in oracle 10g ». In : *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*. 2004 (cf. p. 27).

[2] K. Aouiche. « Techniques de fouille de données pour l'optimisation automatique des performances des entrepôts de données ». Thèse de doct. Lyon 2, 2005 (cf. p. 24, 29–31, 33).

[3] L. BELLATRECHE. « UTILISATION DES VUES MATERIALISEES, DES INDEX ET DE LA FRAGMENTATION DANS LA CONCEPTION LOGIQUE ET PHYSIQUE D'UN ENTREPOT DE DONNEES ». Thèse de doct. 2000 (cf. p. 60).

[4] L. Bellatreche et K. Boukhalfa. « Une répartition statique et dynamique de l'espace entre les vues matérialisées et les index dans les entrepôts de données ». In : *International Symposium on Programming and Systems (ISPS'05), USTHB-Alger*. 2005 (cf. p. 29, 33, 34).

[5] L. Bellatreche, A. Roukh et S. Bouarar. « Step by Step Towards Energy-aware Data Warehouse Design-5th European Summer School on Business Intelligence (eBISS 2016), Tours, France, July 3-8, 2016, Tutorial Lectures ». In : *Lecture Notes in Business Information Processing. Springer* (2016) (cf. p. 10).

[6] R Bouchakri. « Une approche dirigée par la classification des attributs pour fragmenter et indexer des entrepôts de données ». Thèse de doct. Ph. D. Thesis, Ecole nationale Supérieure d'Informatique (ESI), Oued Semar, Alger, 2009 (cf. p. 28).

[7] L. Bouganim, D. Florescu et P. Valduriez. « Dynamic load balancing in hierarchical parallel database systems ». Thèse de doct. INRIA, 1996 (cf. p. 39).

[8] L. Bouganim et al. « A dynamic query processing architecture for data integration systems ». In : *IEEE Data Eng. Bull.* 23.2 (2000), p. 42–48 (cf. p. 40, 41).

[9] K. Boukhalfa. « De la conception physique aux outils d'administration et de tuning des entrepôts de données ». Thèse de doct. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2009 (cf. p. 18, 24, 29–31, 33–35).

[10] K. Boukhalfa, L. Bellatreche et P. Richard. « Fragmentations Primaire et Dérivée : Etude de Complexité, Algorithmes de Sélection et Validation sous Oracle 10g. » In : *EDA*. 2008, p. 123–139 (cf. p. 46).

[11] R. Brick. « Star schema processing for complex queries ». In : *White paper* (1997) (cf. p. 31).

[12] S. Chaudhuri et V. Narasayya. « Self-tuning database systems : a decade of progress ». In : *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment. 2007, p. 3–14 (cf. p. 19).

[13] R. Elmasri et S. Navathe. *Fundamentals of database systems.*, 2009 (cf. p. 12–14, 16).

[14] A. Kerkad et al. « A query beehive algorithm for data warehouse buffer management and query scheduling ». In : *International Journal of Data Warehousing and Mining (IJDWM)* 10.3 (2014), p. 34–58 (cf. p. 60).

[15] G. Kotonya et I. Sommerville. *Requirements engineering : processes and techniques*. Wiley Publishing, 1998 (cf. p. 12).

[16] M. Laguna, F. J. García-Peñalvo et O López. « Metamodeling for requirements reuse ». In : (2002) (cf. p. 13).

[17] C. Maier et al. « Parinda : an interactive physical designer for postgresql ». In : *Proceedings of the 13th International Conference on Extending Database Technology*. ACM. 2010, p. 701–704 (cf. p. 27).

[18] S. Mitra, S. K. Pal et P. Mitra. « Data mining in soft computing framework : a survey ». In : *IEEE transactions on neural networks* 13.1 (2002), p. 3–14 (cf. p. 39–41).

[19] S. B. Navathe. « Evolution of data modeling for databases ». In : *Communications of the ACM* 35.9 (1992), p. 112–123 (cf. p. 15).

[20] A. P. Pons. « Database tuning and its role in information technology education ». In : *Journal of Information Systems Education* 14.4 (2003), p. 381 (cf. p. 24).

[21] P. Rob et C. Coronel. *Design, Implementation and Management*. 2004 (cf. p. 16).

[22] C. Rolland. « Ingénierie des Besoins : L'Approche L'Ecritoire ». In : *Journal Techniques de l'Ingénieur* (2003), p. 1 (cf. p. 12).

[23] D. Tsichritzis et A. Klug. « The ANSI/X3/SPARC DBMS framework report of the study group on database management systems ». In : *Information systems* 3.3 (1978), p. 173–191 (cf. p. 10).

[24] P. Valduriez. « Join indices ». In : *ACM Transactions on Database Systems (TODS)* 12.2 (1987), p. 218–246 (cf. p. 30).

[25] D. C. Zilio et al. « DB2 design advisor : integrated automatic physical database design ». In : *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment. 2004, p. 1087–1097 (cf. p. 27).