République Algérienne Démocratique Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

**Université d'Ibn Khaldoun – Tiaret**

Faculté des Mathématiques et de l'Informatique

**Département Informatique**

Thème

# Vers une explicitation des composantes d'un Advisor de BD: Cas de sélection des Indexes

Pour l'obtention du diplôme de Master II

**Spécialité :** Génie Informatique

**Option :** système d'information et technologie de web

**Rédigé par :** BEKKAR Ilyes Abdelhamid

HENNI Ahmed Fouzi

**Réalisé par :**

M.OUARED AEK

**Année universitaire** : 2016-2017

# Remerciements

*Avant tout nous remercions Dieu le tout puissant de nous avoir donné le courage et nous avoir guidé pour pouvoir mener à bien ce modeste travail.* A notre maître et président de mémoire Monsieur OUARED Abdelkader. Nous avons eu le privilège de travailler parmi votre équipe et d'apprécier vos qualités et vos valeurs. Votre sérieux, votre compétence et votre sens du devoir nous ont énormément marqués. Veuillez trouver ici l'expression de notre respectueuse considération et notre profonde admiration pour toutes vos qualités informatique.

Ce travail est l'occasion pour nous de vous témoigner notre profonde gratitude. Un grand Merci aux enseignants ainsi que l'administration de la faculté informatique qui a veillé sur notre formation et notre suivi durant tout le cursus d'étude.

Nous remercions également les membres du jury d'avoir accepté d'évaluer notre modeste travail. En fin nous adressons nos remerciements à tous ceux qui ont contribué par leurs conseils ou leurs encouragements à l'aboutissement de ce travail.

*Je dédie ce modeste travail, En Hommage à mon Père, que ton âme repose en paix ! , A ma très chère mère Tu représentes pour moi la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.*
*A mes deux frères Belkacem et Mohamed et ma petite nièce Allaa.*
*A mes deux chères srs Soumia et ma petite nièce et Asma.*
*A toute ma famille.*
*A tous mes amis, particulièrement:*
*Amine, Sara, Rachda, Asma, Badra.*
*A mon binôme.*
*... Atika*
***Amina**.*

*En Hommage à mon Père, que ton âme repose en paix !*
*À tous ceux qui me sont chers :*
*Ma mère,*
*Mes frères.*
*Mes sœurs.*
***Atika**.*

# 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. 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. La maturité de la recherche réalisée dans la conception physique nous motive à aller plus loin et à capitaliser les connaissances et l'expertise en termes de processus, d'algorithmes, de modèles de coûts utilisés pour quantifier le bénéfice des structures d'optimisation sélectionnées, etc. Dans ce mémoire, nous proposons d'abord un langage de conception physique qui permet de décrire toutes les entrées et les sorties de la phase de conception physique. Deuxièmement, 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.

# 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.

# Table des matières

# Table des figures

# Liste des tableaux

# Glossaire

**SP** : Structure physique.
**BNF** : besoin non fonctionnel.
**DBA** : Administrateur de la base de données.
**DB2** : Data base 2.
**SQL** : Structured Query Language.
**SGBD** : Système de gestion de base de donnée.
**BD** : Base de données.
**E/S** : Entrée / Sortie.
**E/A** : Entité / Association.
**BI** : Ingénieur des besoin.
**MCD** : Le modèle conceptuel de données.
**MCT** : Le modèle conceptuel de traitement.
**MLD** : Le modèle logique de de données.
**MPD** : Le modèle physique de données.
**UML** : Langage de modélisation unifié.
**LDD** : Langage de définition des données.
**LDS** : Langage de définition de stockage.
**CP** : Conception physique.
**CBO** : Rule Based Optimizer.
**RBO** : Rule Based Optimizer.
**CPU** : Central Processing Unit
**Ix** : Index
**VM** : vue Matérialisée
**FH** : fragmentation horizontal
**FV** : fragmentation verticale
**TP** : traitement parallèle.
**CL** : Clustring.
**PARINDA** :PARtition and INDex Advisor.
**MdC** : Modèle de cout.

**NP-complet** : Non polynomiale complet.
**MDD** : Modèle de données.
**MS** : Modèle de stockage.
**XML** : Extensible Markup Language.
**XMI** : XML Metadata Interchange.
**DM** : Dispositif matérielle.
**DT** : Dispositif de Traitement.
**HDD** : hard Disk Drive
**SSD** : Solid State Drive
**RAM** : Random Access Memory
**GPU** : General Purpose Computing
**FPGA** : Field Programmable Gate Array
**API** : Application Programming Interface
**VLDB** : Very Large Databases
**SIGMOD**: Special Interest Group on Management of Data
**MDA** : Model driven Architecture
**IDE** : Integrated Development Environment
**EMF** : Eclipse Modeling Framework.

# Introduction générale

## 1 Introduction Générale

### 1.1 Contexte et Motivation

De nos jours, nous assistons au phénomène de déluge de données qui pousse la communauté de la base de données à trouver des solutions pour stocker et gérer de manière efficace des requêtes complexes qui interrogent ces données. Grâce aux structures physiques $\mathcal{SP}$ (telles que les vues matérialisées, les index, le partitionnement des données) sélectionnés lors de la phase de conception de la base de données *conception physique*, des exigences non fonctionnelles (BNF) imposées par l'administrateur de base de données (DBA) (comme la *performance* de la requête, la *fiabilité*, la *facilité* d'utilisation, etc.) peut être satisfait. Le processus de sélection de ces $\mathcal{SP}$s est difficile en raison de l'espace de recherche étendu associé à chaque $\mathcal{SP}$. En conséquence, le processus de sélection d'une $\mathcal{SP}$ donné est formalisé comme un *problème d'optimisation* pour satisfaire une ou plusieurs exigences non fonctionnelles et respecter un ensemble de contraintes telles que le stockage de données nécessaires au $\mathcal{SP}$ sélectionné.

Pour illustrer cette proposition, nous considérons la formalisation la plus traditionnelle du *problème de sélection des index* pour une base de données. Étant donné une *charge de requêtes* et une *base de données*, *recommander* un ensemble d'index ayant le bénéfice maximal (BNF) pour la charge de requêtes. L'*espace de stockage* (*contrainte*) requis pour construire ces index *ne doit pas dépasser* une certaine limite spécifiée par l'utilisateur. Cette sélection est habituellement effectuée par des algorithmes guidés par des modèles de coûts qui quantifient le bénéfice de $\mathcal{SP}$ sélectionné (par exemple, les index). Chaque modèle de coût doit intégrer le maximum d'informations pour donner une *mesure précise*. Ces informations sont généralement liées à la base de données (par exemple : Taille des tables, nombre d'enregistrement, etc.), la charge de requêtes (par exemple : Le nombre de jointures, les facteurs de sélectivité des prédicats), les plates-formes de déploiement (par exemple centralisée, parallèle), Modèle de stockage (par exemple : orienté ligne, orientée colonne). Plusieurs types d'algorithmes sont utilisés dans la conception physique: des algorithmes gourmand, aux algorithmes évolutifs, en passant par des

algorithmes de programmation linéaire. Un algorithme appartenant à chaque type présente certaines caractéristiques telles que *taux de mutation*, *Crossover*. Le choix de ces caractéristiques affecte la qualité de la solution finale.

## 1.2 Problématique

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* (voir la figure 1) pour aider le DBA pendant leurs tâches dans le choix de sélection des $\mathcal{SP}$s.

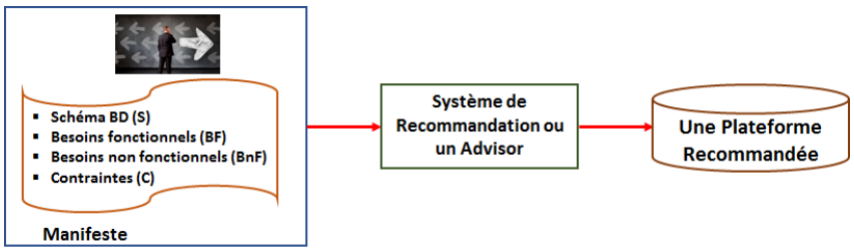


FIGURE 1 – Illustration d'un advisor

D'un point de vue de l'industrie, nous pouvons citer *Index Tuning Wizard*, *Tuning Advisor* dans Microsoft SQL Server [**?**],*PARINDA* pour PostgreSQL [**?**]. Ces advisors peuvent être considérés comme une *boite noire* qui masque tous les éléments de la conception physique, tels que les algorithmes utilisés (et leurs caractéristiques), les modèles de coûts utilisés (leurs informations), etc. Cette situation n'autorise pas les utilisateurs Et le DBA a personnaliser les paramètres de ces outils. Une autre limitation de ces outils est qu'ils ne couvrent que quelques $\mathcal{SP}$. Par exemple, Oracle SQL Tuning Advisor [**?**] recommande trois $\mathcal{SP}$ (vues matérialisées, index et partitionnement), tandis que IBM DB2 *Design Advisor* [**?**] prend en charge quatre $\mathcal{SP}$ (index, vue matérialisée, partitionnement, clustering). L'outil *PARINDA* ne considère que deux $\mathcal{SP}$ qui sont des partitions et des index. Sur la base de ces exemples, nous constatons l'absence d'un *consensus* sur le $\mathcal{SP}$ choisi. Enfin, l'exigence de DBA en termes de sélection de $\mathcal{SP}$ est très limitée, puisque ces advisors supposent que le DBA veut choisir a priori des $\mathcal{SP}$, sans affiner ses besoins en termes Des algorithmes, de leurs paramètres, des modèles de coûts utilisés, etc. Cette exigence est prise en charge en cochant le nom de $\mathcal{SP}$ que le DBA recherche. Avec l'explosion du nombre de $\mathcal{SP}$ et la diversité des modèles de coûts et le types d'algorithmes, les exigences traditionnelles doivent être étendues pour inclure cette diversité. Cela peut être effectué par la présence d'un fichier manifeste définissant tous les besoins d'un DBA.

## 1.3 Objectif

Cette situation nous motive à capitaliser les connaissances et l'expertise de la conception physique en proposant d'utiliser un méta-modèle qui explicite tous les éléments de la conception physique. Ce méta-modèle peut être considéré comme un *Advisors de conception physique*. Les principales étapes de notre approche sont:

- **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).

La figure 2 illustre notre démarche adoptée.

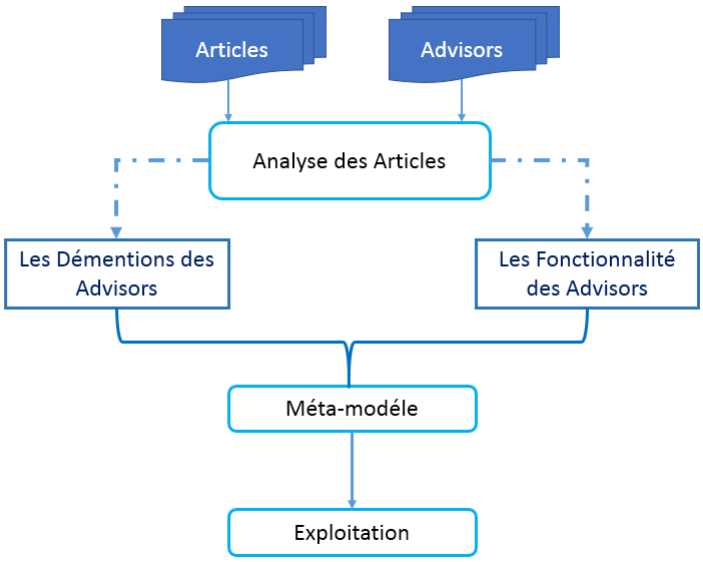


FIGURE 2 – Notre démarche

## 1.4 Plan de mémoire

Ce document est divisé en deux parties principales.

- **La Partie I** présente un état de l'art, elle comporte les deux chapitres suivantes:
  - Le chapitre 1 décrit un état de l'art sur la conception physique de base données.
  - le chapitre 2 présente un état de l'art sur l'administration et tuning des base de données.
- **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.
- Nous présentons dans Le chapitre 5 les aspects liés à la gestion de projet où nous détaillerons les facteurs qui nous ont permis d'atteindre les objectifs voulus.
- Enfin,nous clôturons ce mémoire par une conclusion générale et les perspectives liées à ce projet de fin d'études.

La Figure 3 schématise les grands axes de ce travail et leur répartition dans chaque chapitre.

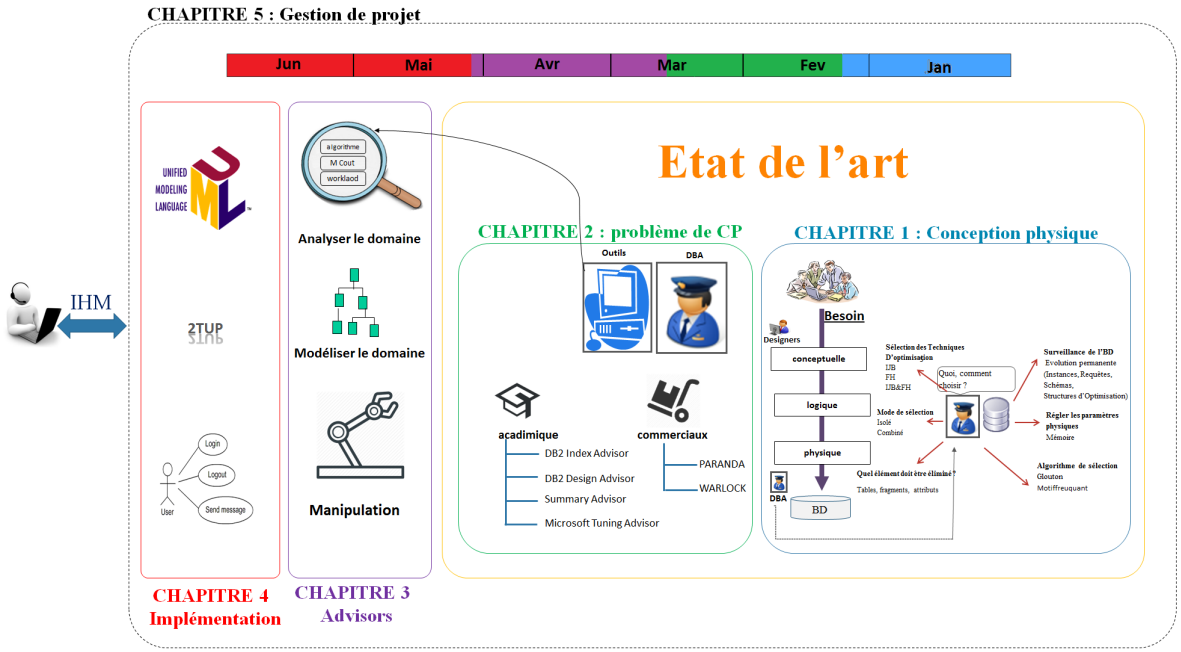


FIGURE 3 – La répartition des chapitres de manuscrit

# Première partie

# État de l’art

Chapitre 1

# Conception Physique des Bases de Données

## Sommaire

# 1 Introduction

Dans ce chapitre, nous présentons un état de l'art 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 Cycle de vie des base 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 [**?**] dans la figure 1

## 2.1 Évolution verticale des modèles de données

L'évolution des modèles de données dite "verticale" (voir la figure 2) a été unifiée dans l'architecture ANSI/SPARC [**?**] qui est actuellement universellement admise dans la communauté des BD.

Cette architecture distingue les trois niveaux d'abstraction : le niveau conceptuel, externe et interne (voir la figure 3):

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,

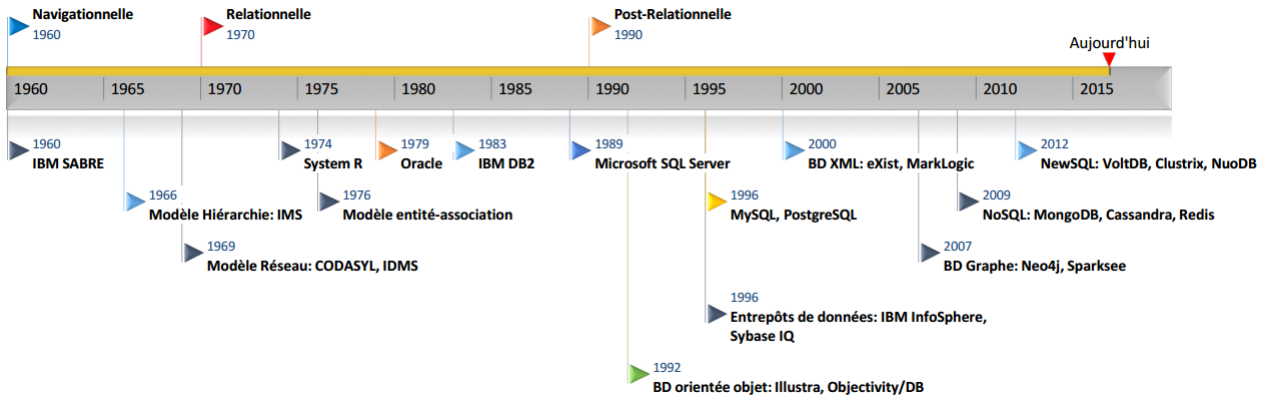


FIGURE 1 – Une brève évolution de la technologie des bases de données.

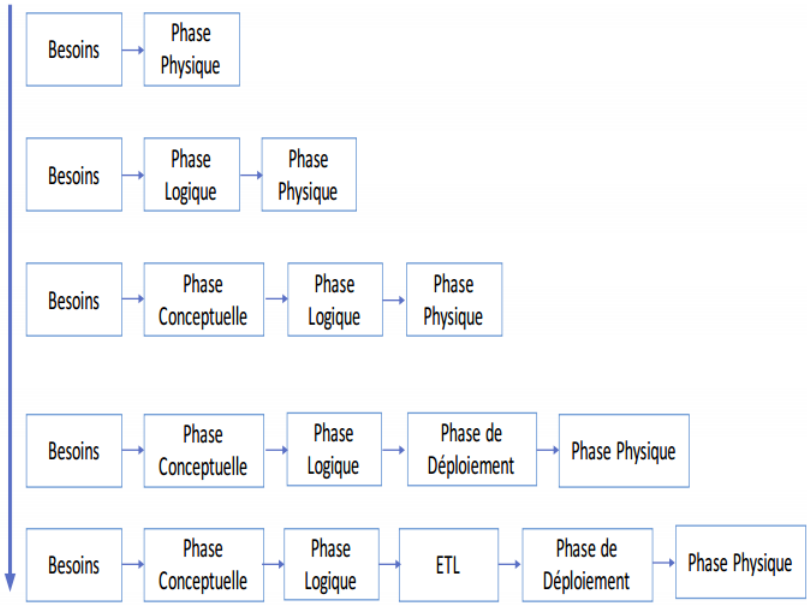


FIGURE 2 – Évolution verticale du cycle de conception des BD

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 4:

- Définition des besoins des utilisateurs,
- Modélisation conceptuelle,
- Modélisation logique,
- Modélisation physique.

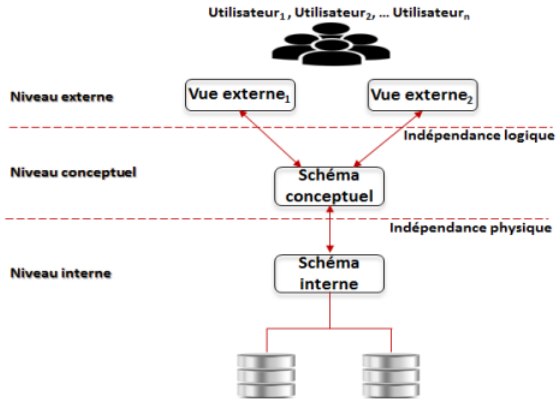


FIGURE 3 – Larchitecture ANSI/SPARC

– Déploiement et maintenance

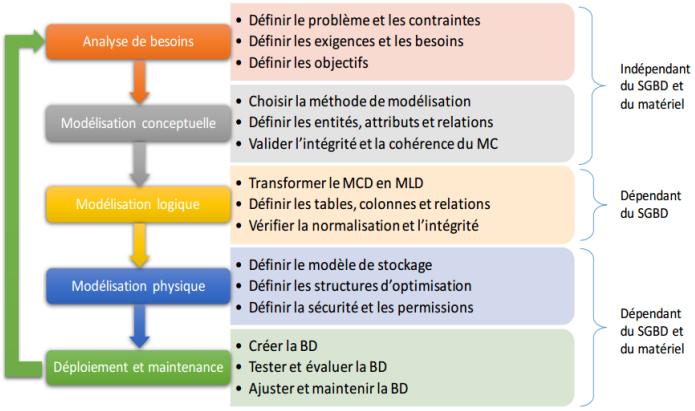


FIGURE 4 – Les phases du cycle de vie et leurs rôles

# 3 Niveau de modélisation

Le domaine relatif à la définition des besoins des utilisateurs est le domaine de lingénierie des besoins (IB). Rolland [**?**] 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 [**?**]. 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 [**?**] :

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 dune manière plus structurée [**?**] : la figure 2 montre l'intégration des besoins dans les différente 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 [**?**].

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.

## 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 [**?**].

- 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 lapplication 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 [**?**]. Les opérations peuvent être de trois différents types [**?**] : 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 [**?**]. 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).

### 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 5.

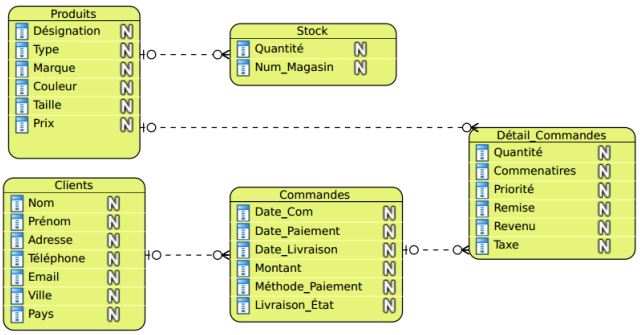


FIGURE 5 – Exemple d'un schéma de modélisation conceptuelle.

## 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 [**?**]. 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.

### 3.2.1 Exemple 2

La Figure 6 représente le schéma logique de notre exemple de gestion de vente. En 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* ».

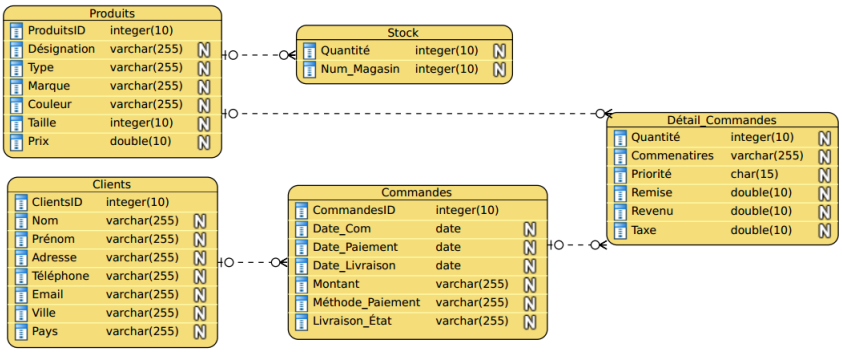


FIGURE 6 – Exemple d'un schéma de modélisation logique.

## 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 [**?**]. 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.

### 3.3.1 Exemple 3

La Figure 7 représente le schéma physique de notre exemple de gestion de vente. En 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* ».

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

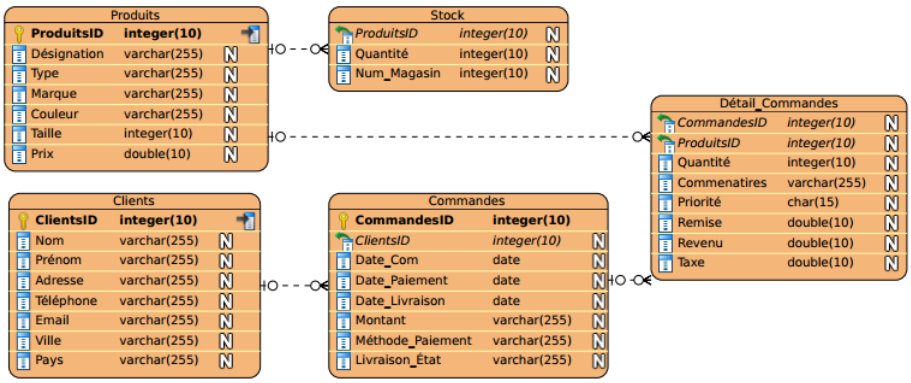


FIGURE 7 – Exemple d'un schéma de modélisation physique

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 [**?**].

## 3.4 La phase de déploiement

Durant cette phase le choix du SGBD et sa plateforme est identifié [**?**]. 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 (voit figure 9) peut se faire dans ce cas lors de la phase de déploiement l'administrateur choisit la plateforme 8 qui répond le mieux aux besoins de l'application. La phase de déploiement est devenue de plus en plus complexe avec l'arrivée des machines parallèles et distribués.

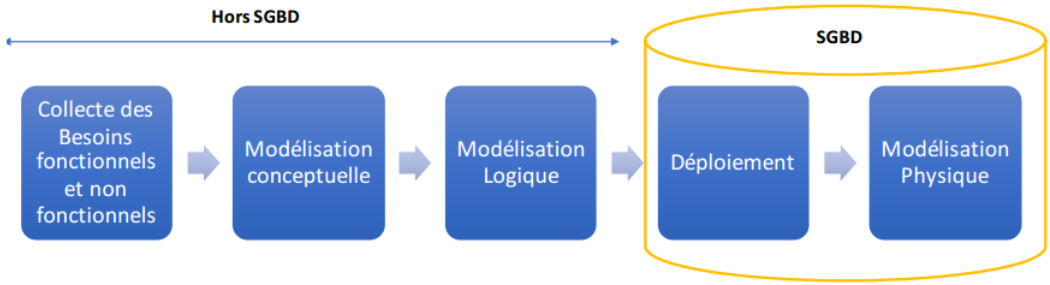


FIGURE 8 – Le choix du SGBD et sa plateforme

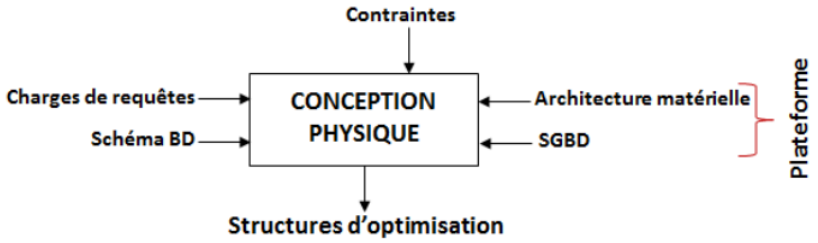


FIGURE 9 – La place de la phase de déploiement dans le cycle de vie

# 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 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 définie la phase de conception physique ensuite le processus d'exécutions d'une raquette âpre nous allons expliqué le rôle de DBA.

## 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 technique est le cur de la conception physique [**?**]. 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.

## 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 uvre de la base de donnée. la figure 10 montre les entrées est les sorties de la *CP*.

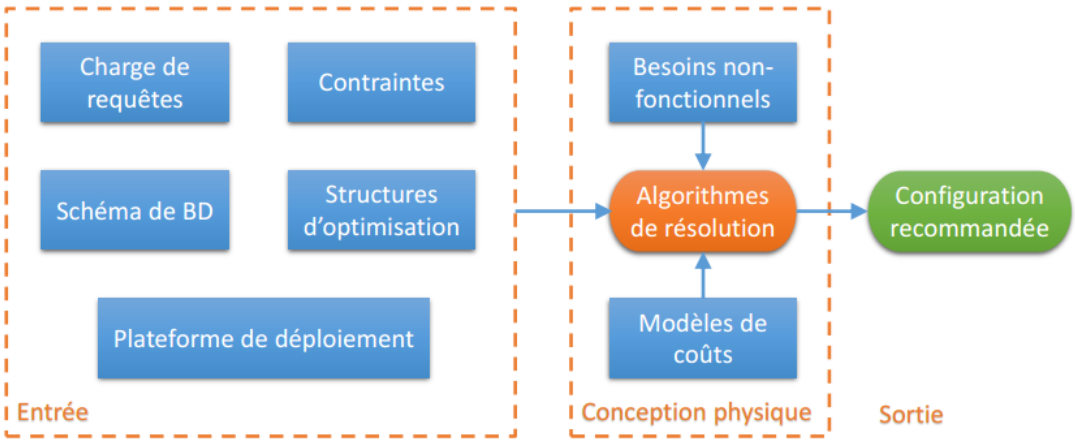


FIGURE 10 – Les entrés et les sorties de la *CP*

## 4.3 Le processus d'exécution des requêees 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 [**?**]. 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.

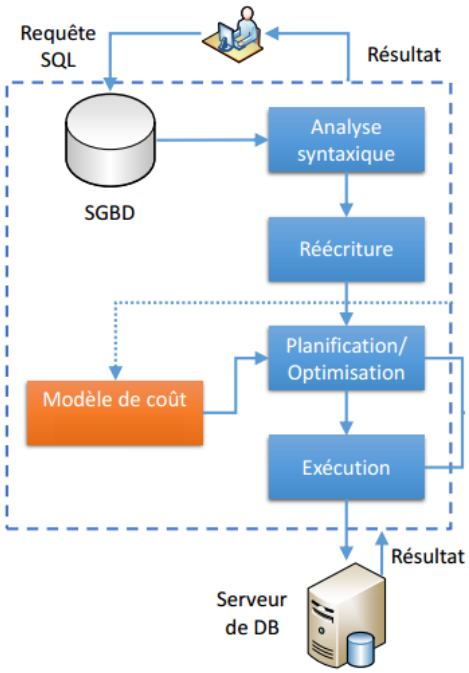


FIGURE 11 – processus d'exécution du requête

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 figure 4.3 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.

### 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 travail 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 nuds 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 12 illustre un exemple d'un d'un arbre d'analyse.

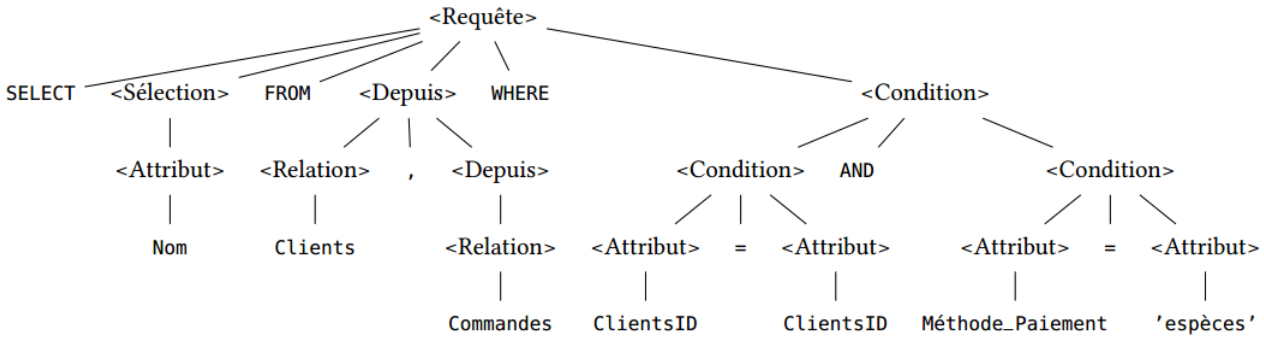


FIGURE 12 – Exemple d'un arbre d'analyse.

### 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.

### 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) (voit figure 13).

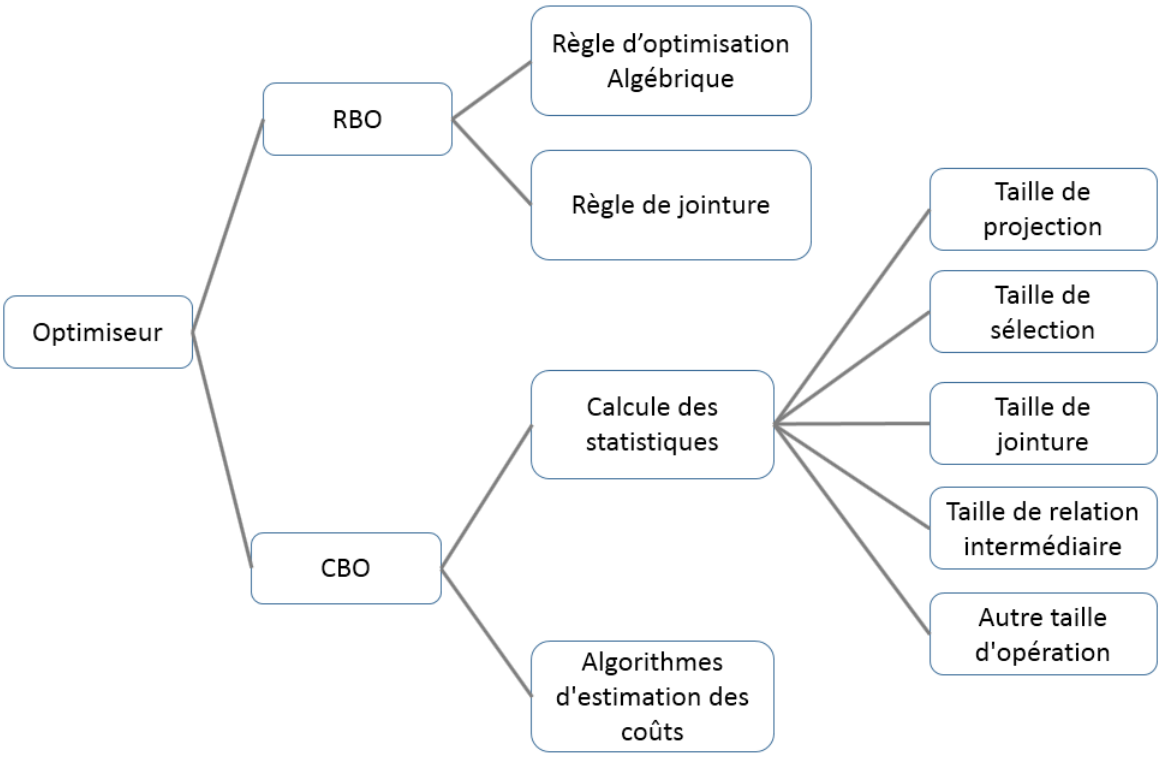


FIGURE 13 – Les deux modes d'optimisation.

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 cur 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.

### 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 dun ou plusieurs fichiers comme entrée, le traité, et ensuite la génération d'un fichier de résultat comme sortie. Si l'opération est mise en uvre 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 14 illustre ce processus.

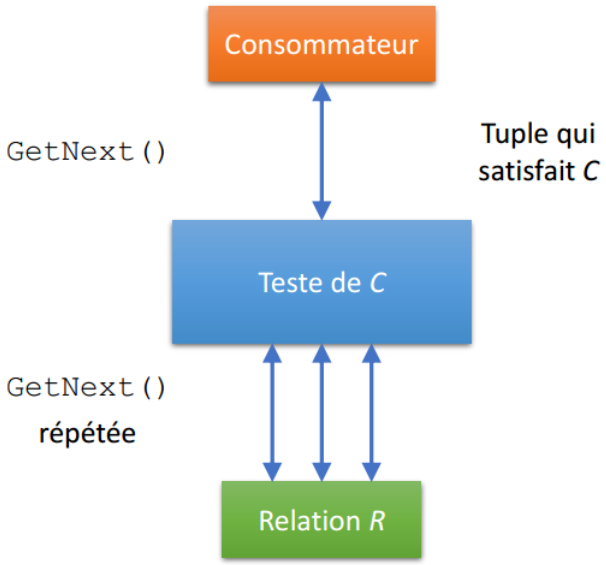


FIGURE 14 – Exécution d'une sélection en mode pipeline à l'aide d'itérateurs.

## 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 doptimisation. 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.[**?**] 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

– Sélectionnée des technique 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 doptimisation a plusieurs algorithmes et approches d'implémentation, Ladministrateur peut sélectionner des algorithme de ces dernier par exemple : pour les Index (Glouton,motif frequant..ect)[**?**]
- 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).

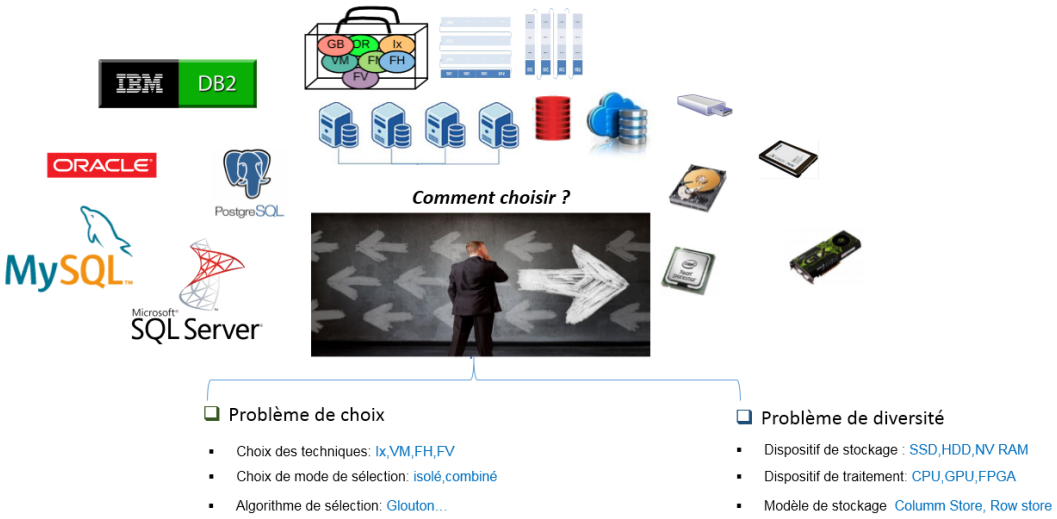


FIGURE 15 – Le problème liés a la conception physique

# 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 2

# Les Advisors : Outils d'assistance à la conception physique

## Sommaire

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

# 1 Introduction

La validation et le déploiement des solutions doptimisation : les recommandations obtenues dun 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 [**?**] et DB2 Design Advisor [**?**] et académiques comme Parinda [**?**] proposent ces services . Cependant, ces outils présentent des limites liées au choix des structures d'optimisation et au mode de sélection fourni.

# 2 Fondements théoriques

La conception physique des bases de données vise à optimiser les performances de la base de données en ajoutant des caractéristiques de conception, telles que des partitions horizontales et verticales, des index ou des vues matérialisées, afin d'accélérer les requêtes dans la charge de travail.La seule façon pour qu'un DBA choisissent des structures de conception physique optimale est de les construire manuellement, puis d'estimer le temps d'exécution de la requête pour les combinaisons des caractéristiques de conception. Cette tâche est à la fois lourde et coûteuse, comme les index, prennent beaucoup de temps et de planification. Par conséquent, l'automatisation de la sélection de la conception physique est cruciale.

## 2.1 Outils d'aide à la conception physique

les outils assistant les administrateurs de BD confrontés à plusieurs décisions complexes à prendre durant les phases de conception physique et tuning. Les premiers travaux dans ce sens traitaient l'élaboration d'outils et d'approches permettant de sélectionner une configuration d'index et depuis l'idée a pris de l'envergure pour inclure les différentes structures d'optimisation.

### 2.1.1 Conception physique et tuning des BD

Le problème de la conception physique consiste à optimiser l'exploitation de la BD et plus particulièrement l'exécution des requêtes par rapport à un ou plusieurs critères non fonctionnels (BNF), en jouant sur plusieurs paramètres de la configuration physique (structures doptimisation : Index (Ix), fragmentation horizontale (FH), fragmentation verticale (FV), vues matérialisées (VM), gestion de buffer (GB), ordonnancement de requêtes (OR), clustering (CL), traite-

ment parallèle (TP), etc., modèle de déploiement, logiciel (application SGBD/noyau système) et matériel (support(s) de stockage, processeur(s)) (voir figure 1)

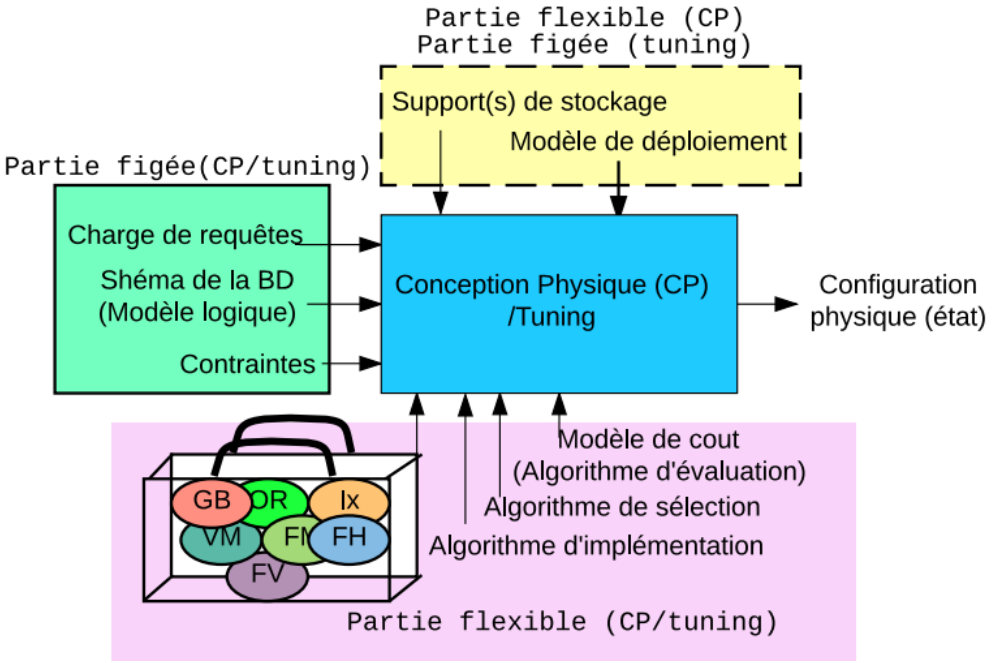


FIGURE 1 – Aperçu du problème de la conception physique et du tuning

### 2.1.2 Comparaison des principaux advisors

Les éditeurs de SGBD commerciaux ont très vite investi dans le tuning, notamment pour les raisons suivantes:

- L'évolution continue des technologies de BD offre plus de choix de tuning à l'administrateur, ce qui complique davantage sa tâche d'administration, surtout en présence dinteraction entre les différentes composantes du SGBD.
- De nouvelles métriques de performance (énergie, sécurité, etc.) se sont imposées dans le marché mouvementé des BD, à côté du traditionnel temps d'exécution (throughput) ce qui, à son tour, a augmenté la complexité du tuning.
- L'administrateur de BD se voit affecter plusieurs tâches aussi compliquées l'une que l'autre, à gérer en même temps ce qui diminue l'efficacité et augmente les coûts de cette administration, contrairement au coût de matériel qui est en baisse constante.

Dans lobjectif daider ladministrateur à faire les bons choix dans un tel contexte, tout en réduisant la charge de travail qui lui incombe, les éditeurs des SGBD se sont tournés vers des solutions dassistance *semi-automatiques* qui proposent des recommandations de tuning concernant un ensemble de structures doptimisation [**?**] [**?**]. Ces solutions sont connues sous le nom **d'Advisor** ou d'outils d'aide et sont généralement propres à un SGBD donné. Une comparaison

entre les outils de certains SGBD leaders du marché est illustrée dans le tableau. L'évaluation des recommandations générées par les advisors est réalisée soit par un modèle de coût mathématique, soit par le modèle de l'optimiseur du SGBD.

TABLE 1 – Comparaison des principaux advisors

| Outil | SGBD | Ix | FH | FV | VM | TP | CL | Modèle de coût |
|---|---|---|---|---|---|---|---|---|
| Oracle Access Advisor | Oracle | X | X | | X | | | Optimiseur |
| Database Tuning Advisor | SQL Server | X | X | X | X | | | Optimiseur |
| DB Advisor | DB2 | X | X | | X | | X | Optimiseur |
| WarLock | - | X | X | | | X | | Mathématique |

### 2.1.3 Catégorie des Advisors

Pour aider l'administrateur dans ses tâches de conception physique et de tuning, des outils d'aide sont nécessaires. Deux architectures peuvent être définies [**?**] pour ces outils : architecture centrée administrateur et architecture centrée outil .

– Dans la première architecture,trois principaux acteurs participent dans la tâche d'administration : le SGBD, l'administrateur et l'outil d'aide. L'administrateur joue un rôle essentiel dans cette architecture. Il constitue un intermédiaire entre l'outil d'aide et le SGBD. L'administrateur reçoit un ensemble d'informations des deux autres acteurs, sélectionne celles qui lui semblent pertinentes et les transmets vers l'autre acteur. voir figure 2

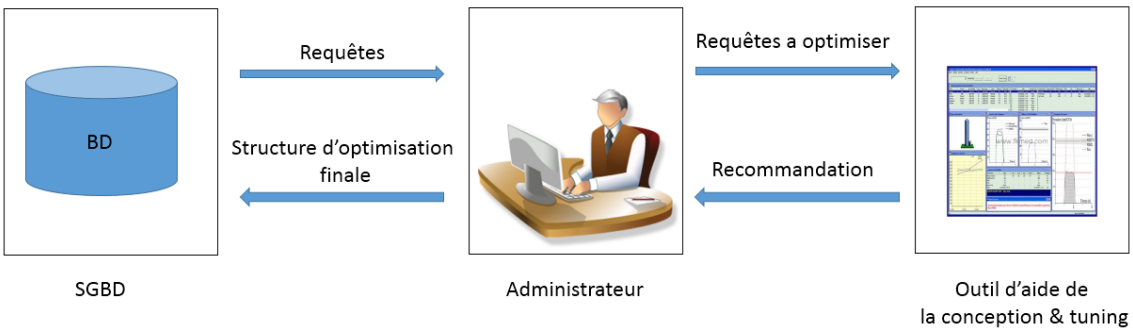


FIGURE 2 – Architecture centrée DBA

– Dans la deuxième architecture, l'administrateur est éloigné du processus d'administration figure 3. Les outils ayant cette architecture entrent dans le cadre de l'auto-administration [**?**]. Ces outils remplacent totalement l'administrateur dans le processus d'administration. A partir de l'ensemble des requêtes fournies par le SGBD, l'outil sélectionne automatiquement un ensemble de techniques d'optimisation optimisant la performance des requêtes.

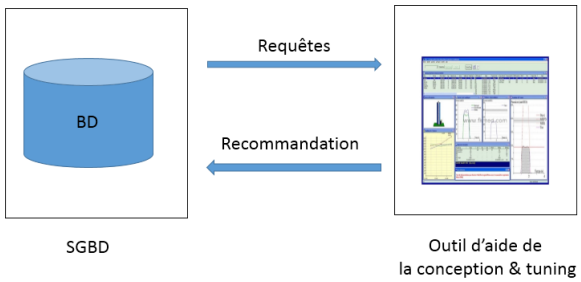


FIGURE 3 – Architecture centrée outils

## 2.2 Solutions académique et industrielle

L'intérêt de la phase de conception physique a motivé les communautés académique et industrielle à développer des outils (connues sous le nom Advisors) assistant les administrateurs de BD dans leurs tâches en leur recommandant des structures d'optimisation (vues matérialisées (VM), index (Ix), fragmentation horizontale (FH), fragmentation verticale (FV), traitement parallèle (TP), clustering (CL)). Nous pouvons citer les advisors proposés par les éditeurs de SGBD traditionnels : Oracle SQL Access Advisor [**?**], Data Tuning Advisor de Microsoft SQL Server [**?**], Design Advisor de DB2 dIBM [**?**]. D'autres advisors académiques ont été proposés comme Parinda [**?**] et SimulPhD [**?**].

## 2.3 PARINDA outil d'aide a la conception

Dans l'objectif d'aider l'administrateur à faire les bons choix dans un tel contexte, tout en réduisant la charge de travail qui lui incombe, les éditeurs des SGBD se sont tournés vers des solutions d'assistance semi-automatiques qui proposent des recommandations de tuning concernant un ensemble de structures doptimisation [**?**]

les auteurs ont devloppé un outil PARINDA(PARtition and INDex Advisor) [**?**]qui est une solution d'un SGBD open source *(PostgreSQL)*.

PARINDA ne permet pas de supprimer l'espace candidat avidement. Par conséquent, il recherche à travers toutes les fonctionnalités candidates utiles avant de proposer l'ensemble optimal de fonctionnalités. Il permet au DBA d'estimer de manière interactive le bénéfice des nouvelles caractéristiques physiques de conception en simulant efficacement les caractéristiques de conception. Enfin, il réécrit automatiquement les requêtes pour bénéficier pleinement des fonctionnalités de conception proposées. la figure 2.3 montre l'architecture de cette outil

- Les composants *what-if* sont utilisés pour simuler des partitions physiques, des index et la présence ou l'absence de méthodes de jointure. Il existe trois composants utilisant les composants *what-if* : le composant d'indexation automatique, le composant de partitionnement automatique et le composant de partitionnement/indexation interactif.
- Le composant de partitionnement automatique prend comme entrée la charge de travail de

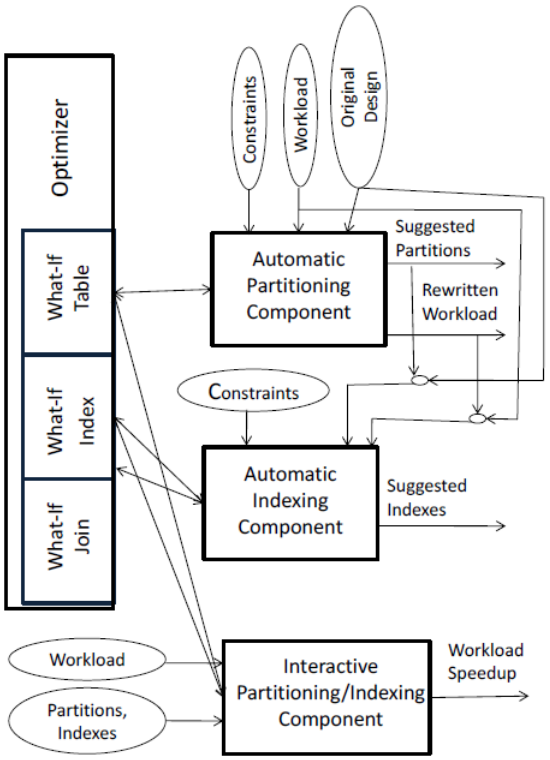


FIGURE 4 – Architecture de PARINDA

requête, la conception physique d'origine et plusieurs contraintes définies par DBA telles que l'espace maximum pris par les colonnes répliquées dans les partitions. La sortie est composée des nouvelles partitions qui améliorent de manière optimale le temps d'exécution de la charge de travail et des nouvelles requêtes réécrites reflétant les nouvelles partitions.

– Le composant d'index automatique a comme entrée la charge de travail, la conception physique et les contraintes de taille. La sortie représente l'ensemble des index suggérés.
– L'entrée du composant de partitionnement/indexage interactif est donnée comme la charge de travail de la requête et la conception originale. Il produit un nouveau design et estime également l'avantage d'utiliser le nouveau design.

## 2.4 Les structures physique

Plusieurs techniques d'optimisation ont été proposées dans la littérature et supportées par les SGBD commerciaux.L-Bellatreche et al, ont classer en deux catégories principales [**?**] : techniques redondantes et techniques non redondantes. Les techniques redondantes optimisent les requêtes, mais exigent un coût de stockage et de maintenance. Cette catégorie regroupe les vues matérialisées [**?**], les index [**?**], la fragmentation verticale [**?**],etc. Les techniques non redondantes ne nécessitent ni coût de stockage ni coût de maintenance.Cette catégorie regroupe

la fragmentation horizontale [**?**], le traitement parallèle [**?**],Cluster

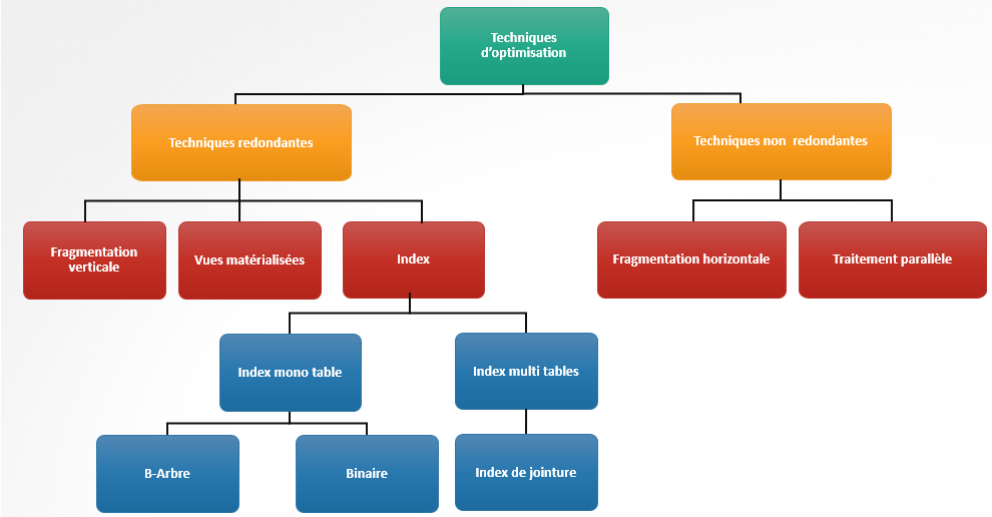


FIGURE 5 – Les structures d'optimisation

### 2.4.1 les techniques d'optimisation redondantes

Nous présentons dans qui suit deux techniques d'optimisation redondantes,l'indextation, les vue matérialisée et La fragmentation verticale .

1. **Index**: L'indexation est l'une des techniques d'optimisation redondantes qui minimisent le volume de données à exploiter dans les calculs. La création d'un index permet d'améliorer considérablement le temps d'accès aux données en créant des chemins d'accès directs. Deux types d'index sont disponibles : les mono-index (B-tree, index binaire, projection, etc.) et les multi-index (index de jointure). Les index mono-table sont des index définis sur un ou plusieurs attributs de la même table, et les index multi-tables sont des index définis sur plusieurs tables. Nous présentons dans les sections suivantes les principales techniques d'indexation utilisées dans les SGBD relationnels et les bases de données.
   - **Techniques d'indexation**: Plusieurs techniques d'indexation peuvent implémenter sur les SGBD commerciaux dont les plus connues sont : les *index B-arbre*, les *index binaires*, les *index de jointure*, etc. Un administrateur de base de données a besoin d'information sur la façon dont les données sont stockées pour optimiser l'exécution des requêtes.
   - **Les index simples**: Regroupent les index B-arbres et les index de projection. L'index B-arbre est défini sur un attribut d'une table. Les nuds de l'arbre renferment les valeurs ordonnées de l'attribut et les feuilles contiennent des clés vers les tuples de données stockées sur le disque.

- ***L'index B-arbre*** est l'index par défaut pour la plupart des SGBD commerciaux
- *Les 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. 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. La figure qui suit 6 montre un index de projection défini sur l'attribut Ville de la table Client.

| Table client | | | | |
|---|---|---|---|---|
| ID | Nom | Age | Sexe | Ville |
| 616 | ILYES | 23 | H | TIARET |
| 515 | KHALED | 30 | H | ALGER |
| 414 | FATIMA | 24 | F | ORAN |
| 313 | AHMED | 33 | H | ORAN |
| 212 | FOUZI | 26 | H | TIARET |
| 111 | SALMA | 29 | F | TIARET |

| Index de projection sur l'attribut Ville |
|---|
| Ville |
| TIARET |
| ALGER |
| ORAN |
| ORAN |
| TIARET |
| TIARET |

FIGURE 6 – Index de projection

- **Les index bitmap**: Les index Bitmap (IB) stockent les données de chaque dimension (ou d'attribut) séparément, et permettent l'accès rapide aux différentes dimensions qui sont nécessaires pour répondre à une requête. Par exemple, un index bitmap simple (IBS) sur un attribut Sexe, avec le domaine Homme, Femme, représenté sous forme de deux vecteurs bitmap noté 'H','F'. Pour H, le bit est un ensemble de 1 si le correspondant de tuple à la valeur « Homme » pour l'attribut Sexe, sinon le bit est un ensemble de 0. De même pour le F, le bit est un ensemble de 1 si le tuple associé à la valeur « Femme » pour l'attribut Sexe, sinon le bit est un ensemble de 0, Pour un autre attribut, par exemple Ville ayant des tuples (TIARET,ORAN,ALGER), il y a un vecteur bitmap correspondant à chacune des trois valeurs, noté B1-B3. Tuples associés à la valeur « 1 » pour l'attribut Ville sont représentés par des vecteurs binaires « B1=1, B2=0, B3=0 » la figure qui suit 7 montre ce exemple.
- **Les 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 [**?**] qui matérialise les liens existant entre deux tables en utilisant une table à deux colonnes chacune représentant l'identifiant dune table. Voir Figure 8 . 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.
- **Index de jointure en étoile**: L'index de jointure en étoile est adapté aux requêtes définies sur les bases de données modélisées en étoile. Il permet de stocker le résultat de la jointure entre les tables . Il est dit complet s'il regroupe toutes les tables de dimen-

| Table client | | | | |
|---|---|---|---|---|
| ID | Nom | Age | Sexe | Ville |
| 616 | ILYES | 23 | H | TIARET |
| 515 | KHALED | 30 | H | ALGER |
| 414 | FATIMA | 24 | F | ORAN |
| 313 | AHMED | 33 | H | ORAN |
| 212 | FOUZI | 26 | H | TIARET |
| 111 | SALMA | 29 | F | TIARET |

| Index binaire sur l'attribut Ville | | |
|---|---|---|
| TIARET | ORAN | ALGER |
| 1 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |

FIGURE 7 – Index Bitmap

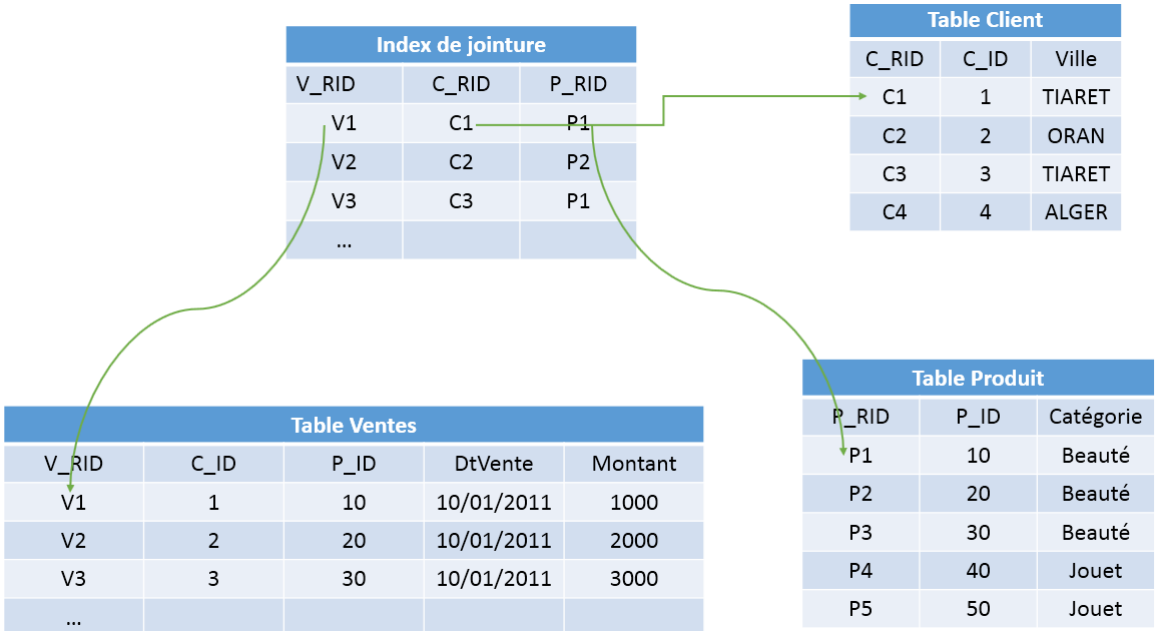


FIGURE 8 – Index de jointure

sion sinon partiel. Il accélère l'opération de jointure mais exige beaucoup d'espace de stockage et un coût de maintenance très élevé.

– **Index de jointure binaire *(IJB)***: Dans les bases de données les requêtes impliquent des opérations de jointure, construire un index de jointure sur les colonnes de jointure améliore le temps de d'exécution de la requête. L'index de jointure binaire (IJB) est construit en créant n'importe quel type d'index Bitmap sur une table T basé sur une colonne A de la table S, où A est généralement l'attribut joindre. Il est habituellement utilisé avec de données de cardinalité réduite. Supposons, un attribut A ayant n valeurs distinctes, et appartenant à une table T1, et une table T2 composée de m instances, la construction dun index de jointure binaire IJB défini sur l'attribut A se fait de la manière suivante : 9 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.
Pour une base de données modélisée par un schéma relationnel, les index de jointure binaires sont souvent utilisés pour accélérer les requêtes de jointure en étoile connues par leur nombre d'opérations de jointure. L'IJB est défini sur un ou plusieurs attributs appartenant à plusieurs tables.

**Table Client**

| C_RID | C_ID | Nom | Ville |
|---|---|---|---|
| 6 | 616 | ILYES | Tiaret |
| 5 | 515 | KHALED | Alger |
| 4 | 414 | FATIMA | Oran |
| 3 | 313 | AHMED | Oran |
| 2 | 212 | FOUZI | Tiaret |
| 1 | 111 | SALMA | Tiaret |

**Table Ventes**

| V_RID | V_ID | P_ID | T_ID | MONT |
|---|---|---|---|---|
| 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 | 14 |
| 7 | 111 | 101 | 44 | 200 |
| 8 | 111 | 101 | 33 | 270 |
| 9 | 212 | 101 | 11 | 1000 |
| 10 | 313 | 102 | 11 | 2000 |
| 11 | 414 | 102 | 11 | 102 |
| 12 | 414 | 102 | 55 | 103 |

**Index de jointure binaire**

| RID | Tiaret | Oran | Alger |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 |
| 7 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 |
| 10 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 |
| 12 | 0 | 1 | 0 |

FIGURE 9 – Index binaire de jointure

2. **Vue matérialisée**: Une vue est une requête nommée. Elle est dite matérialisée si son résultat est stockée 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. Les vues dans le contexte OLTP ont été largement utilisées pour répondre à plusieurs rôles : la sécurité, la confidentialité, l'intégrité référentielle, etc. 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. Les répliquas permettent de résoudre des requêtes uniquement par des accès locaux. Deux problèmes majeurs sont liés aux vues matérialisées : (1) le problème de sélection des vues matérialisées et (2) le problème de maintenance des vues matérialisées.
3. **Partitionnement**:Dans la littérature, deux termes sont utilisée pour la segmentation d'une relation : partitionnement et fragmentation. Le partitionnement d'une relation se définit par la division de cette dernière en plusieurs partitions disjointes. La fragmentation consiste en la division en plusieurs fragments (sous-ensembles de la relation) qui peuvent être non disjoints. Cependant, la plupart des chercheurs utilisent les deux termes indifféremment.

1 **La fragmentation verticale**:La fragmentation verticale est considérée comme une technique d'optimisation redondante Elle permet de décomposer une table en plusieurs sous ensembles disjoints appelés fragments verticale,résultant de l'application de l'opération de projection chacun contient un sous ensemble d'attribut. Elle favorise naturellement le traitement des requêtes de projection portant sur les attributs utilisés dans le processus de la fragmentation, en limitant le nombre de fragments auxquels accéder. Mais elle requiert des jointures supplémentaires lorsqu'une requête accède à plusieurs fragments.

### 2.4.2 les techniques d'optimisation non-redondantes

Les techniques d'optimisation non-redondantes n'impliquent pas une surcharge de stockage et de maintenance. 2 **fragmentation horizontal** La fragmentation horizontale constitue un aspect important dans la conception physique des bases de données [**?**] [**?**]. Elle est considérée comme une technique d'optimisation non redondante

La fragmentation horizontale (FH) est une technique d'optimisation considérant comme l'une des structures d'optimisation dans le cadre des base de données relationnels. Elle permet de décomposer une table en plusieurs sous ensembles disjoints appelés fragments horizontaux, chacun contient un sous ensemble de tuples. On distingue deux types de FH :

(1) FH primaire définie sur une table de dimension en fonction de ses propres attributs, et (2) FH dérivée définie sur la table des faits en fonction des dimensions fragmentées. La fragmentation dérivée est adaptée au contexte des bases de données relationnels.

## 2.5 Le Problème de sélection des Structures Physique

Cette tâche est souvent liée à ladministrateur. Durant la phase de conception physique, ladministrateur doit effectuer quatre tâches principales : (1) le choix des structures doptimisation, (2) le choix de leur mode de sélection, (3) le développement des algorithmes de sélection et (4) la validation et le déploiement des solutions doptimisation [**?**].

- **Choix des structures d'optimisation** : il existe une large panoplie de structures d'optimisation pour la conception physique. L'identification des structures pertinentes exige un haut niveau d'expertise, car elle dépend de l'étude de la charge à optimiser et de la plateforme sur laquelle celle-ci s'exécute. Nous pouvons citer la fragmentation, les vues matérialisées, les index, la compression, factorisation, réplication, etc.
- **choix du mode de sélection** : l'optimisation peut être effectuée en utilisant une ou plusieurs structures d'optimisation. Dans le premier cas, il s'agit d'une sélection isolée. Dans le deuxième, il sagit dune sélection multiple. Dans ce cas, plusieurs scenarii sont possibles pour combiner ces techniques [**?**]. Le choix de l'ensemble des structures à employer et du mode de combinaison est déterminant en matière de performance du système.
- **Développement des algorithmes de sélection** : les algorithmes proposés pour chaque

problème doptimisation subissent une certaine évolution, en passant dalgorithmes simples vers des algorithmes lourds. Les approches simples sont souvent faciles à mettre en uvre, mais donnent une faible efficacité (comme les approches de la gestion du buffer [**?**]). Les algorithmes lourds donnent une meilleur efficacité mais avec un temps. doptimisation élevé (comme les approches de la fragmentation horizontale [**?**]). Les compromis s'avèrent très rares dans le problème de la conception physique.

– **La validation et le déploiement des solutions doptimisation** : les recommandations obtenues dun 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 [**?**] et DB2 Design Advisor [**?**] et académiques comme Parinda [**?**] proposent ces services . Cependant, ces outils présentent des limites liées au choix des structures d'optimisation et au mode de sélection fourni.

Le problème consiste à sélectionner un ou plusieurs structures d'optimisation qui satisfont l'ensemble de besoins non fonctionnels et de satisfaire les contraintes définies dans C. Le problème de conception physique est connu comme NP-complet[**?**], en raison de la taille de son espace de recherche.

Plusieurs algorithmes de résolution de ce problème ont été proposés. Pour quantifier leur qualité, des modèles de coût mathématique estimant les besoins non fonctionnels ont été également définis [**?**]. Ces modèles de coût prennent en compte les paramètres pertinents liés au contexte de la BD, son SGBD et sa plateforme, la politique dexécution de requêtes (centralisée/parallèle) [**?**], etc. Souvent tout modèle de coût est développé en fonction des composantes principales dun SGBD [**?**] d'où sa décomposition en trois parties : (1) le coût des entrées sorties (E/S) pour lire et écrire entre la mémoire et le support de stockage comme les disques, (2) le coût CPU et (3) le coût de communication sur le réseau COM (si les données sont réparties sur le réseau). Ce dernier est généralement exprimé en fonction de la quantité totale des données transmises. Il dépend de la nature de la plateforme de déploiement.

# 3 Diversité dans la conception physique

La diversité implique le besoin de faire adapter les systèmes à un contexte particulier. Les prouesses technologiques et la vulgarisation de la numérisation ont fait exploser la quantité et les types des exigences/applications ce qui, à son tour, a multiplié les méthodes, modèles, outils et paradigmes utilisés dans la conception, et a attisé notre curiosité vis-à-vis de la gestion de cette diversité des choix auxquels les concepteurs sont confrontés. En raison de la diversité des structures d'optimisation, nous considérons les dimensions de modèle de stockages, l'architecture de déploiement et les paramètres matérielles. En effet, comme l'étude de contexte a montré, le besoin de gérer la diversité et la variabilité croissante au sein de la conception des BD devient un enjeu important. Ces bases s'avèrent de plus en plus insuffisantes face à la croissante complexité et diversité du processus de conception

## 3.1 Le modèle de données

Le modèle de données ($\mathcal{MDD}$) est le plan pour construire une base de données [**?**]. Il se compose de trois éléments : (i) la collection de structures de données, (ii) la collection d'opérateurs qui définissent les manipulations possibles sur ces structures pour extraire et pour calculer les données, et (iii) la collection de règles d'intégrité. Compte tenu des différents besoins en matière de modélisation (Relationnel, Objet, XML etc.) et de traitement des données dans les applications. Une classification des systèmes des bases de données est donnée dans la Figure 10.

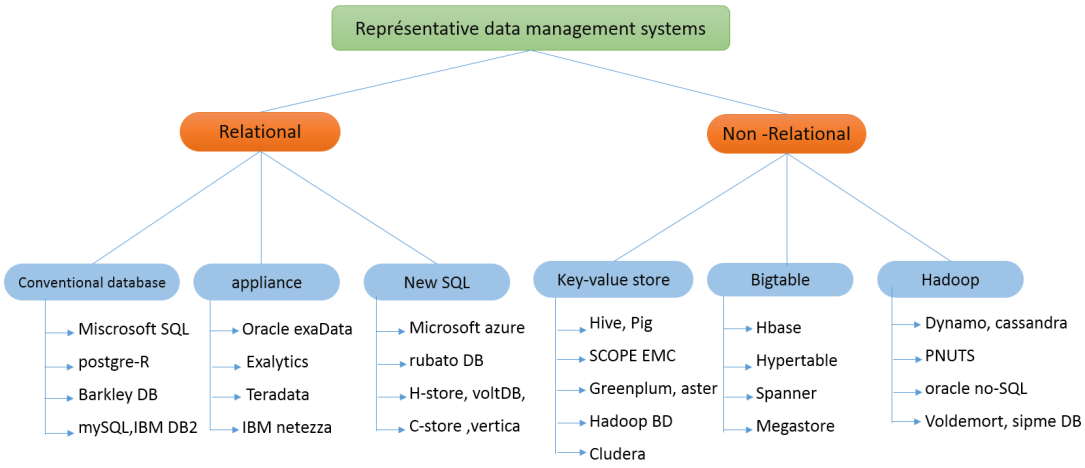


FIGURE 10 – Taxonomie du système de gestion de donnée [**?**].

Afin de mieux comprendre cette nouvelle Ère de multiplicité de solutions Systèmes de Gestion de Bases de Données (SGBD) hébergeant les BD. Pour ce faire, nous avons mené une enquête basée sur le classement du mois d'octobre 2016 livré par le site spécialisé DB-Engines [1], où 316 candidats étaient en lice pour le titre du SGBD le plus populaire.

## 3.2 Le modèle de stockage

Le modèle de stockage ($\mathcal{MS}$) utilisé à un impact important sur les performances de la requête de base de données, car ils ont une grande influence sur le nombre de pages qui doit être lu à partir du disque pour répondre une requête, il est possible de stocker des données dans le support de stockage par deux méthodes : *Row Store* et *Column Store*. Ainsi, une combinaison des deux modèles de stockage est possible, comme indique le travail de [**?**]. Ils affirment qu'il est possible de stocker des données de manière redondante dans un système de stockage hybride (des tables orientée colonne et des tables orientée ligne).

1. http://db-engines.com/en/ranking, publie un classement mensuel des SGBD depuis 2012

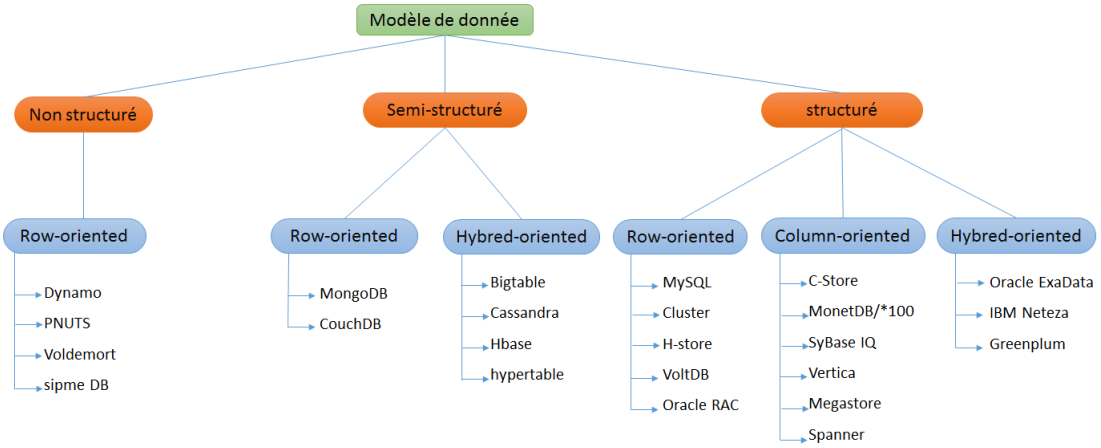


FIGURE 11 – Taxonomie du modèle de données [**?**].

## 3.3 Taille de page système

Une autre propriété importante des bases de données est le format de la taille de la page système ($\mathcal{PS}$). La base de données est divisée en pages ESPS (*Entrée/Sortie par second*) nécessaire pour stocker une table ou un résultat intermédiaire. Les données d'un système de base de données, tels que les colonnes ou les lignes des tables ainsi que les structures (index et VM), sont stockées sur des pages qui sont synchronisés avec le stockage persistant. Les petites pages peuvent entraîner de nombreuses lectures, mais offre un accès plus granulaire et sélective que les grandes pages. Par conséquent, la quantité d'espace de stockage par page est un facteur d'impact important sur la performance des opérations de base de données. Comme la taille de la page peut prendre une valeur numérique arbitraire qu'il peut prendre plusieurs valeurs.

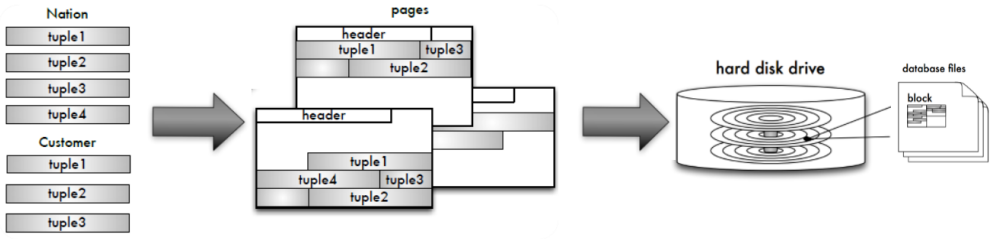


FIGURE 12 – Les tuples manipulés par des requêtes sont stockés dans des pages

## 3.4 Architecture de déploiement

Le déploiement de base de données ($\mathcal{A}r_D$) est fait selon différentes politiques, centralisée, distribuée, parallèle ou architecture Cloud. L'architecture centralisée est caractérisée par le fait que la base de données est stockée dans un seul support de stockage. Dans l'architecture distribuée les données sont stockées dans plusieurs supports de stockage dans divers sites connecté

par un réseau informatiques. Lorsque les données sont identifiées dans différents sites de l'architecture distribuée on parle de l'architecture répliquée. L'architecture parallèle possède les mêmes caractéristiques de l'architecture distribuée mais chaque support de stockage dans le même site à plusieurs CPU.

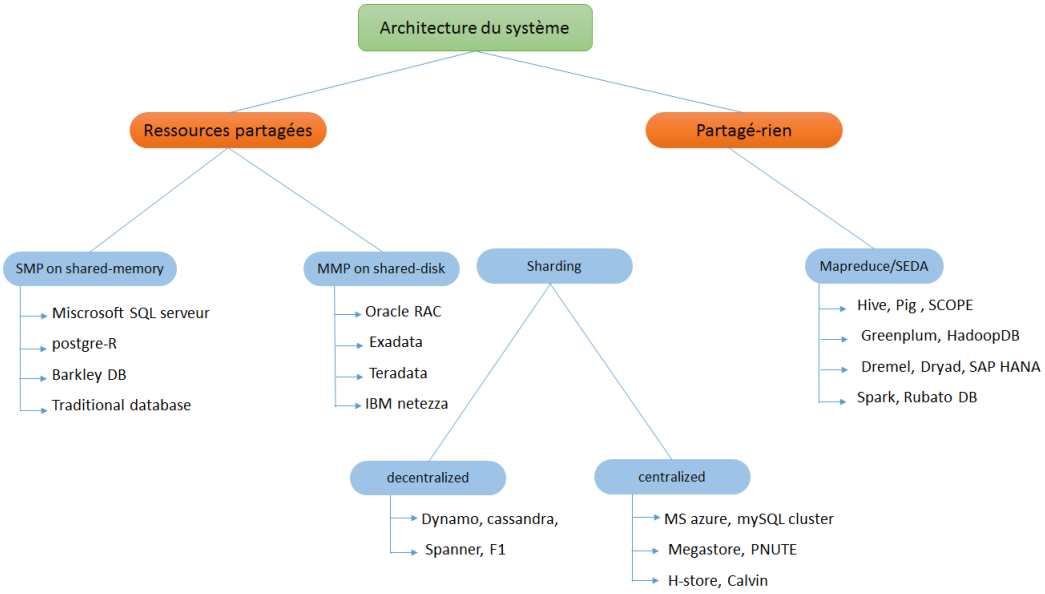


FIGURE 13 – La taxonomie de l'architecture du système.

## 3.5 Dispositif matériel

Décrire comment les dispositif matériels $\mathcal{DM}$ <stockage, traitement>:Un autre groupe de facteur d'impact qui influe sur la performance des opérations de base de données sont des paramètres matériels. Surtout changer la capacité et le type de matériel de traitement influence la conception d'algorithmes.

## 3.6 Dispositif de stockage

$\mathcal{DS}$: considérons des disques durs (Hard Disc Drive HDD), disque flash Solide State Drives (SSD), et la RAM en tant que dispositifs de stockage. Les scénarios de stockage possible sont: $\{\{HDD \vee SSD \wedge RAM\},\{HDD \wedge RAM \wedge SSD\}\ ,\{RAM\}\ \}$

## 3.7 Dispositifs de traitement

$\mathcal{DT}$:En tant que dispositifs de traitement, on classe les composants matériels qui traitent des données en vue d'effectuer une opération de base de données. Il est possible de choisir plusieurs

dispositifs de traitement comme CPU, GPU et FPGA.

## 3.8 Gestion de Buffer

Gestion de Buffer $\mathcal{GB}$ est un des facteurs spécifiques aux BDs est la gestion de buffer. Le but principal de la gestion du buffer est de minimiser les E/S physiques car les accès à une page sur la mémoire secondaire (HDD, Flash, Cloud ...) sont beaucoup plus couteux que sur le buffer. La $\mathcal{GB}$ est optionnelle, car il y a des configurations qui ne nécessitent pas la gestion de buffer, par exemple le cas dune base de données en mémoire fonctionne sans gestionnaire de buffer, puisque toutes les données sont déjà dans la RAM. Chaque gestionnaire de buffer a une taille fixe généralement définie par le nombre de pages qui est variée d'un système à un autre. Il existe des politiques d'allocation et de remplacement de pages introduites dans la littérature comme LRU (Least Recenty Used), FIFO (Firts In First Out) et Random (algorithme de remplacement de pages aléatoires). La gestion du buffer nécessiste l'adaptation des politiques de gestion du buffer aux nouvelles caractéristiques physiques de l'environnement de base de données [**?**].

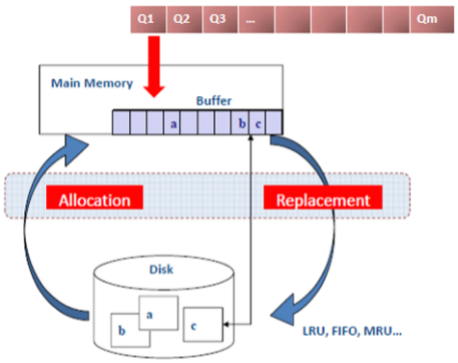


FIGURE 14 – Technique d'évaluation

## 3.9 La diversité des besoins non fonctionnels

Un autre changement qui est venu déstabiliser le cycle standard, est la diversité des besoins non fonctionnels. En effet, la performance (l'efficacité) des BD se mesurait essentiellement en matière de temps d'exécution des requêtes. Face aux différentes évolutions précédemment évoquées et les nouvelles exigences du marché, d'autres mesures se sont imposées, à l'instar de lénergie et de la sécurité. De ce fait, non seulement certaines phases de conception doivent sadapter, comme loptimisation logique et physique [**?**], mais aussi un autre aspect vient rendre le processus encore plus compliqué, celui de l'interaction entre ces mesures et la nature multi-objectifs du problème. Ces mesures sont plus communément connues sous le nom de besoins non fonctionnels (BnF). Ainsi, la conception de BD s'apparente davantage à un problème de satisfaction des besoins non fonctionnels.

# 4 Conclusion

Ce chapitre représente le cur de nos discussions et nos réflexions sur la nécessité de considérer le problème de sélection de plateforme pour une application donnée. La résolution de ce problème assistera le déployeur. Vu la complexité du problème de déploiement d'une BD, nous avons donné une instance de ce problème. Dans ce dernier, nous supposons la présence d'un déployeur avec son manifeste incluant les éléments suivants : (a) le schéma de BD, (b) les besoins fonctionnels, (c) les besoins non fonctionnels et (d) ses contraintes. Pour résoudre ce problème, nous supposons que ses éléments sont similaires à ceux proposés par les bancs d'essai. Cette hypothèse nous a amenée à présenter les différents bancs d'essai utilisés par la communauté

# Deuxième partie

# Notre Contribution

Chapitre 3

# Présentation de notre approche

*Passion is energy. Feel the power that comes from focusing on what excites you.*
*Oprah Winfrey*

## Sommaire



## 3.1 Introduction

Comme nous avons vu aux chapitres de l'état de l'art, 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.

## 3.2 Formalisation de problème de sélection d'index

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 donné 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* [**?**] 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. Ces dimensions permettent de classer les advisors existant dans l'état de l'art (voir figure 3.1).

Ces dimensions permettent de classer les solutions existant dans l'état de l'art. La Figure 3.2 illustre notre cube.

## 3.3 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 uvre du domaine comme illustre dans la figure suivant : 3.3

- **Analyse de domaine**: Pour identifier les principales dimensions d'un advisors.
- **Modélisation de domaine**: Pour lier les dimensions identifiés entre eux.


FIGURE 3.1 – Classification des travaux des advisors pour les Index (Ix)

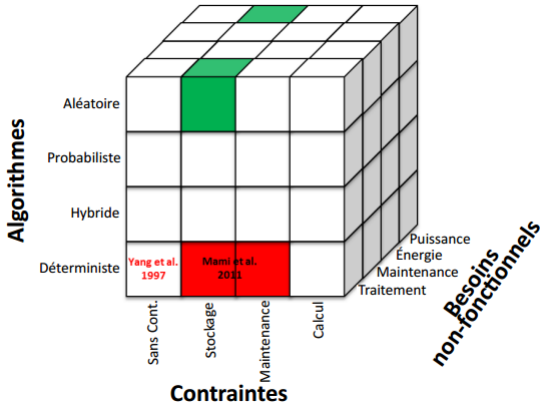


FIGURE 3.2 – Méthodes de résolutions du PCP

– **Usage**: Pour Manipuler les entités identifiées on se basant sur une API (Application Programming Interface ) .

### 3.3.1 Les étapes de l'approche

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

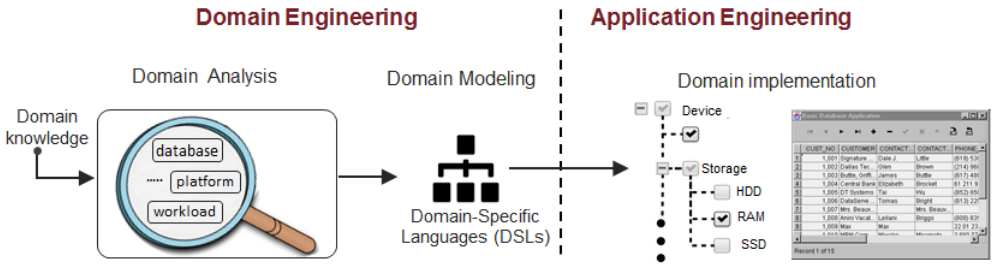


FIGURE 3.3 – notre approche

#### 3.3.1.1 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.

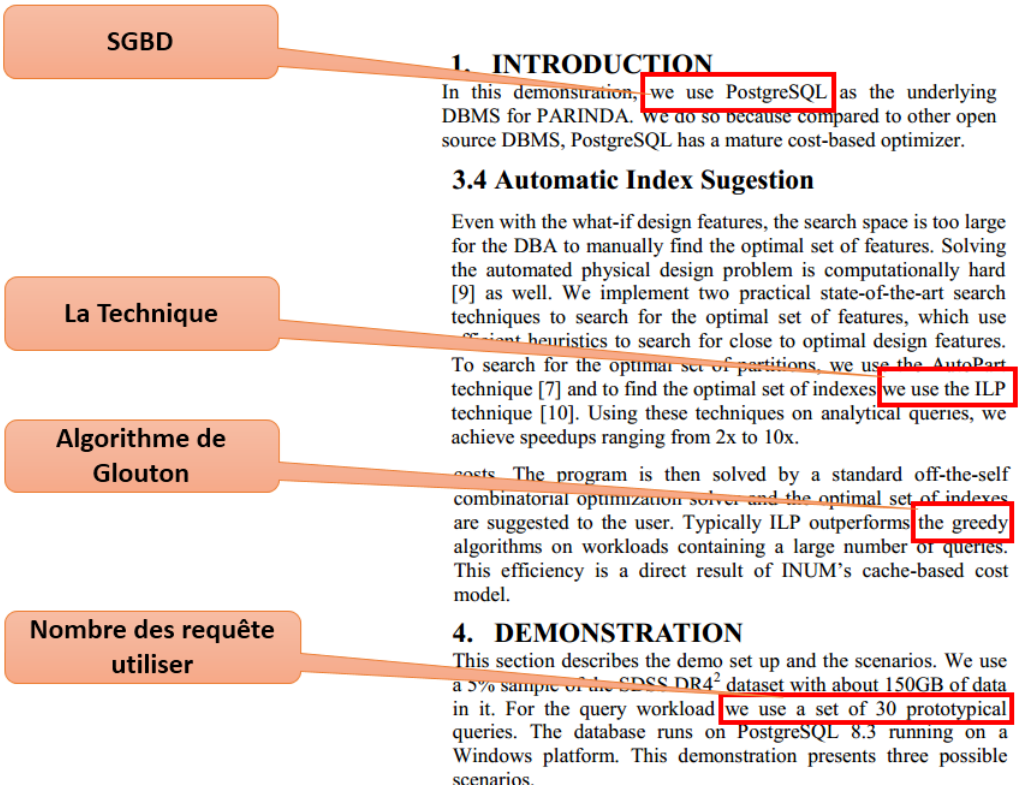


FIGURE 3.4 – Les composants de PARINDA

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 automatiques 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 3.3 illustre trois mode de sélection: (i) Automatique (ii) semi automatique et (iii) Interactive.

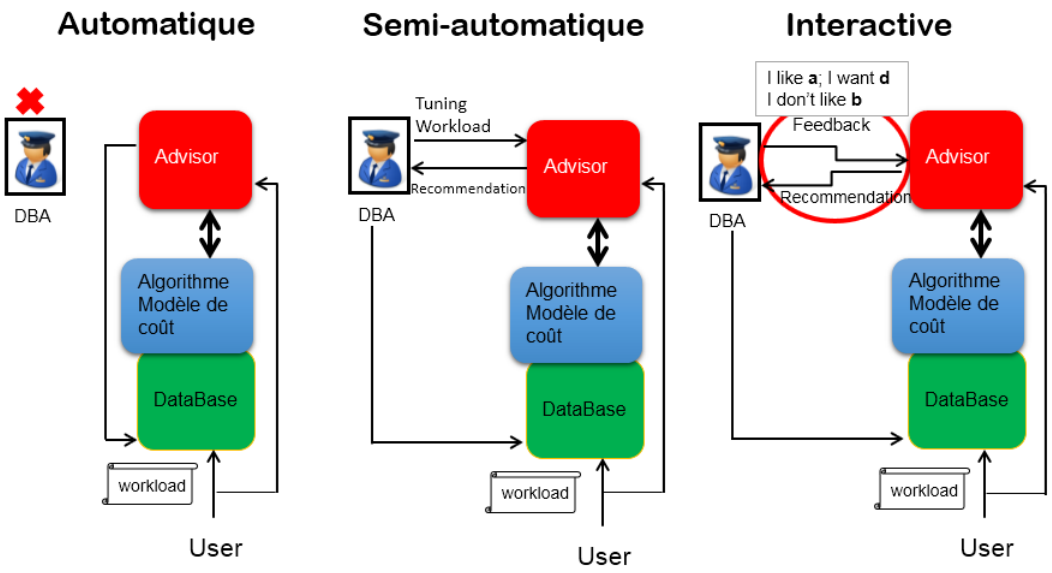


FIGURE 3.5 – Classification des Advisors

La figure 3.6 donne un aperçu général du processus de conception physique. Il montre trois domaines principaux. La première zone est consacrée aux DBA, 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 uti-

lisés doivent supporter comme entrée les spécificités du système sous la conception. Bien que certains outils de 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

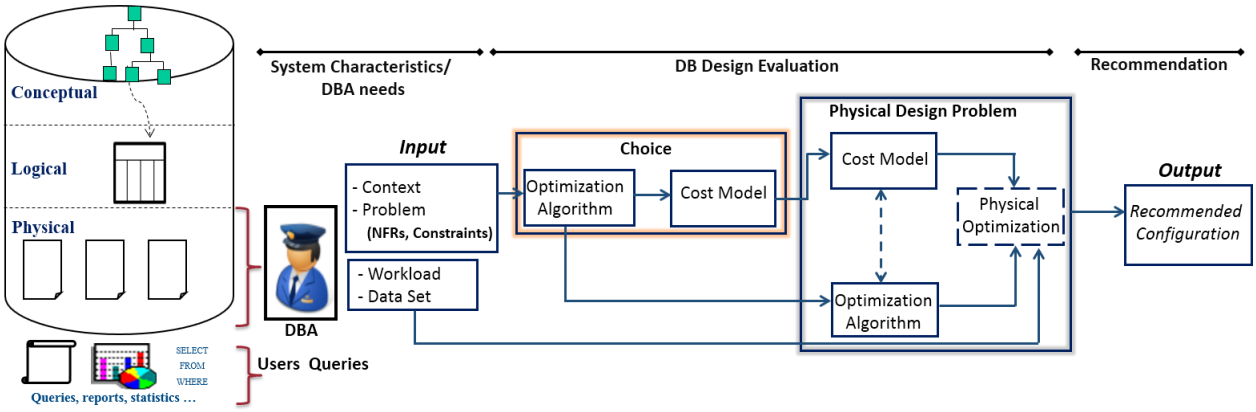


FIGURE 3.6 – Processus conceptuel de la conception physique de la base de données

### 3.3.1.2 Example

Nous présentons un exemple de problème de conception physique qui a été présenté dans [**?**]. Les auteurs de [**?**] considèrent un *problème* de "vues", où l'objectif est de minimiser la consommation d'énergie et le temps de réponse en fonction de l'espace de stockage et des contraintes de coûts de maintenance. Le classement non dominé des algorithmes génétiques *(NSGA-II)* a été choisi comme un algorithme d'optimisation afin de résoudre le problème pour une charge de travail donnée. Les auteurs proposent également un modèle de coût pour estimer la consommation totale d'énergie et le temps d'exécution des requêtes en considérant leurs coûts d'E/S et de processeur. Le système sous la conception est décrit par plusieurs paramètres liés à la base de données, aux requêtes et à la plate-forme. Les paramètres de la base de données sont: le schéma *relationnel*, *stockage orienté en ligne*, pas de compression de données, et *pipeline* [2] Stratégie de traitement des requêtes. Les paramètres de requêtes considérés sont: type de requêtes OLAP sans exécution simultanée. Les paramètres de la plate-forme sont: architecture de déploiement *centralisé*, *mémoire principale* en tant que périphérique de stockage principal et *disque dur* comme périphérique de stockage secondaire.

2. Une pipeline est un ensemble d'opérateurs de requêtes exécutant simultanément.

TABLE 3.1 – Exemple de problème de conception physique

| | | |
|---|---|---|
| **Problème** | Type | Problème de sélection des vues |
| | BnF | Consommation dénergie et temps de réponse |
| | Contraintes | Coût de maintenance, coût de stockage |
| **Paramètres du système** | Base de donnée | Stockage orienté ligne |
| | | Pas de compression de données |
| | | pipeline |
| | Requêtes | Type de requêtes sans exécution simultanée |
| | Plate-forme | Architecture de déploiements centralisés |
| | | Mémoire principale en tant que périphérique de stockage |
| | | Disque dur en tant que dispositif de stockage secondeur |
| **Modéle de Coût** | E/S et le Coûts d'énergie de CPU | |
| **Algorithme** | Tri non dominé des algorithmes génétiques (NSGA-II) | |

#### 3.3.1.3 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 3.7. 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.

   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 la figure 3.7) :

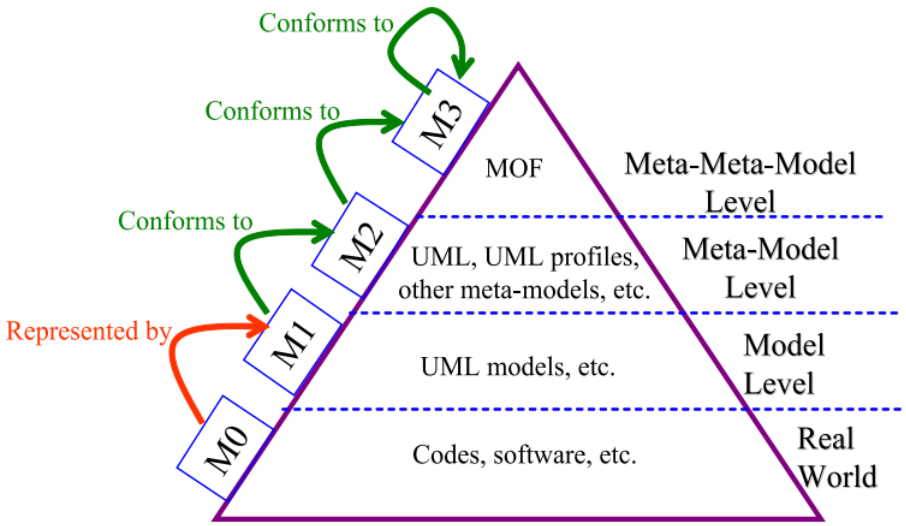


FIGURE 3.7 – L'architecture à quatre couches de MDA

- **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.

#### 3.3.1.4 Les éléments corps

Cette section est consacrée à présenter notre contribution basée sur un modèle. La figure 3.8 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 `hypothesis` dinstance.

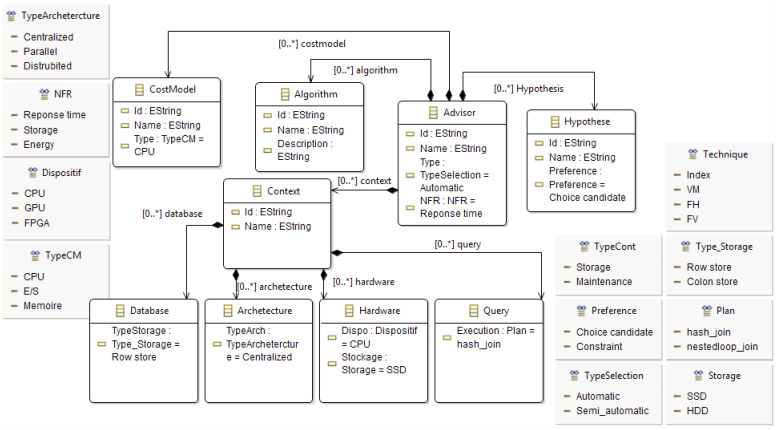


FIGURE 3.8 – Méta-modèle de l'Advisor

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 la figure 3.9). Il existe quatre types spécifiques d'algorithme: algorithmes *déterministes*, algorithmes *randomisés*, algorithmes *hybrides*, algorithmes *aléatoires*, algorithmes *hybrides* et *programmation dynamique*.

La figure 3.10 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 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 3.11 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

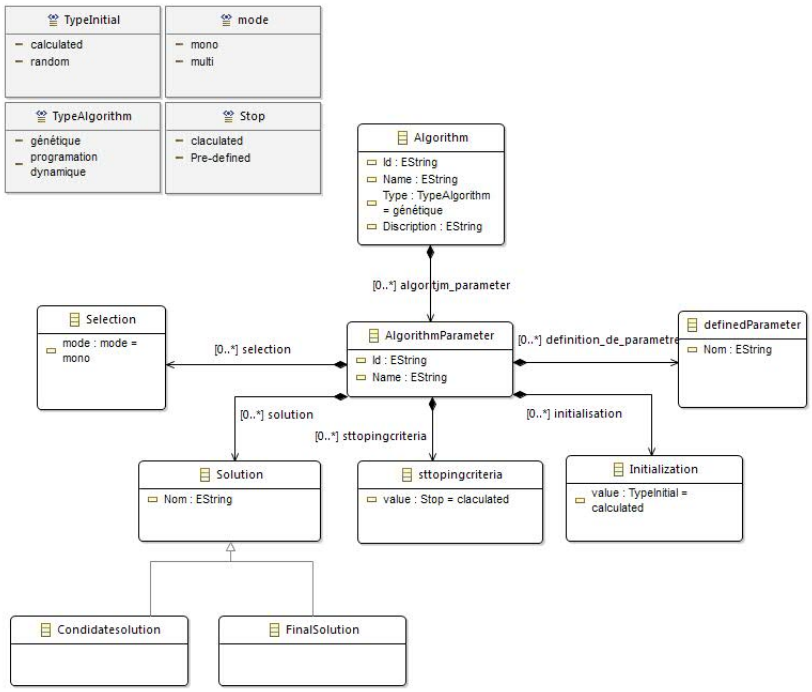


FIGURE 3.9 – Meta-modéle de l'algorithme de sélection

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 3.12.

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

**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

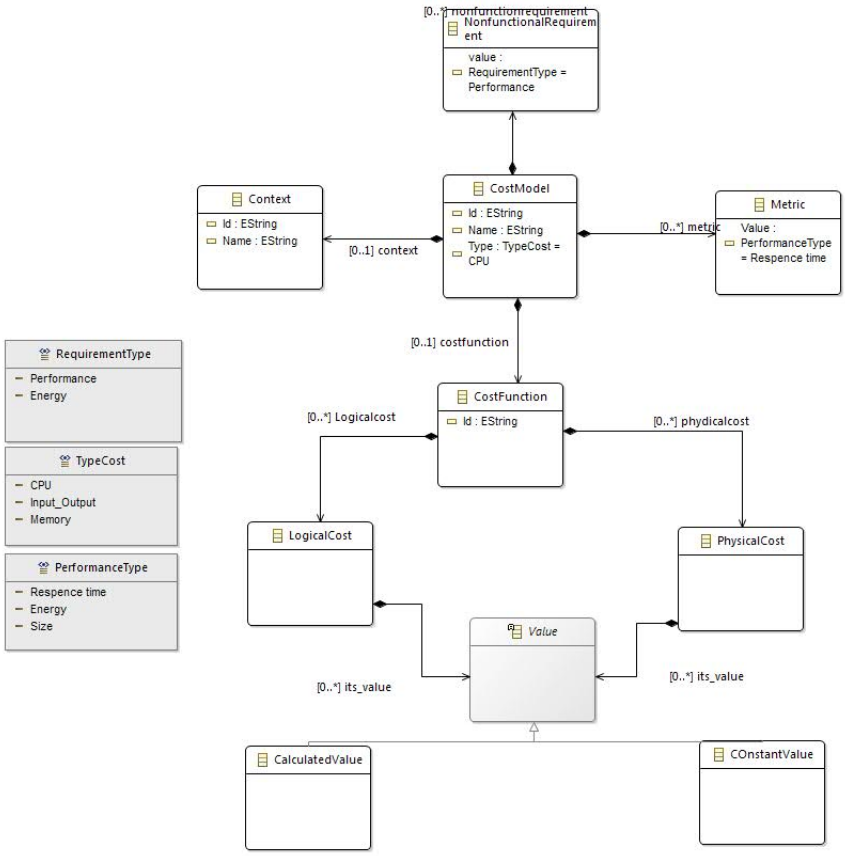


FIGURE 3.10 – Méta-modèle du modèle de coût

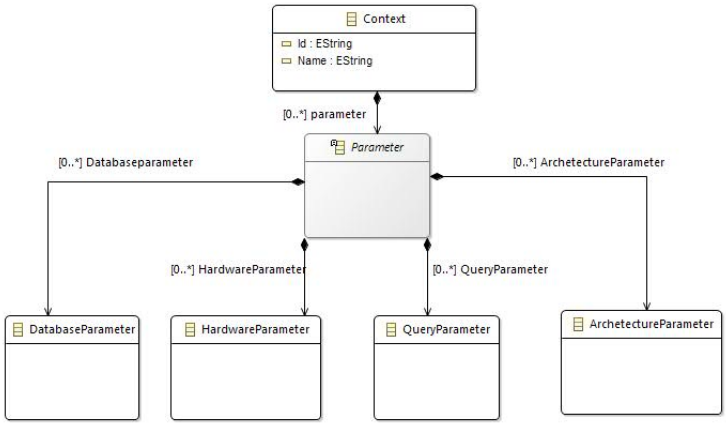


FIGURE 3.11 – Méta-modèle du Contexte

partagée, le disque partagé, etc.) et leurs paramètres comme le nombre de nuds dans un environnement parallèle. La figure 3.16 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

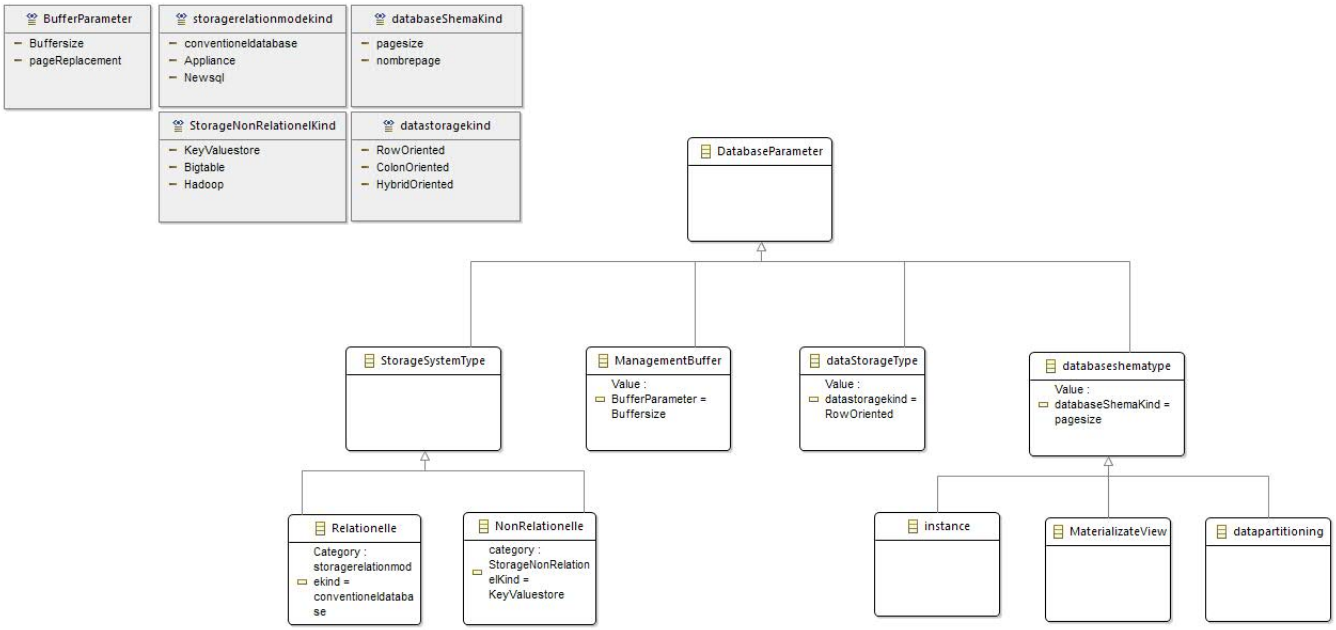


FIGURE 3.12 – Méta-modèle de la base de donnée

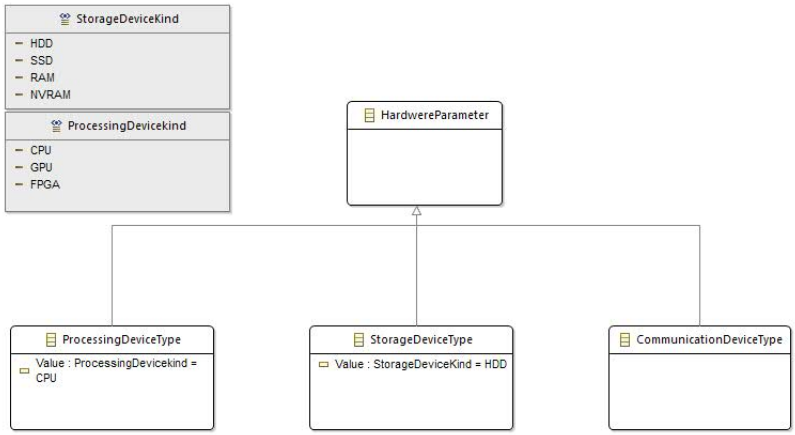


FIGURE 3.13 – Méta-modèle des paramètres matérielles

table, des attributs, est des structures physiques.

#### 3.3.1.5 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.

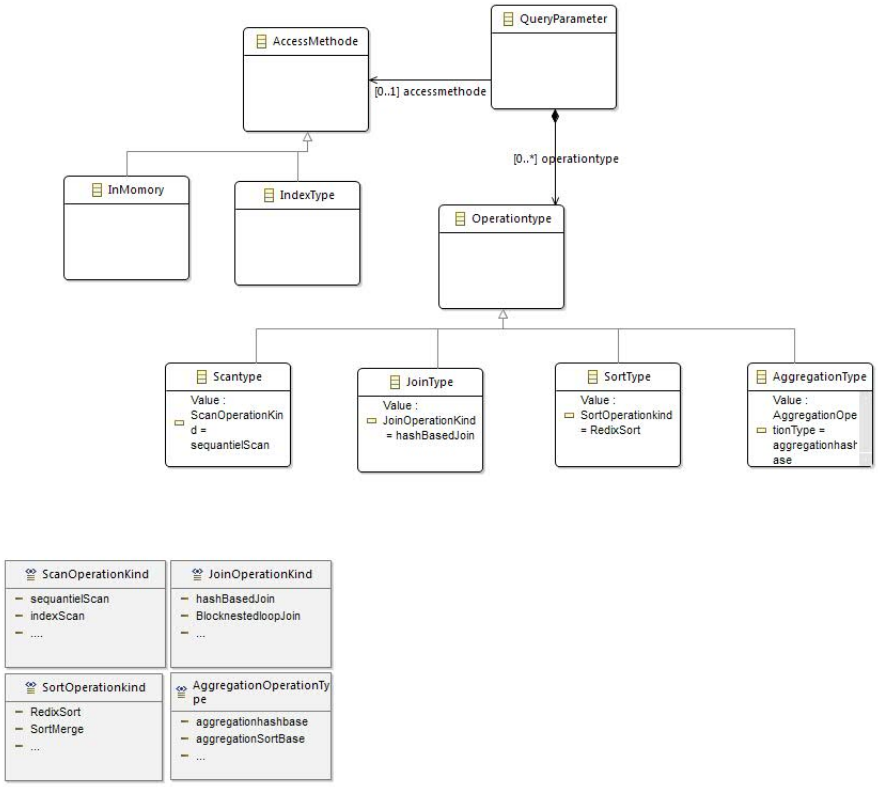


FIGURE 3.14 – Méta-modèle des requêtes

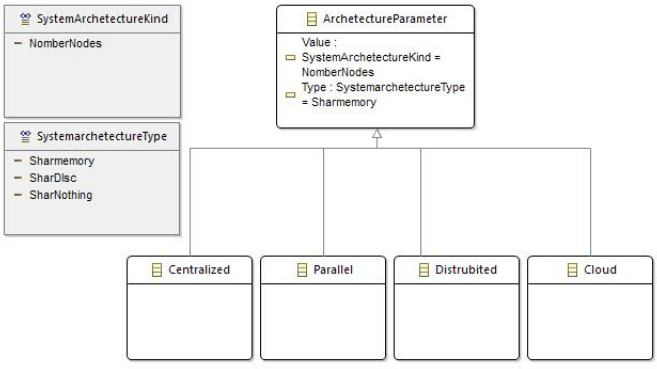


FIGURE 3.15 – Méta-modèle de paramètres d'architecture

Nous présentons dans cette section une instanciation du Framework proposé pour l'exemple présenté dans la section 3.3.1.2. 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 3.17. La figure 3.17 illustre l'ensemble des concepts de la spécification présentés dans l'exemple précédent.

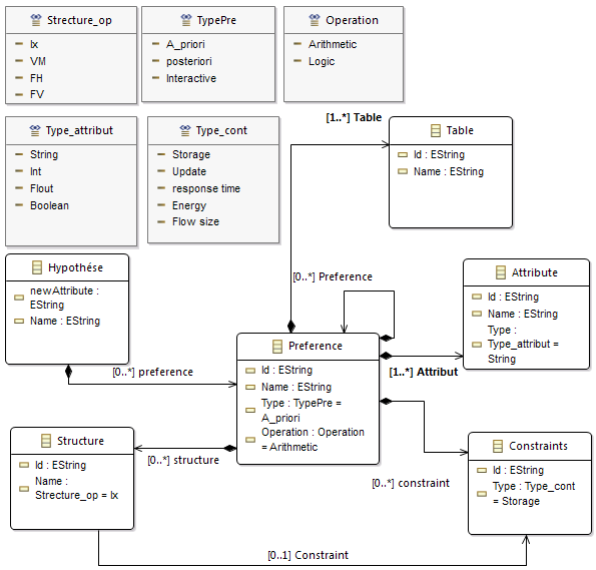


FIGURE 3.16 – Méta-modèle de préférence

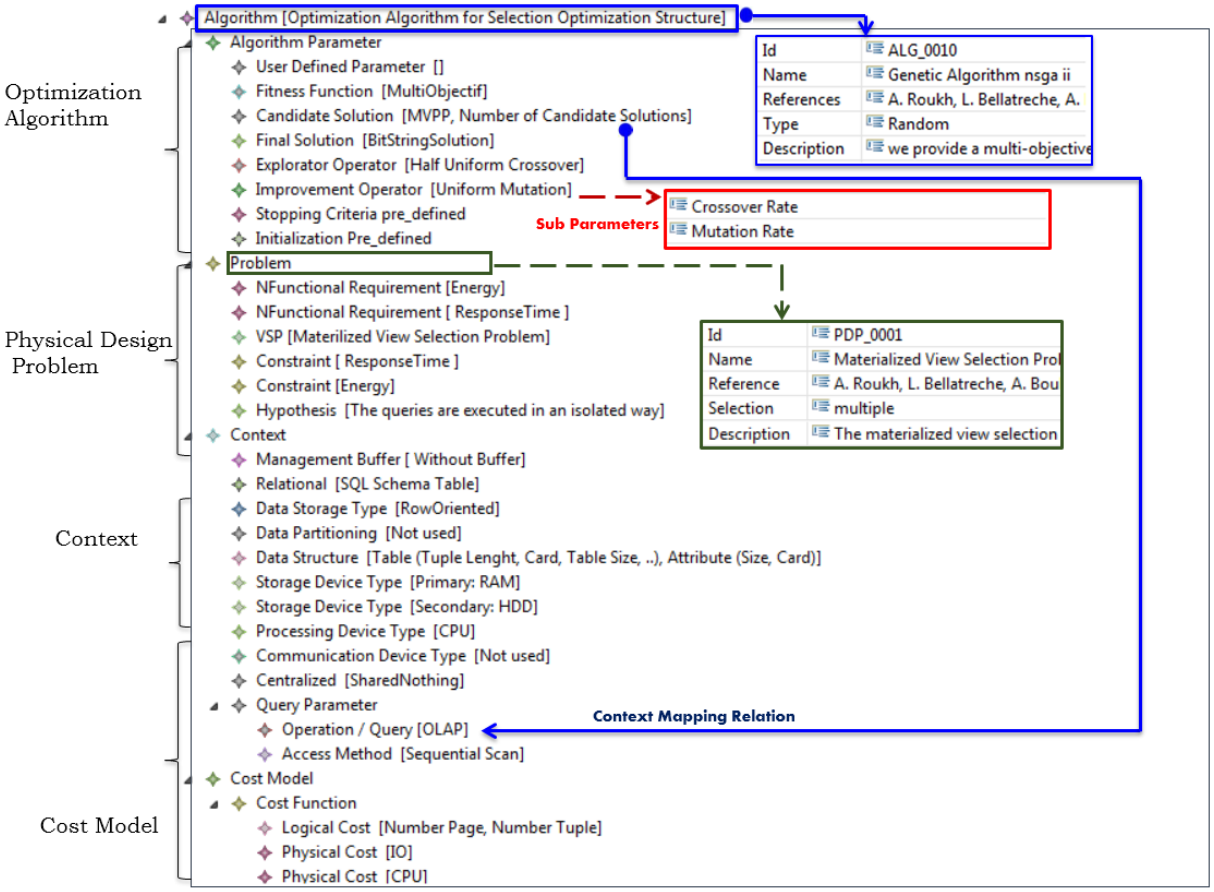


FIGURE 3.17 – Exemple d'instanciation de l'exemple dans 3.3.1.2

#### 3.3.1.6 Processus conceptuelle de sélection des structure d'optimisation

En terme d'usage, nous avons proposé un processus de structure d'optimisation prenons on considération des préférences de DBA pour améliorer la qualité de la solution finale. Le processus de sélection déroule principalement en quatre étapes qui sont :

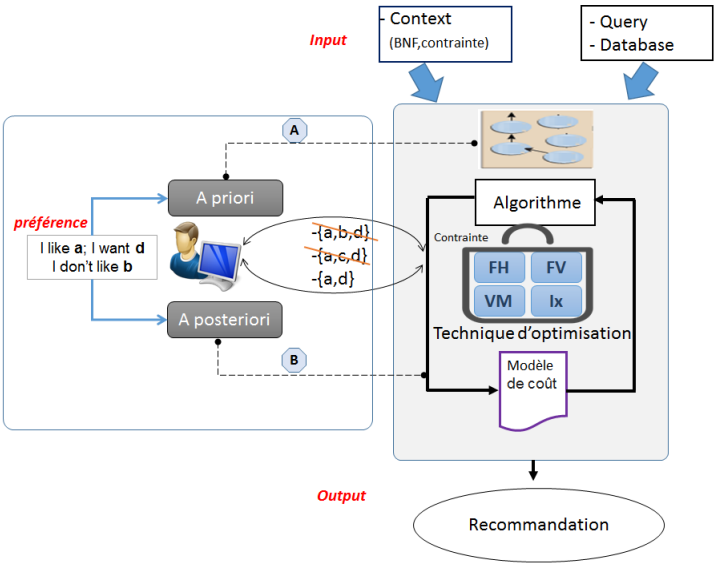


FIGURE 3.18 – Processus de sélection des structures physiques

(a) Génération de l'espace de solution: Cette étape commence par énumérer l'ensemble des attributs indexables. Ces attributs sont choisis parmi ceux présents dans les clauses WHERE, GROUP BY et ORDER BY. Le but de cette étape est de sélectionner une meilleure configuration pour chaque requête d'une manière indépendante.

(b) Appliquer les préférences des utilisateurs afin de réduire l'espace de recherche initial. Durant cette étape, plusieurs structures physiques non utiles apportant moins de gain de coût sont éliminés. Il permet de sélectionner les $k$ meilleurs structures physiques parmi les n structures physiques candidats.

(c) Appliquer l'algorithme de recherche basé sur un modèle de coût et sur des préférences a posteriori

(d) Trouver la meilleure configuration (configuration physique). La sélection des configurations physiques se fait en trois étapes :

   i. Trouver la meilleure configuration de taille m (m très petit par rapport à k). Un algorithme exhaustif peut être utilisé pour trouver cette configuration.

   ii. Ajouter la structure qui minimise le coût d'exécution à la configuration actuelle.

   iii. Répéter l'étape 2 tant qu'il y a des structures à ajouter réduisant le coût de requêtes.

Suivant la combinaison du processus d'optimisation et la prise de décision, les méthodes d'optimisation multi-objectifs peuvent être classées en trois catégories:

– **Méthodes a priori**. Prise de décision avant la recherche : Les fonctions objectifs du PCP sont regroupées en une seule objectif qui inclut implicitement les informations de compromis fournies par le décideur.
– **Méthodes interactives**. Prise de décision durant la recherche : Le décideur peut donner ses préférences de compromis au cours du processus d'optimisation d'une manière interactive. Après chaque étape d'optimisation, un certain nombre de compromis alternatifs sont présentés aux décideur, sur la base desquels il spécifie une autre information de préférence, qui guide respectivement la recherche.
– **Méthodes a posteriori**. Prise de décision après la recherche : L'optimisation est effectuée sans aucune préférence donnée. Le résultat du processus de recherche est un ensemble de solutions candidates (d'optimum de *Pareto*) dont le choix final de la meilleure solution est fait par le décideur.

## 3.4 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 de vues 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 do 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 le DBA ainsi que la conception de cette solution.

## Sommaire

Chapitre 4

# L'implémentation et la mise en œuvre de notre application

## Sommaire



## 4.1 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.

## 4.2 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 4.1 – La technologie de développement utiliser

### 4.2.1 Présentation du langage Java

**JavaSE** (Standard édition), édition standard, est une plateforme normalisée, destinée au développement de logiciels pour des ordinateurs personnels ainsi que des serveurs. La plateforme comporte une suite d'interfaces de programmation, qui permettent notamment de créer des interfaces graphiques, de manipuler des bases de données, des fichiers, d'utiliser le réseau.

### 4.2.2 Présentation de NetBeans IDE

NetBeans est un environnement de développement intégré (EDI), développé par Sun Microsystems, conçu en java. En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi langage, refactoring, éditeur graphique d'interfaces et de pages Web). Netbeans

comporte un ensemble complet d'outils de développement permettant de générer des applications Web en JEE, des Services Web XML, des applications bureautiques et des applications mobiles (Heffelfinger, 2008).

### 4.2.3 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.

### 4.2.4 ORACLE 11.g

Pour la gestion de la base de données de notre application on a choisi ORACLE 11.g. cest 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.

### 4.2.5 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.

### 4.2.6 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.

### 4.2.7 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.

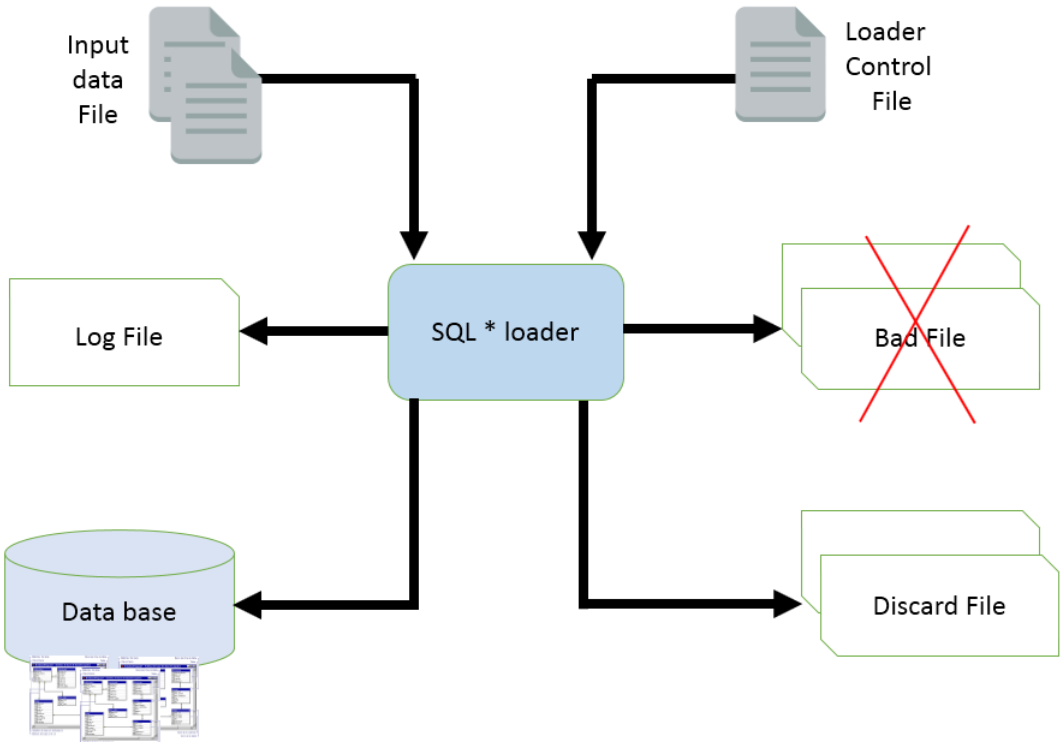


FIGURE 4.2 – Mécanisme de SQL Loader

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 4.3 et 4.4 illustre le mécanisme expliqué par avant sous exemple de la table « DIM_DATE » :

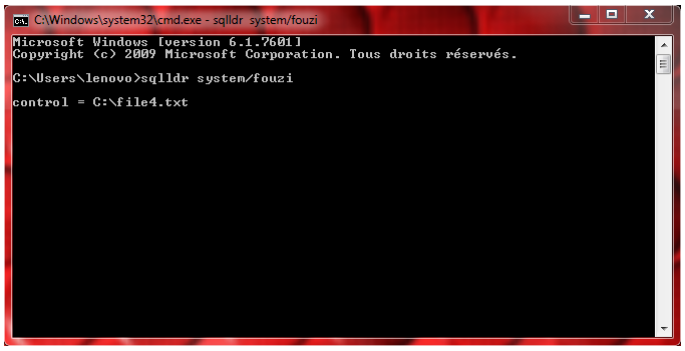


FIGURE 4.3 – Exemple de chargement de table « DIM_DATE »

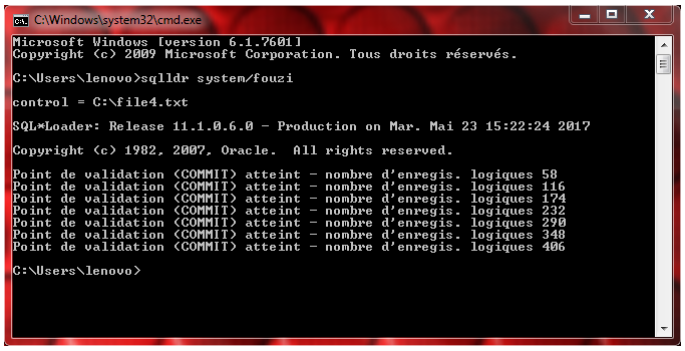


FIGURE 4.4 – Chargement de la Table Dim_Date

## 4.3 Présentation de notre outil

### 4.3.1 Objectifs

Notre outil gère principalement la technique d'optimisation des index IJB 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.

### 4.3.2 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.

#### 4.3.2.1 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)

**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

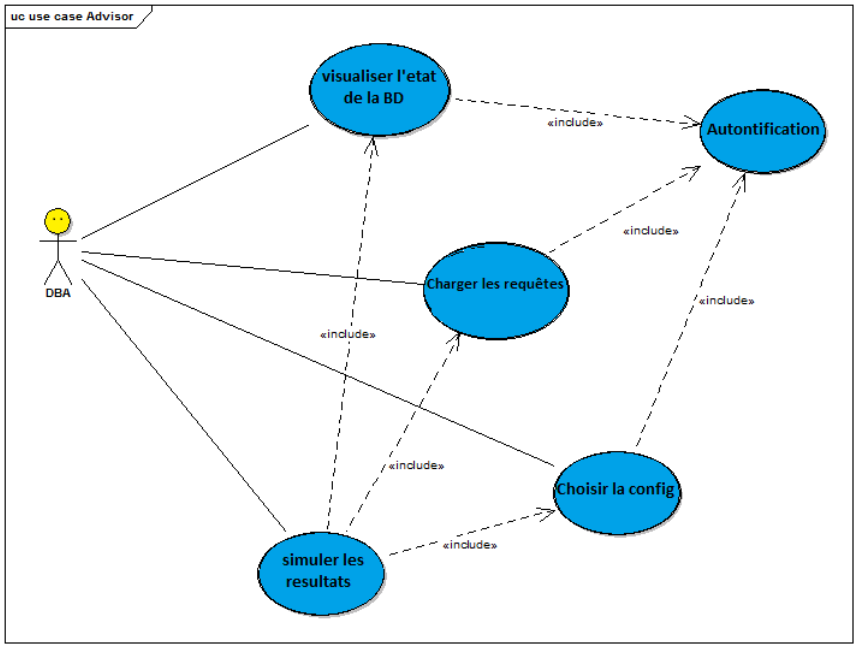


FIGURE 4.5 – Diagramme de cas d'utilisation

**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.

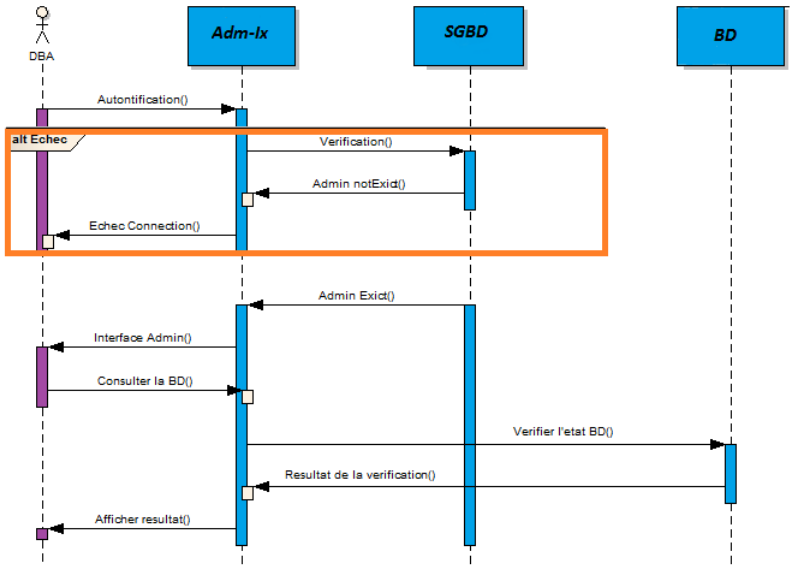


FIGURE 4.6 – Diagramme de séquence : visualisation de BD

***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.

**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.

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

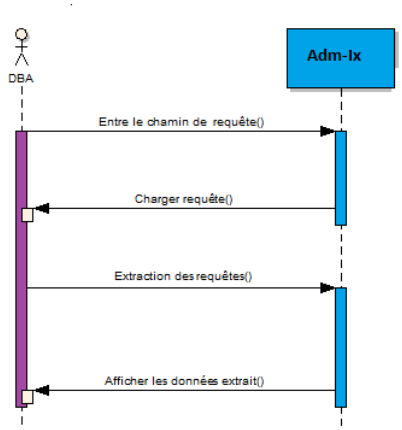


FIGURE 4.7 – Diagramme de séquence : chargement des requêtes

#### 4.3.2.2 Présentation des interfaces

***Architecture de notre outil*** : Dans cette section nous allons présenter l'architecture globale du fonctionnement de notre outil figure 4.9. 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 :

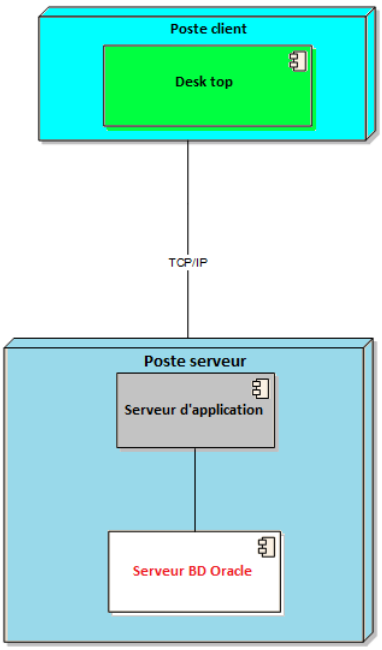


FIGURE 4.8 – Diagramme de déploiement

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.

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

Listing 4.1 – Exemple: Script d'extraction des informations des tables

```
TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH
                    FROM ALL_TAB_COLUMNS
                    WHERE table_name ='CUSTOMER'
                    or table_name ='PART'
                    or table_name ='DIM_DATE'
                    or table_name ='LINEORDER'
                    or table_name ='SUPPLIER' ORDER BY COLUMN_NAME
```

Listing 4.2 – Exemple: Script d'extraction des informations des attributs

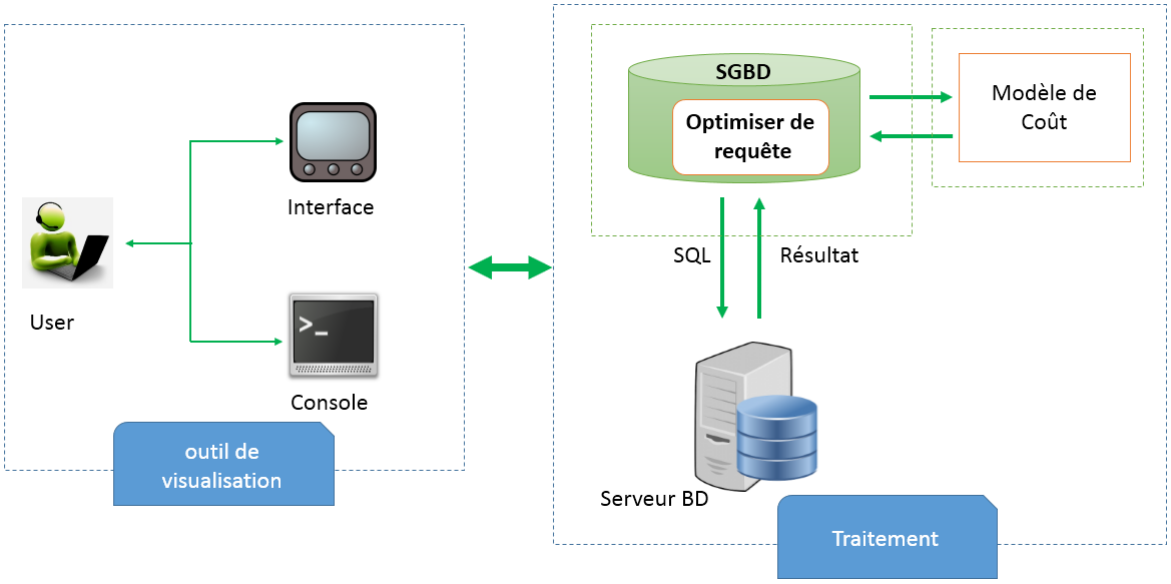


FIGURE 4.9 – L'architecture globale de notre l'outil

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.

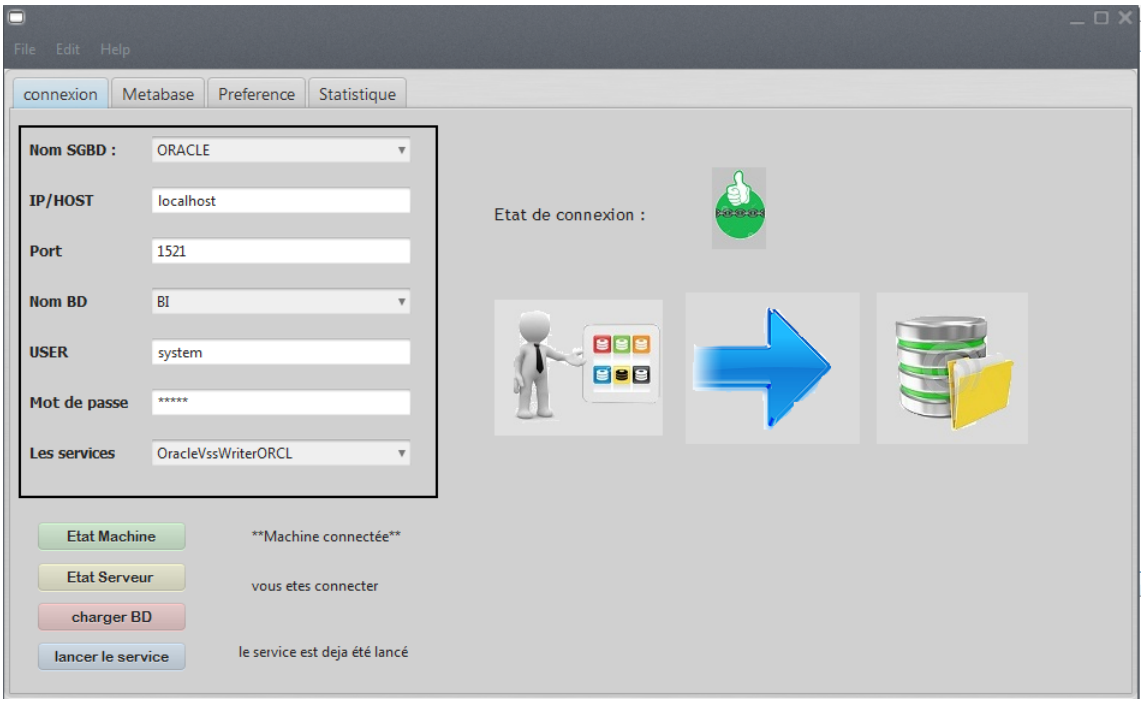


FIGURE 4.10 – Interface d'authentification.

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.

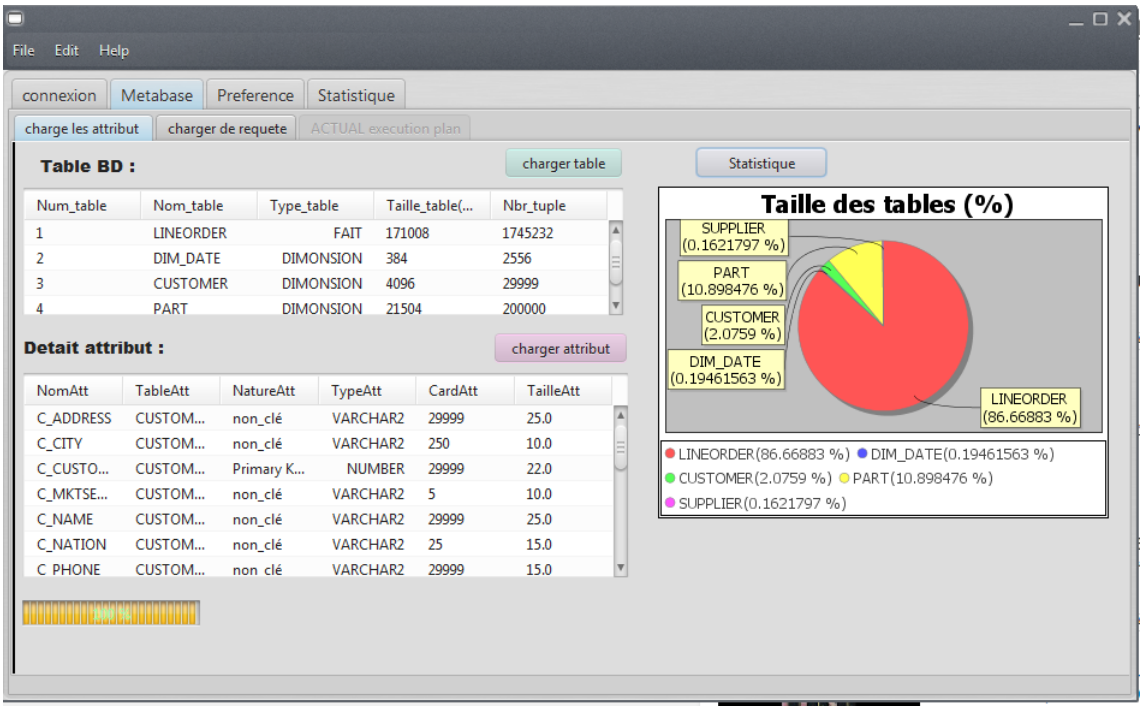


FIGURE 4.11 – Interface de chargement de méta-base.

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

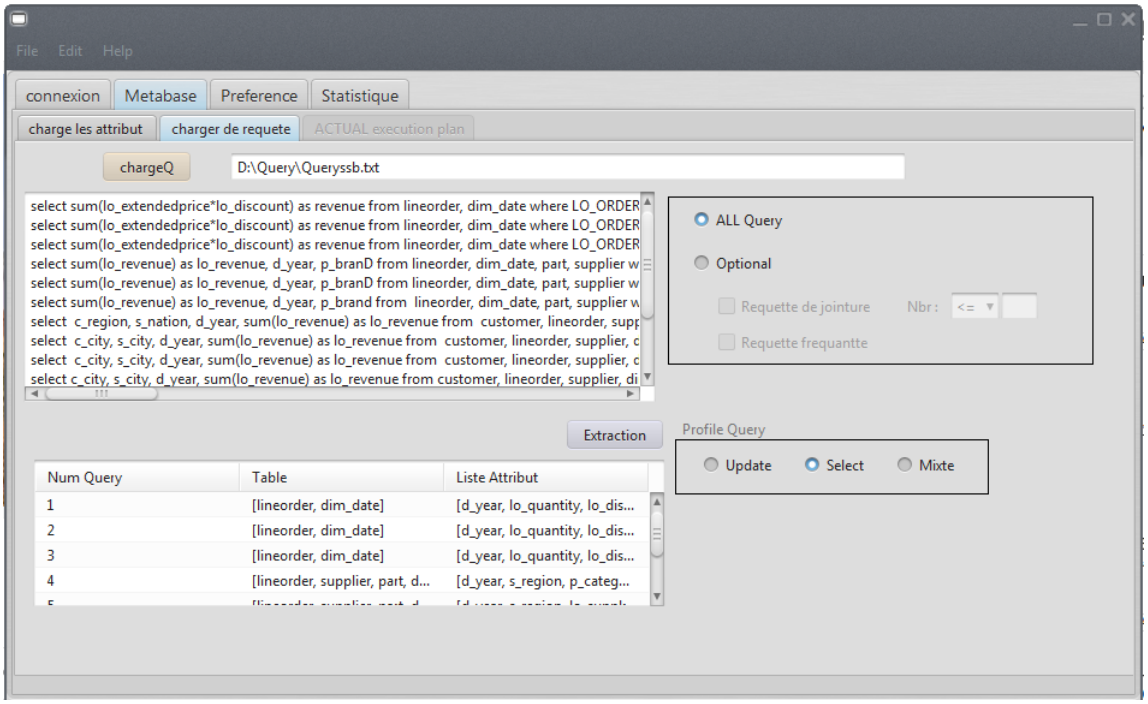


FIGURE 4.12 – linterface de chargement de requêtes

## 4.4 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.

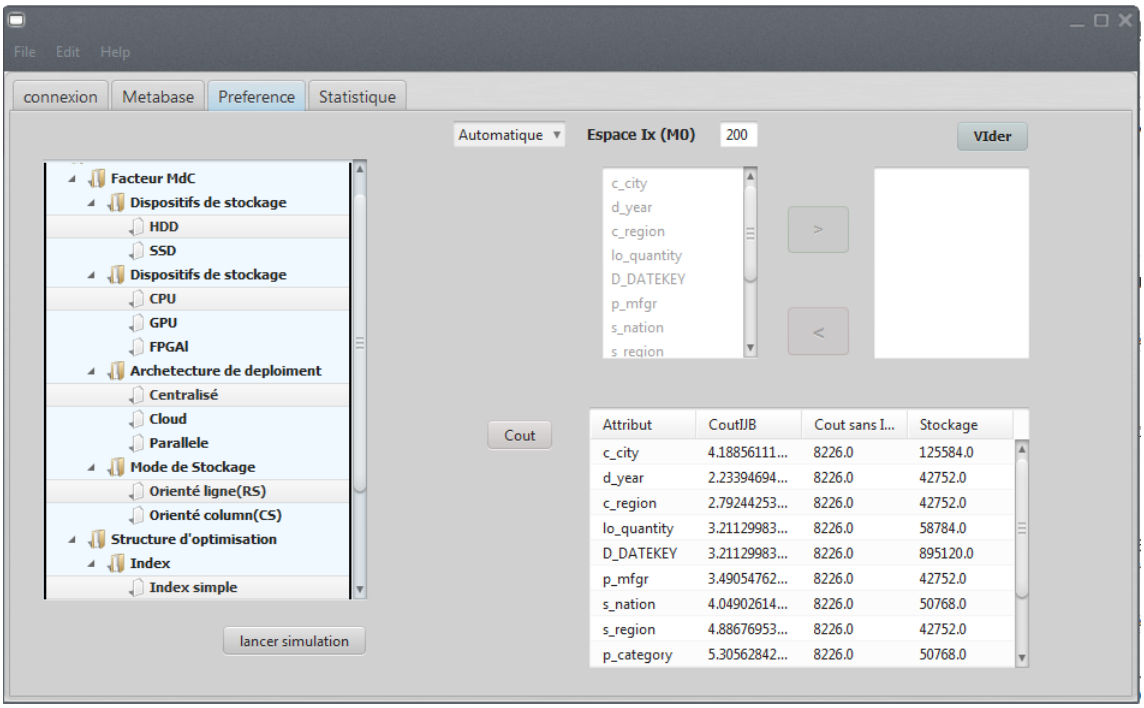


FIGURE 4.13 – Interface de configuration des paramètres

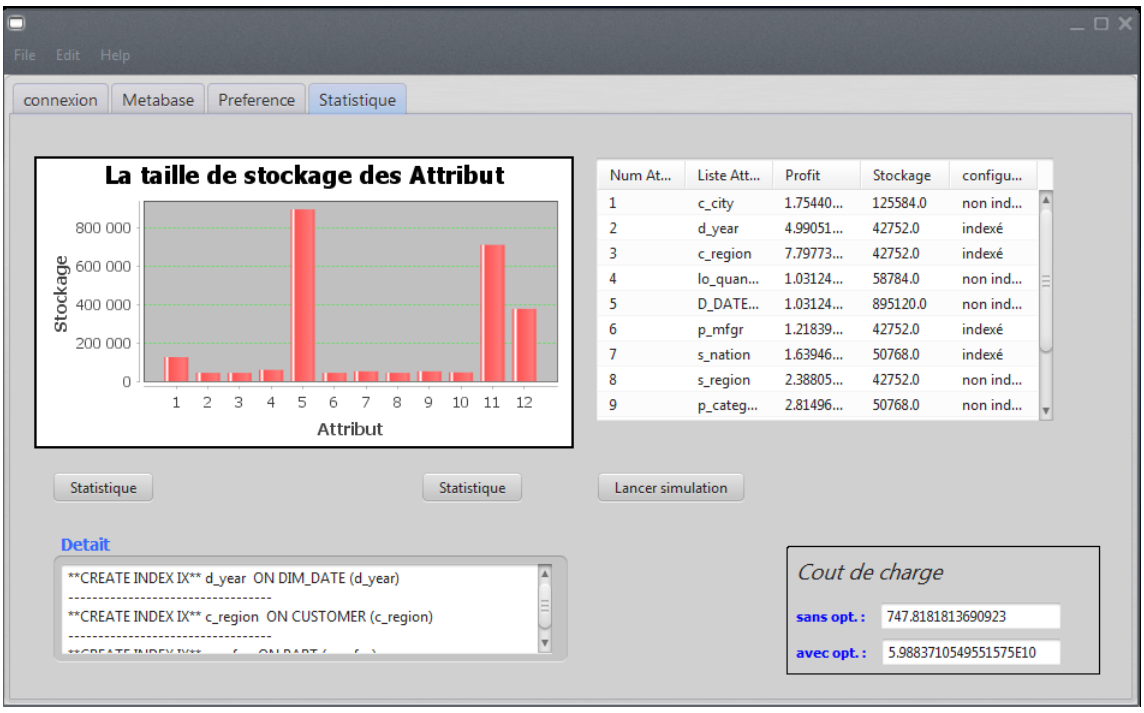


FIGURE 4.14 – Interface de visualisation de résultat

## Sommaire

Chapitre 5

# Gestion du Projet

*"If you tell people where to go, but not how to get there, youll be amazed at the results.".*
*- George Patton*

## 5.1 Introduction

L'initiation de tout projet nécessite une phase de planification qui a pour objectif de définir les tâches à réaliser, maitriser les risques et rendre compte de l'état d'avancement du projet. Dans ce chapitre, nous détaillons la méthode qu'on a suivie pour gérer le mémoire. Premièrement, nous détaillons notre organisation avec l'ensemble des personnes impliquées dans le mémoire. Puis, nous présentons la planification de l'ensemble du travail. Nous discutons ensuite, l'étude des risques, et nous terminons enfin avec les courbes d'avancement de mémoire

## 5.2 Démarche de développement

La méthode choisie doit s'adapter au mieux aux conditions dans lesquelles notre mémoire se construit.Le suivi est assuré à travers des réunions mensuelles et un contact permanent à travers les e-mails. À chaque réunion, nous présentons notre état d'avancement, nous développons les différents points avec l'encadreur. La démarche à suivre doit donc nous permettre une adaptation rapide aux changements. En conséquence, nous avons opté pour la démarche prototypage qui correspond à la nature de ce mémoire à savoir
– l'adaptation aux changements avec l'apparition de nouvelles spécifications.
– amélioration de la qualité de conception.
Le prototypage suit une approche itérative de développement, dans le sens où le processus de développement est constitué de plusieurs itérations où chaque itération livre un prototype (une partie du système) fonctionnel. D'une manière générale, la démarche de prototypage est constituée de quatre étapes décrites ci-après.

(a) **Analyse préliminaire des besoins** : cette étape nécessite l'identification des fonctionnalités principales du système nécessaires pour la proposition d'un premier prototype.

(b) **Développement du prototype** : construction du prototype

(c) **Évaluation du prototype** le prototype est évalué par l'encadreur pour valider s'il correspond aux besoins exigés.

(d) **Révision du prototype** : selon les résultats de l'évaluation (feedback), soit le prototype est validé, soit il doit être revisité en incorporant les nouvelles modifications pour satisfaire les besoins

Les différentes étapes de la démarche de prototypage sont synthétisées dans la figure 4.3

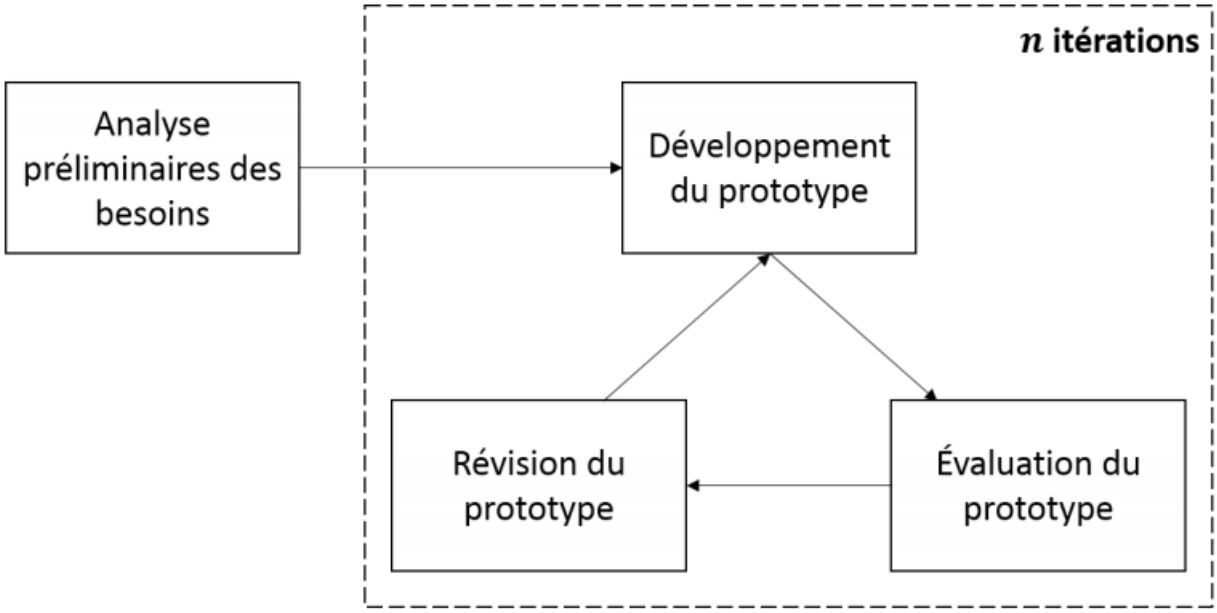


FIGURE 5.1 – Les quartes étapes du cycle de vie prototypage

Un prototype étant l'élément clé de la démarche de prototypage, nous le définissons par : *Un prototype est une version initiale/intermédiaire d'un système, utilisée pour démontrer des concepts et faire des essais de choix de conception*. [**?**]. Plusieurs méthodes de prototypage existent, comme le maquettage (prototypage rapide) qui permet la validation des spécifications, le prototypage expérimental qui permet d'explorer des choix conceptuels ou le prototypage évolutif qui permet d'avoir un produit final. Le choix d'une méthode de prototypage dépend de la nature du projet

## 5.3 Suivi du projet

Ce projet de fin d'étude a été suivi par l'encadrant pédagogique. Notre objectif principal est d'effectuer un travail complet de master issu de l'universitaire ibn khaldoune. Pour assurer le suivi de notre projet, nous envoyons chaque semaine un compte rendu pour lister les tâches effectuées, les actions à réaliser, les comptes rendus des réunions effectuées Le rôle de l'encadrant intervient dans la phase d'exploration des concepts théoriques, les orientations durant les principales phases du projet et dans la dernière phase de rédaction du rapport final du projet. Nous nous sommes basés sur une méthode de travail qui nécessite le contact hebdomadaire avec nôtre encadreur. Ce contact se traduit par les différents types des réunions Figure présentés dans le tableau suivant.

TABLE 5.1 – Le contact hebdomadaire avec l'encadreur

| **Type de contacte** | **Durée** | **Participant** | **Objectif** | **Redondance** |
|---|---|---|---|---|
| Mails hebdomadaires | | Étudiant Encadreur | échange d'information | Forte |
| Réunion de discussion | 20min a 2H | Étudiant Encadreur | Résoudre des problème | Forte |
| Réunion de présentation du travail | 30min a 2H | Étudiant Encadreur | Présentation du travail effectué | Mensuelle |
| Réunion de correction et validation | 20 a 40min | Étudiant Encadreur | Validation du, travail effectuée | Mensuelle |

### 5.3.1 Réunions

Des réunions hebdomadaire avec l'encadreur ont été programmées en présentiel pour présenter l'état d'avancement et effectuer une démonstration du prototype. Ces différentes réunions nous ont permis de discuter avec l'encadreur les différents choix conceptuels, d'échanger avec l'encadreur. À chaque réunion, nous prenons des décisions et nous fixons la date de la prochaine réunion.

### 5.3.2 livrables

Les livrables que nous allons énumérer dans cette section ont sont été partagés avec l'encadrant pédagogique pour assurer le suivi de notre projet.

#### 5.3.2.1 Documentation

tout au long du projet, nous avons réalisé des livrables intermédiaires qui ont permis aux encadrant d'être à jour avec les différentes fonctionnalités proposées,les livrables intermédiaires sont ceux suivants.

(a) le plan des chapitres.
(b) Architecture général du prototype.
(c) Fonctionnement des différents prototypes proposés.

#### 5.3.2.2 Rapport de PFE

il permet de mieux comprendre le raisonnement, le fonctionnement du système, son implémentation ainsi que l'évaluation des performances de l'architecture finale conçue.

### 5.3.3 Étude des risques

Sur un projet, nous sommes tous, amenés à rencontrer un ou plusieurs risques qui peuvent empêcher son avancement dans les délais prévus. C'est pour cette raison que nous avons évalué au préalable un ensemble de risques susceptibles de se manifester dans le cadre de ce travail. Comme le montre Le tableau suivant.

TABLE 5.2 – Le contact hebdomadaire avec l'encadreur

| Numéro | Nom du risque | Catégorie |
|---|---|---|
| 1 | Le niveau d'anglais | Technique |
| 2 | Compréhension technique | Technique |
| 3 | Absence de l'encadreur | Humain |
| 4 | Absence des étudiants | Humain |
| 5 | Diversité des outils | Technique |

#### 5.3.3.1 *Niveau d'anglais*

Étant donné, que la plupart des publications scientifiques sont rédigées en anglais. Ces dernières demandent un bon niveau technique d'anglais pour les biens assimiler. Actions à mettre en uvre :
– Ne pas hésiter à interroger son entourage.
– Lire beaucoup d'articles, afin de se familiariser avec les concepts techniques du domaine.
– Suivre des cours d'anglais.

#### 5.3.3.2 *Adaptation aux outils technologiques*

La diversité des outils que nous avons exploités durant la préparation du mémoire à demander un temps assez long dont on a appris la maitrise de ces derniers. les deux mois qui ont précédé ont été consacrés à l'apprentissage des outils.
Actions à mettre en uvre :
– prévoir au préalable les outils a exploités durant votre travail pour s'adapter.

#### 5.3.3.3 *Absence de l'encadreur*

L'absence de l'encadreur peut poser des problèmes pour la suite du projet, car, les étudiants peuvent effectivement avoir des difficultés en ce qui concerne :
– La validation de son travail, pour pouvoir passer à une autre tâche.
– Un manque de pistes de travail.
Les actions à mettre en uvre :

(a) Prendre rendez-vous avec le tuteur après son retour pour discuter des résultats du travail.

(b) Définir et comprendre au mieux la problématique de recherche afin de trouver des pistes de recherche.

(c) Avoir une bonne bibliographie afin d'enrichir son état de l'art, en laissant des articles scientifiques, si jamais l'étudiant est bloqué techniquement.

#### 5.3.3.4 *Absence des étudiants*

Un stagiaire est amené à tout instant d'être malade, ou avoir des situations qui l'obligent dêtre absent dans son stage. Actions à mettre en uvre :

(a) Avoir une version synchronisée de son travail, dans le cas où l'étudiant peut travailler de chez-lui.

(b) Faire des heures de travail supplémentaires à son retour.

#### 5.3.3.5 période des examens

La préparation des examens notamment la révision n'ont pas étés favorable quant à la disponibilité du temps libre.

## 5.4 Rédaction du rapport du PFE

la rédaction du mémoire s'est divisée en deux grandes périodes. La partie état de l'art du projet a été rédigée au fur et à mesure du développement durant les deux premiers mois. nous ne pouvions pas commencer la rédaction de la partie conception, réalisation et évaluation avant cette mars. Ainsi la deuxième phase de rédaction a commencé en fin de mars et s'est étalée jusqu'au mois de juin. Comme le montre le diagramme synthèse Gantt dans la figure 5.2 les étapes principales du projet sont celles suivantes.

## 5.5 Difficulté

Quant aux difficultés rencontrées durant ce projet, nous citons. Le manque d'expertise par rapport aux technologies imposées qui a donc nécessité un effort supplémentaire

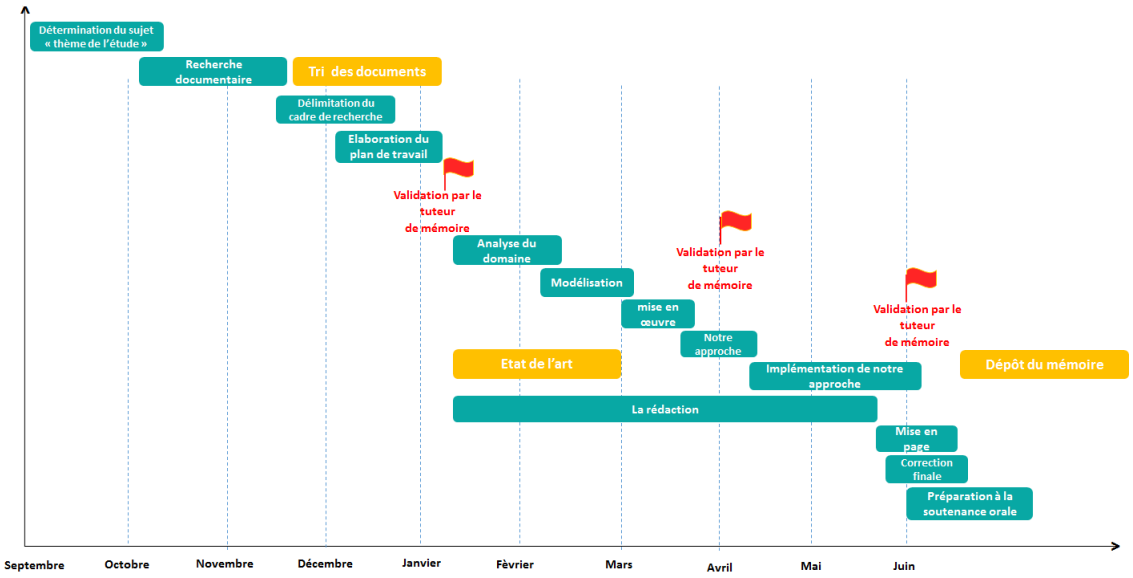


FIGURE 5.2 – Diagramme Gantt synthèse du déroulement de projet

d'apprentissage et de recherche durant le développement pour l'adaptation aux nouveaux outils technologiques.

## 5.6 Conclusion

Dans ce chapitre, nous avons montré que nous sommes en mesure d'évaluer des risques afin de les surmonter si ces derniers sont avérés. Comme par exemple le risque Absence de l'encadreur qu'a été rencontré, et que nous l'avons géré avec succès. La planification a montré aussi ses avantages en ayant toujours un repère de l'état d'avancement du travail et en permettant de respecter les délais des taches prévus.

# Conclusion

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

## 7 Conclusion

L'intérêt majeur de ce mémoire est dassisté 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 cur de ce travail. Nous avons présenté une nouvelle approche basée sur l'ingénierie dirigée par les modèles afin de faciliter et daider l'administrateur de la base de donnée DBA dans les tâches de conception physique et de tuning. Notre outil permet à l'administrateur deffectuer plusieurs choix, comme le choix de ces préférence, les objets de base de donnée (tables, attributs) participant dans le processus doptimisation, la charge de requêtes à optimiser, etc. Si ladministrateur 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 : lanalyse 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 dautres préférences.
– Implémenter dautre Algorithme de sélection.

- Enrichir l'outil par dautres techniques doptimisation, comme VM, FH, et FV.
- Détailler les dimensions identifiées (Dispositif de traitement : GPU, FPGA).
- Généraliser l'application sur dautre SGBD

.

# Annexe 1
# Modèle de cout

## 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.

### 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 $\mid A \mid -1 + \frac{\mid A \mid}{(m-1)} + d\frac{\| F \|}{8PS} + \| F \| (1 - e^{-Nr/\| F \|})$
avec Nr= d $\mid F \mid_{\overline{\mid A \mid}}$

### 5.8.2 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 $\times(\mid F \mid + \mid D_1 \mid) + \sum_{i=2}^{k} 3\times \mid D_i \mid$

TABLE 5.3 – 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 lattribut 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 |

# Annexe 2
# Liste des requêtes SSB

### Q01.1

```
select sum(lo_extendedprice*lo_discount) as revenue
from lineorder l, DATE d
where l.lo_orderdate=d.d_datekey
and d.d_year='1993' and l.lo_discount in (1,2,3) and l.lo_quantity=25;
```

Listing 5.1 – Requête Q01.1

### Q01.2

```
select sum(lo_extendedprice*lo_discount) as revenue
from lineorder l, DATE d
where l.lo_orderdate=d.d_datekey
and d.d_year='1994' and l.lo_discount in (1,2,3) and l.lo_quantity=25;
```

Listing 5.2 – Requête Q9.2

### Q01.3

```
select sum(lo_extendedprice*lo_discount) as revenue
from lineorder l, DATE d
where l.lo_orderdate=d.d_datekey
and d.d_year='1995' and l.lo_discount in (1,2,3) and l.lo_quantity=25;
```

Listing 5.3 – Requête Q1.3

### Q04.1

```

select sum(lo_revenue), d_year, p_brand
from lineorder l, DATE d, part p, supplier s
where l.lo_orderdate=d.d_datekey
and l.lo_partkey=p.p_partkey and l.lo_suppkey=s.s_suppkey
and p.p_category='MFGR\#12'and s.s_region='AMERICA'
group by d.d_year, p.p_brand order by d.d_year, p.p_brand;
```

Listing 5.4 – Requête Q04.1

### Q04.2

```
select sum(lo_revenue), d_year, p_brand
from lineorder l, DATE d, part p, supplier s
where l.lo_orderdate=d.d_datekey
and l.lo_partkey=p.p_partkey and l.lo_suppkey=s.s_suppkey
and p.p_category='MFGR\#12' and s.s_region='AFRICA'
group by d.d_year, p.p_brand order by d.d_year, p.p_brand;
```

Listing 5.5 – Requête Q14.2

### Q04.3

```
select sum(lo_revenue), d_year, p_brand
from lineorder l, DATE d, part p, supplier s
where l.lo_orderdate=d.d_datekey and l.lo_partkey=p.p_partkey
and l.lo_suppkey=s.s_suppkey and p.p_category='MFGR\#12'
and (s.s_region='ASIA' or s.s_region='MIDDLE EAST' or s.s_region='EUROPE')
group by d.d_year, p.p_brand order by d.d_year, p.p_brand;
```

Listing 5.6 – Requête Q04.3

### Q05.0

```
select c_city, s_city, d_year, sum(lo_revenue) as revenue
from customer c, lineorder l, supplier s, DATE d
where l.lo_custkey=c.c_custkey
and l.lo_suppkey=s.s_suppkey and l.lo_orderdate= d.d_datekey
and c.c_nation='UNITED STATES' and s.s_nation='UNITED STATES'
and d.d_year in(1992,1993,1994,1995,1996,1997)
group by c.c_city,s.s_city, d.d_year order by d.d_year asc, revenue desc;
```

Listing 5.7 – Requête Q05.0

### Q06.0

```
select sum(lo_revenue), d_year, p_brandc f
rom lineorder l, DATE d, part p, supplier s
where l.lo_orderdate=d.d_datekey
and l.lo_partkey= p.p_partkey and l.lo_suppkey= s.s_suppkey
and p.p_brand='MFGR\#2239' and s.s_region='EUROPE'
group by d.d_year, p.p_brand order by d.d_year, p.p_brand;
```

Listing 5.8 – Requête Q06.0

### Q07.0

```
select c_nation, s_nation, d_year, sum(lo_revenue) as revenue
from customer c, lineorder l, supplier s, DATE d
where l.lo_custkey=c.c_custkey
and l.lo_suppkey=s.s_suppkey
and l.lo_orderdate=d.d_datekey
and c.c_region='ASIA' and s.s_region='ASIA'
and d.d_year in(1992,1993,1994,1995,1996,1997)
group by c.c_nation, s.s_nation, d.d_year order by d.d_year asc, revenue desc;
```

Listing 5.9 – Requête Q07.0

### Q08.0

```
select sum(lo_revenue), d_year, p_brand
from lineorder l, .DATE d, part p, supplier s
where l.lo_orderdate=d.d_datekey
and l.lo_partkey=p.p_partkey
and l.lo_suppkey=s.s_suppkey
and p.p_brand in ('MFGR\#2221','MFGR\#2222','MFGR\#2223','MFGR\#2224',
'MFGR\#2225','MFGR\#2226','MFGR\#2227','MFGR\#2228')
and s.s_region='ASIA'
group by d.d_year, p.p_brand order by d.d_year, p.p_brand;
```

Listing 5.10 – Requête Q08.0

### Q09.0

```
select c_city, s_city, d_year, sum(lo_revenue) as revenue
from customer c, lineorder l, supplier s, DATE d
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_orderdate=d.d_datekey
and (c.c_city='UNITED KI1' or c.c_city='UNITED KI5')
and (s.s_city='UNITED KI1' or s.s_city='UNITED KI5')
and d.d_year in(1992,1993,1994,1995,1996,1997)
group by c.c_city, s.s_city, d.d_year order by d.d_year asc, revenue desc;
```

Listing 5.11 – Requête Q09.0

### Q10.0

```
select c_city, s_city, d_year, sum(lo_revenue) as revenue
from customer c, lineorder l, supplier s, DATE d
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_orderdate= d.d_datekey
and (c.c_city='UNITED KI1' or c.c_city='UNITED KI5')
and (s.s_city='UNITED KI1' or s.s_city='UNITED KI5')
and d.d_yearmonth = 'Dec1997'
group by c.c_city, s.s_city, d.d_year order by d.d_year asc, revenue desc;
```

Listing 5.12 – Requête Q10.0

### Q11.0

```
select d_year, c_nation, sum(lo_revenue - lo_supplycost) as profit
from DATE d, customer c, supplier s, part p, lineorder l
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_partkey=p.p_partkey and l.lo_orderdate=d.d_datekey
and c.c_region='AMERICA' and s.s_region='AMERICA'
and (p.p_mfgr='MFGR\#1' or p.p_mfgr='MFGR\#2')
group by d.d_year, c.c_nation order by d.d_year, c.c_nation;
```

Listing 5.13 – Requête Q11.0

### Q12.0

```
select d_year, s_nation, p_category,sum(lo_revenue - lo_supplycost) as profit
from DATE d, customer c, supplier s, part p, lineorder l
where l.lo_custkey=c.c_custkey
and l.lo_suppkey=s.s_suppkey and l.lo_partkey=p.p_partkey
and l.lo_orderdate=d.d_datekey
and c.c_region='AMERICA' and s.s_region='AMERICA'
and (d.d_year=1997 or d.d_year=1998)and(p.p_mfgr='MFGR\#1'
or p.p_mfgr= 'MFGR\#2')
group by d.d_year,s.s_nation,p.p_category
order by d.d_year,s.s_nation, p.p_category;
```

Listing 5.14 – Requête Q12.0

### Q13.0

```
select d_year, s_city, p_brand, sum(lo_revenue - lo_supplycost) as profit
from DATE d, customer c, supplier s, part p, lineorder l
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_partkey=p.p_partkey and l.lo_orderdate=d.d_datekey
and s.s_nation='UNITED STATES' and (d.d_year=1997 or d.d_year=1998)
and p.p_category='MFGR\#14'
group by d.d_year, s.s_city, p.p_brand
order by d.d_year, s.s_city, p.p_brand;
\end{verbatim}


\textbf{Q14.0 }
\begin{lstlisting}[caption=  Requ te Q14.0 , label=fig:scriptInsert][language=sql]

select d_year, s_city, p_brand1, sum(lo_revenue - lo_supplycost) as profit
from DATE d, customer c, supplier s, part p, lineorder l
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_partkey=p.p_partkey and l.lo_orderdate=d.d_datekey
and s.s_nation='EGYPT' and (d.d_year=1997 or d.d_year=1998)
and p.p_category='MFGR\#14'
group by d.d_year, s.s_city, p.p_brand
order by d.d_year, s.s_city, p.p_brand;
\end{verbatim}

\end{small}

\newpage
\textbf{Q15.0}
\begin{lstlisting}[caption=  Requ te Q15.0 , label=fig:scriptInsert][language=sql]
```

```
select d_year, s_city, p_brand, sum(lo_revenue - lo_supplycost) as profit
from DATE d, customer c, supplier s, part p, lineorder l
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_partkey=p.p_partkey and l.lo_orderdate=d.d_datekey
and s.s_nation='ALGERIA' and (d.d_year=1997 or d.d_year=1998)
and p.p_category='MFGR\#14'
group by d.d_year, s.s_city, p.p_brand order by d.d_year, s.s_city, p.p_brand;
```

Listing 5.15 – Requête Q13.0

### Q16.0

```
select d_year, s_city, p_brand, sum(lo_revenue - lo_supplycost) as profit
from DATE d, customer c, supplier s, part p, lineorder l
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_partkey=p.p_partkey and l.lo_orderdate=d.d_datekey
and s.s_nation='ALGERIA' and (d.d_year=1996 or d.d_year=1997)

and p.p_category='MFGR\#14'
group by d.d_year, s.s_city, p.p_brand order by d.d_year, s.s_city, p.p_brand;
```

### Q17.0

```
select d_year, s_city, p_brand, sum(lo_revenue - lo_supplycost) as profit
from DATE d, customer c, supplier s, part p, lineorder l
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_partkey=p.p_partkey and l.lo_orderdate=d.d_datekey
and s.s_nation='CANADA' and (d.d_year=1997 or d.d_year=1998)

and p.p_category='MFGR\#14'
group by d.d_year, s.s_city, p.p_brand order by d.d_year, s.s_city, p.p_brand;
\end{verbatim}


\textbf{Q18.0 }
\begin{lstlisting}[Requ te Q18.0 , label=fig:scriptInsert][language=sql]

select c_nation, s_nation, d_year, sum(lo_revenue) as revenue
from customer c, lineorder l, supplier s, DATE d
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
 and l.lo_orderdate=d.d_datekey and c.c_region='AMERICA'
 and s.s_region='AMERICA' and d.d_year in(1992,1993,1994,1995,1996,1997)
group by c.c_nation, s.s_nation, d_year order by d.d_year asc, revenue;
```

### Q19.0

```
select c_nation, s_nation, d_year, sum(lo_revenue) as revenue
from customer c, lineorder l, supplier s, DATE d
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_orderdate=d.d_datekey and c.c_region='MIDDLE EAST'
and s.s_region='MIDDLE EAST' and d.d_year in(1992,1993,1994,1995,1996,1997)
group by c.c_nation, s.s_nation, d_year order by d.d_year asc, revenue desc;
```

Listing 5.16 – Requête Q19.0

### Q20.0

```
select c_nation, s_nation, d_year, sum(lo_revenue) as revenue
from customer c, lineorder l, supplier s, DATE d
where l.lo_custkey=c.c_custkey and l.lo_suppkey=s.s_suppkey
and l.lo_orderdate=d.d_datekey and c.c_region='EUROPE'
and s.s_region='EUROPE' and d.d_year in(1992,1993,1994,1995,1996,1997)
group by c.c_nation, s.s_nation, d_year order by d.d_year asc, revenue desc;
```

# Bibliographie


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

[2] Sanjay Agrawal, Surajit Chaudhuri, Lubor Kollar, Arun Marathe, Vivek Narasayya, and Manoj Syamala. Database tuning advisor for microsoft sql server 2005: demo. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 930–932. ACM, 2005.

[3] Sanjay Agrawal, Vivek Narasayya, and Beverly Yang. Integrating vertical and horizontal partitioning into automated physical database design. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 359–370. ACM, 2004.

[4] Kamel Aouiche. *Techniques de fouille de données pour l'optimisation automatique des performances des entrepôts de données*. PhD thesis, Lyon 2, 2005.

[5] Franck Barbier, Corine Cauvet, Mourad Oussalah, Dominique Rieu, Sondes Bennasri, Carine Souveyet, Franck LIUPPA, Aix-Marseille LSIS, and Paris CRI. Composants dans lingénierie des systèmes dinformation: concepts clés et techniques de réutilisation. *Actes des deuxièmes assises nationales du GDR-I3*, 2002.

[6] Ladjel Bellatreche and Kamel Boukhalfa. Yet another algorithms for selecting bitmap join indexes. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 105–116. Springer, 2010.

[7] Ladjel Bellatreche, Kamel Boukhalfa, and Mukesh Mohania. Pruning search space of physical database design. In *International Conference on Database and Expert Systems Applications*, pages 479–488. Springer, 2007.

[8] Ladjel Bellatreche, Amine Roukh, and Selma 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. *Lecture Notes in Business Information Processing. Springer*, 2016.

[9] Soumia Benkrid. *Le déploiement, une phase à part entière dans le cycle de vie des entrepôts de données: application aux plateformes parallèles*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2014.

[10] Selma Bouarar. *Vers une conception lo-*

*gique et physique des bases de données avancées dirigée par la variabilité*. PhD thesis, Chasseneuil-du-Poitou, Ecole nationale supérieure de mécanique et d'aérotechnique, 2016.

[11] Kamel Boukhalfa. *De la conception physique aux outils dadministration et de tuning des entrepôts de données*. PhD thesis, Citeseer, 2009.

[12] Kamel Boukhalfa, Ladjel Bellatreche, and Pascal Richard. Fragmentations primaire et dérivée: Etude de complexité, algorithmes de sélection et validation sous oracle 10g. In *EDA*, pages 123–139, 2008.

[13] Stefano Ceri, Mauro Negri, and Giuseppe Pelagatti. Horizontal data partitioning in database design. In *Proceedings of the 1982 ACM SIGMOD international conference on Management of data*, pages 128–136. ACM, 1982.

[14] Surajit Chaudhuri and Vivek Narasayya. Autoadmin what-if index analysis utility. *ACM SIGMOD Record*, 27(2):367–378, 1998.

[15] Surajit Chaudhuri and Vivek Narasayya. Self-tuning database systems: a decade of progress. In *Proceedings of the 33rd international conference on Very large data bases*, pages 3–14. VLDB Endowment, 2007.

[16] Hong Tai Chou and David J DeWitt. An evaluation of buffer management strategies for relational database systems. *Algorithmica*, 1(1):311–336, 1986.

[17] Benoit Dageville, Dinesh Das, Karl Dias, Khaled Yagoub, Mohamed Zait, and Mohamed Ziauddin. Automatic sql tuning in oracle 10g. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1098–1109. VLDB Endowment, 2004.

[18] Iman Elghandour, Ashraf Aboulnaga, Daniel C Zilio, Fei Chiang, Andrey Balmin, Kevin Beyer, and Calisto Zuzarte. An xml index advisor for db2. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1267–1270. ACM, 2008.

[19] Ramez Elmasri and Sham Navathe. *Fundamentals of database systems*. , 2009.

[20] Himanshu Gupta, Venky Harinarayan, Anand Rajaraman, and Jeffrey D Ullman. Index selection for olap. In *Data Engineering, 1997. Proceedings. 13th International Conference on*, pages 208–219. IEEE, 1997.

[21] Himanshu Gupta and Inderpal Singh Mumick. Selection of views to materialize in a data warehouse. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):24–43, 2005.

[22] Gerald Kotonya and Ian Sommerville. *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[23] Darl Kuhn, Sam R Alapati, and Bill Padfield. Sql tuning advisor. In *Expert Oracle Indexing and Access Paths*, pages 211–234. Springer, 2016.

[24] MÁ Laguna, Francisco J García-Peñalvo, and O López. Metamodeling for requirements reuse. 2002.

[25] Cristina Maier, Debabrata Dash, Ioannis Alagiannis, Anastasia Ailamaki, and Thomas Heinis. Parinda: an interactive physical designer for postgresql. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 701–704. ACM, 2010.

[26] Stefan Manegold, Peter A Boncz, and Martin L Kersten. Optimizing database architecture for the new bottleneck: memory access. *The VLDB JournalThe International Journal on Very Large Data Bases*, 9(3):231–246, 2000.

[27] Bery Leouro Mbaiossoum. *Conception physique des bases de données à base ontologique: le cas des vues matérialisées*. PhD thesis, Chasseneuil-du-Poitou, Ecole nationale supérieure de mécanique et d'aérotechnique, 2014.

[28] Shamkant B Navathe. Evolution of data modeling for databases. *Communications of the ACM*, 35(9):112–123, 1992.

[29] Patrick O'Neil and Dallan Quass. Improved query performance with variant indexes. In *ACM Sigmod Record*, volume 26, pages 38–49. ACM, 1997.

[30] Abdelkader Ouared, Yassine Ouhammou, and Ladjel Bellatreche. Costdl: a cost models description language for performance metrics in database. In *Engineering of Complex Computer Systems (ICECCS), 2016 21st International Conference on*, pages 187–190. IEEE, 2016.

[31] Stratos Papadomanolakis and Anastassia Ailamaki. Autopart: Automating schema design for large scientific databases using data partitioning. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 383–392. IEEE, 2004.

[32] Peter Rob and Carlos Coronel. Design, implementation and management, 2004.

[33] Colette Rolland. Ingénierie des besoins: L'approche l'ecritoire. *Journal Techniques de l'Ingénieur*, page 1, 2003.

[34] Amine Roukh, Ladjel Bellatreche, Ahcène Boukorca, and Selma Bouarar. Ecodmw: eco-design methodology for data warehouses. In *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP*, pages 1–10. ACM, 2015.

[35] Thomas Stöhr, Holger Märtens, and Erhard Rahm. Multi-dimensional database allocation for parallel data warehouses. In *VLDB*, volume 2000, pages 273–284, 2000.

[36] Dennis Tsichritzis and Anthony Klug. The ansi/x3/sparc dbms framework report of the study group on database management systems. *Information systems*, 3(3):173–191, 1978.

[37] Patrick Valduriez. Join indices. *ACM Transactions on Database Systems (TODS)*, 12(2):218–246, 1987.

[38] Lengdong Wu, Liyan Yuan, and Jiahuai You. Survey of large-scale data management systems for big data applications. *Journal of computer science and technology*, 30(1):163, 2015.

[39] Daniel C Zilio, Jun Rao, Sam Lightstone, Guy Lohman, Adam Storm, Christian Garcia-Arellano, and Scott Fadden. Db2 design advisor: integrated automatic physical database design. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1087–1097. VLDB Endowment, 2004.

[40] Elhoussaine Ziyati. Optimisation de requêtes olap en entrepôts de données: Approche basée sur la fragmentation génétique. 2010.