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

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Logiciel

Par :

AIT YAHIA Fatima Imen
BOUHENOUCHE Khadidja

Sur le thème

Framework d'assistance des utilisateurs dédié à l'optimisation à base de coût des requêtes SQL

Soutenu publiquement le ... / 07 / 2019 à Tiaret devant le jury composé de :

Mr BENGHENI Abdelmalek	Grade	Université MAA	Président
Mr OUARED Abdelkader	Grade	Université MAA	Encadreur
Mr BENOUADA Habib	Grade	Université MAA	Examinateur

Remerciements

C'est avec grand plaisir que nous réservions cette page, en guise de reconnaissance à tous ceux qui nous ont aidé à la réalisation de ce travail.

Nous tenons tout d'abord à remercier Allah le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

En second lieu, nous tenons à remercier notre encadreur de ce mémoire de fin d'étude **Mr.OUARED Abdelkader** pour son suivi, ses précieux conseils,son orientation et son soutien,qu'il n'a pas cessé de nous prodiguer tout au long de notre recherche.

Nous remercions cordialement **Mr. BENGHENI Abdelmalek** pour nous avoir fait l'honneur d'être président du jury, de cette thèse de master. Nous souhaitons adresser également nos remerciements à **Mr. BENAOUDA Habib** d'avoir fait l'insigne honneur d'accepter d'examiner notre travail ;

Nous tenons également à remercie nos familles et nos amis qui par leurs prières et leurs encouragements, nous avons pu surmonter tous les obstacles.

Enfin, nous adressons nos plus sincères remerciements à tous ceux qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicace

Toutes les lettres ne sauraient trouver les mots qu'il faut… Tous les mots ne sauraient exprimer la gratitude, l'amour, Le respect, la reconnaissance… Aussi, c'est tout simplement que.

Je dédie cette Thése :

A Mes lumières de Ma vie, Maman et Papa :

Source de mes joies, secret de ma force, vous serez toujours le modèle. Papa, dans ta détermination, ta force et ton honnêteté, Maman dans ta bonté, ta patience et ton dévouement pour nous. Merci pour tous vos sacrifices pour que vos enfants grandissent et prospèrent, merci de trimer sans relâche, malgré les péripéties de la vie, au bien être de vos enfants. Merci d'être tout simplement mes parents, c'est à vous que je dois cette réussite. Et j'en suis fière de vous l'offrir. Merci Maman, Papa je vous aime très fort.

A Mon frère le bijou de la famille : Messaoud

Je te souhaite mon frère un avenir brillant et tous le bonheur du monde, que dieu t'ouvre les portes de la réussite et il te donne une vie pleine de joie, réussite, santé et sérénité .Je t'exprime à travers ce travail mes sentiments de fraternité et d'amour.

A Mes très chères sœurs : Lydia et Amel

Mes sœurs et amies et tout ce que j'ai dans la vie. Merci pour tous les agréables moments qu'on à passer ensemble, pour vos soutiens et vos amours. Je vous dédie ce travail à vous avec tous mes vœux de bonheur, santé et de réussite.

A Mes chères amies : Nassima et Khadidja

Je ne peux trouver les mots justes et sincères pour vous exprimer mon affectation et mes pensées, vous êtes pour moi des sœurs et des amies sur qui je peux compter. En témoignage de l'amitié qui nous unie je vous dédie ce travail et je vous souhaite une vie pleine de santé et bonheur.

Imen.

Dédicace

Je dédie ce modeste travail:

A mes très chers parents: vous m'avez donné la vie, la tendresse et le courage pour réussir. Tout ce que je peux offrir ne pourra exprimer l'amour et la reconnaissance que je vous porte, je vous remercie pour tous les sacrifices et pour l'affection dont vous m'avez toujours entourée, qu'Allah vous préserve et vous procure santé et langue vie.

A mon cher **frère** et mes adorables **sœurs** sans exception, je vous aime énormément, merci pour tous ce que vous faites pour moi.

A mes jolis **neveux** et **nièces** qui comblent ma vie de joie et du bonheur surtout les petits princes : ***Chourouk***, ***Abdeallah***, ***Abdenour***, ***Walid***, ***Bochera*** et ***Douaa***.

A mes chères amies avec lesquelles j'ai partagé des moments inoubliables : **Nassima** et mon binôme **Imen** qui m'a aidé à accomplir ce travail et sa famille.

***Khadidja*.**

Table des matières

Bibliographie 81

Table des figures

Liste des tableaux

Résumé

Actuellement, nous assistons à une explosion de données dans plusieurs domaines. Cette explosion a fait naître un besoin de stocker, gérer, et interroger d'une manière fiable et efficace cette masse de données. Cette efficacité est souvent assurée par l'optimisation à base de coût (CBO : Cost Based Optimization), qui est généralement réalisée à l'aide d'un modèle de coût estimant la qualité des requêtes définies sur cette masse de données. Ces modèles prennent en considération les paramètres liés à la base de données (ex. les tables, les attributs, et leurs domaines, etc...), aux requêtes, aux systèmes de stockage et à d'autres paramètres liés à l'architecture de déploiement et aux modèles de stockage. L'optimisation à base de coût est difficile à assimiler en raison de la complexité des paramètres liés aux SGBD. Nous explorons la littérature, il y a énormément des travaux qui s'intéressent au CBO mais pas de réflexion sur la facilité de cette expérience pour les étudiants, et les professionnels. Devant cette situation, nous avons identifié un besoin crucial pour les utilisateurs de SGBD (utilisateur novice, développeur, DBA etc.) qui s'articule autour des stratégies d'exécution possibles pour traiter une requête SQL, la réécriture des requêtes et l'utilisation des directives (Hints) pour influencer le plan d'exécution de ces requêtes SQL. Dans ce mémoire nous concentrons sur la résolution de ces difficultés liées au processus d'exécution de requêtes SQL en proposant un cadre qui prend en considération les différents aspects liés au CBO afin d'offrir une image mentale sur ce processus. Une preuve de concept est implémentée pour montrer la faisabilité de l'approche proposée.

Mots-clés : Base de données, Requête, Système de gestion des bases de données, Optimisation, Modèles de coût de l'optimiseur, Optimisation basée sur le coût.

Abstract

Currently, we are witnessing an explosion of data in several areas. This explosion gave rise to a need to store, manage and interrogate this mass of data efficiently. This efficiency is often provided by Cost Based Optimization (CBO). This optimization is often done using a cost model estimating the quality of the requests defined on this mass of data. These models take into account database-related parameters (eg, tables, attributes, attribute domains, etc.), queries, storage system, and other parameters related to the deployment architecture and model. storage. Cost-based optimization is difficult to assimilate because of the complexity of DBMS-related parameters. We explore the literature, there is a lot of work that is interested in CBO but no reflection on the ease of this experience for students, doctoral students and professionals. Faced with this situation, we have identified a crucial need for DBMS users (novice user, developer, DBA etc.) which is based on the possible execution strategies for processing a SQL query, the rewriting of queries and the use of Hints to influence the execution plan of these SQL queries. In this thesis we focus on solving these difficulties related to the execution of SQL queries by proposing a framework that takes into account the different aspects related to the and CBO in order to offer a mental image of this process. Proof of concept implemented to ascertain the feasibility of the proposed approach.

Keywords : Database, Queries, Database Management System, Optimization, Optimizer Cost Models, Cost Based Optimization.

ملخص:

فيما يخص البحث في مجال قاعدة البيانات، تعدّ البيانات الكبيرة ذات أهميّة عالية، إذ تلعب إدارتها دورا رئيسيّا في قاعدة البيانات، على الرغم من أحجامها الكبيرة وتعقيدات معالجتها، إلاّ أنّها تمثّل تحديّا لدى الشركات التّي تسعى من أجل الردّ السريع فيما يخصّ الأسئلة التّي يطرحها المستعملين. و هذا يتم تحقيقه بطريقة التحسين والّذي يعتبر جزءً مهمّا في عملية تنفيذ الأسئلة المرسلة، وكذلك تلعب دوراً مهمّا في جميع أنظمة تسيير قاعدة البيانات.

عملية التحسين تتم بواسطة التحسين المعتمد على حساب المدّة الزّمنية المستغرقة لتنفيذ استعلام معيّن، والتّي تعدّ عملية معقدّة الفهم نظراً للتعقيد الحاصل في مجموعة من العناصر مثل: منصّة العمل، قاعدة البيانات، الاستعلام و تكلفة المحسّن. الصّعوبة والتعقيد يظهران أيضاً في نماذج تكلفة المحسّن، الّذين يعتمدون على مجموعة من الصيغ المخصصة لكلّ عامل في مخطط التنفيذ. نتيجة لهذا، يجد المستعملون صعوبة في إنشاء صورة ذهنية خلال معالجة الاستعلام، بالرغم من وجود عدّة أعمال سابقة فيما يخص تحسين الاستعلامات لكنّها لا تظهر طريقة لتسهيل تعليم هذه المعرفة.

في هذه المذكّرة، يركّز عملنا على حلّ الصّعوبات و ذلك باقتراح إطار يهتم بكتابته و تحسين الاستعلامات. هذا الإطار يعتمد على تقنية بلوم . كبداية في هذه المذكّرة نعرض مبادئ و أساسيات على قاعدة البيانات و أنظمة سير قاعدة البيانات، ومن ثمّ نقدم شرحاً عن كيفية تنفيذ الاستعلامات، لنبرز بعد ذلك في الأخير طريقتنا في تطوير الإطار المقترح أعلاه.

الكلمات المفتاحية: قاعدة البيانات، الاستعلام، نظام تسيير قاعدة البيانات، تحسين، نماذج أنظمة التحسين، التحسين المعتمد على التكلفة.

Glossaire

BD : Base de Données.
DBA : Administrateur de la base de données.
SGBD : Système de Gestion de Base de Données.
DBMS : Data base Management System.
SQL : Structured Query Language.
LDD : Langage de Définition des Données.
LMD : Langage de Manipulation des Données.
SO : Structure d'Optimisation
MC : Modèle Conceptuel.
UML : Unified Modeling Language.
MCD : Modèle Conceptuel de Données.
MPD : Modèle Physique de Données.
E/S : Entité /Association.
ER : Entité Relationnelle..
API : Application Programming Interface.
CPU : Central Processing Unit.
OLAP : Online Analytical Processing..
CBO : Cost Based Optimizer.
RBO : Rule Based Optimizer.
BPMN : Business Process Modeling and Notation.
QBE : Query by Example.
VM : Vues Matérialisées.
Ix : Index.
FH : Fragmentation Horizontale
FV : Fragmentation Verticale
TP : Traitement Parallèle.
IJB : Join Binary Index.
CRUD : Create, Read, Update, Delete.
DC : Design Conceptuel.
DP : Design Physique.
DL : Design Logique.

Introduction Générale

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

0.1 Contexte et Motivation :

De nos jours, le domaine des *bases de données* (BD) connait une large évolution, suite à l'évolution concernant le matériel émergeant (emerging hardware) et du logiciel. Cette évolution concerne *les systèmes de gestion des bases de données* ((SQL, NoSQL)), les différentes plateformes (centralisée, distribuée, cloud etc...),par conséquent la conception de BD (conceptelle, logique, physique et déploiement) est devenue plus complexe. *La phase physique* inclut plusieurs aspects à savoir les structures d'optimisation (vue matérialisée, index et fragmentation...), la formulation des requêtes et leur optimisation.
L'optimisation des requêtes SQL est considérée comme une partie très importante dans les SGBD. Elle détermine le moyen le plus efficace d'exécuter une requête SQL. Cette partie est basée sur *l'optimisation à base de coût* (CBO), qui est très difficile à comprendre en raison de la complexité des plateformes, des BD, des requêtes et des SGBD. De plus, les modèles de coût sont complexes et dépendent à plusieurs paramètres comme les estimations de cardinalité, les propriétés du plan et des formules de coût de la sélectivité de chaque opérateur dans un plan d'exécution. Par conséquent, les apprenants et les praticiens n'arrivent pas d' avoir une image mentale et d'assimiler les diverses techniques pour améliorer les performances de la requête, (choisir le plan d'exécution , utilisation des directives (Hints) comme le HASH JOIN etc.)
Dans ce contexte, il est nécessaire de développer un framework qui aide les utilisateurs de mieux comprendre le processus d'exécution des requêtes SQL en particulier l'étape de l'optimisation.

0.2 Problématique :

Dans la littérature, il y a plusieurs outils et frameworks dédiés pour la conception des BD. Ces travaux ont concentré leurs travaux sur les différents aspects de la BD, tels que la formulation des requêtes, la conception des BD, l'optimisation des requêtes, et sur les différentes phases qui constituent le cycle de vie de la BD. Il existe énormément des travaux dédiés à l'optimisation des requêtes [7] (voir 7). Cette optimisation connait une complexité en termes de compréhension et de visualisation, en raison de la complexité des plateformes, la BD, le SGBD et les requêtes. En outre, les travaux dans cette optique n'ont pas bien focalisé sur l'expression suffisante pour mettre en clair les détails cachés derrière cette partie importante dans le processus d'exécution des requêtes SQL. Par conséquent, les différents praticiens ne peuvent pas extraire une image mentale sur le traitement lié à cette partie, surtout dans le cas où il existe des directives utilisées pour forcer l'optimiseur à choisir le meilleur plan, où la tache devient difficile chez un apprenant d'assimiler mieux le processus d'optimisation et d'avoir une image mentale sur ce dernier.

0.3 Objectifs :

Dans cette thèse, nous proposons un Framework d'assistance pour aider les apprenants (étudiants, industriels etc.) de comprendre l'optimisation des requêtes SQL et leur offrir une vue transparente sur les aspets liés à l'optimiseur de requêtes(Plan dexecution, directives d'optmisation etc.)
Afin de répondre à nos objectifs, nous nous sommes fixées un ensemble d'actions qui sont :

- Analyser l'optimiseur de requête pour identifier les principaux composants.
- Modéliser chaque composant par un modèle qui décrit leur description détaillée.
- Modéliser le processus d'execution de requête SQL par un BPMN (Business Process Modeling and Notation) qui décrit l'utlisation de notre Framework par Intégration de l'utilisateur dans les différentes étapes de ce processus (Formulation, plan d'excution, etc.).
- Proposer un outil prototype de notre framework assistant.

0.4 Organisation du mémoire :

Ce document est divisé en trois parties principales.

Partie 1 : Etat de l'art
Cette partie présente un état de l'art de ce mémoire, elle comporte les deux chapitres suivants :
– Le chapitre 1 décrit un état de l'art sur des concepts et des généralités de la base de données et les systèmes de gestion des bases de données.
– Le chapitre 2 présente un état de l'art sur le processus d'exécution des requêtes SQL.

Partie 2 : Contribution
La deuxième partie présente une description de notre framework qui décrit notre solution. Elle concerne notre approche proposée et son implémentation. Elle comporte les deux chapitres 3 et 4.

– Le chapitre 3 décrit notre contribution, il comporte les trois sections suivantes : – la section 1 correspond à la description de la méthodologie associée à notre framework, et comporte un scénario expliquant l'étape de l'optimisation. – la section 2 correspond à la description de notre approche. – la section 3 correspond à une représentation graphique qui décrit la structure et le processus de notre framework. – Le chapitre 4 décrit notre outil prototype.

Partie 3 : Conclusion et perspectives
Cette partie présente une synthèse de notre solution, des limites et des perspectives de notre proposition. La figure 1.1 illustre l'organisation de notre mémoire.

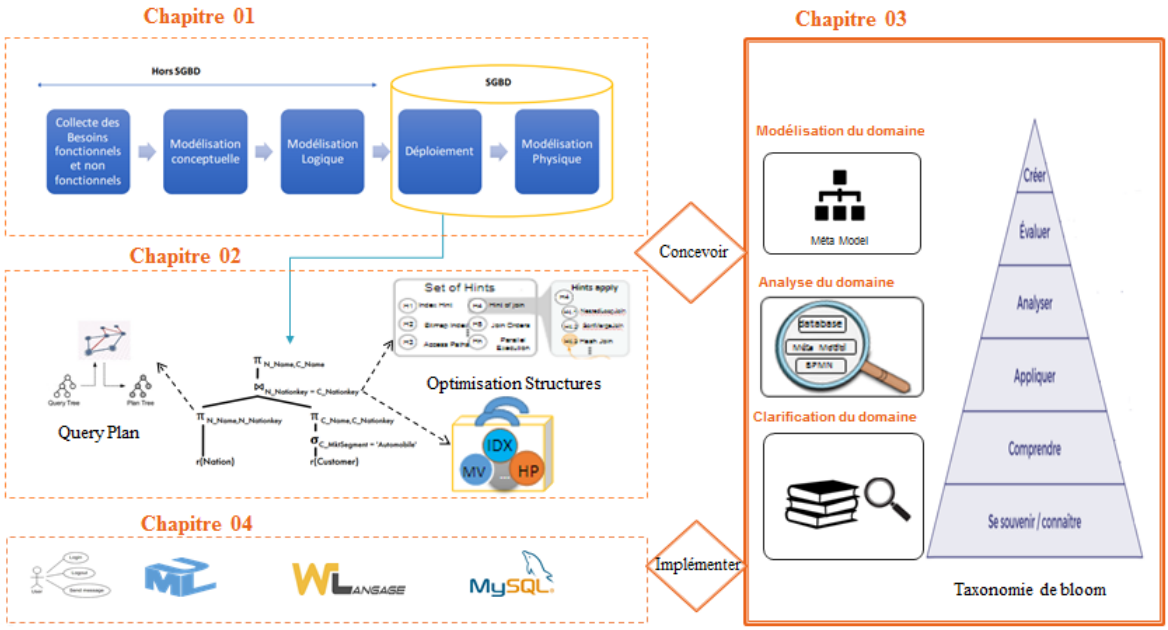


FIGURE 1 – Répartition des chapitres de notre mémoire.

Première partie

Etat de l'art

Chapitre 1

Généralités sur les systèmes des bases de données

« The true teaching is not to speak to you but to lead you. »
— Antoine De Saint-Exupéry

1.1 Introduction

Les données volumineuses qui se trouvent dans des différents endroits (ex. entreprises, universités, ...) nécessitent un stockage de façon organisée et plus ordonnée pour mieux les gérer et les exploiter, c'est pour ce faire la notion de base de données (BD) a eu naissance, donc une BD est un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter leur exploitation. Pour faire manipuler, gérer et partager les données stockées dans la BD, il faut un moyen qui est le système de gestion de base de données (SGBD).
Un SGBD (en anglais DBMS Data base Management System) permet d'inscrire, de retrouver, de modifier, de trier, et de transformer les informations de la BD. Il permet de comporter des mécanismes pour assurer la cohérence des informations, éviter des informations dues à des pannes, assurer la confidentialité.
Nous présentons dans ce chapitre un état de l'art portant sur les BD et les SGBD. En premier lieu, nous présentons la BD : son cycle de vie, des outils assistants durant ce cycle de vie, et les architectures de la BD. En second lieu, nous abordons le SGBD : son architecture, ses composantes et ses fonctionnalités.

1.2 Cycle de vie de base de données

La création et l'évolution des bases de données suivent un schéma itératif appelé cycle de vie, ce dernier inclut un ensemble de phases, nous distinguons cinq principales phases :

1.2.1 Analyse 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.

1.2.2 Modélisation conceptuelle

L'objectif de cette étape est de concevoir une base de données indépendante du logiciel et des détails physiques de l'implémentation. La sortie de ce processus est un modèle de données conceptuel qui décrit les principales entités de données, les attributs, les relations et les contraintes d'un domaine de problème donné. Cette conception a une forme descriptive et narrative. C'est-à-dire qu'elle est généralement composée d'une représentation graphique ainsi que des descriptions textuelles des principaux éléments de données, relations et contraintes. L'objectif est d'intégrer toutes les parties dans un schéma conceptuel global complet, non redondant et cohérent.

1.2.3 Modélisation logique

L'étape de modélisation logique prend en entrée le MCD et fournit en sortie un Modèle Logique des Données (MLD) en suivant des règles de traduction prédéfinies. L'objectif de la modélisation logique est de concevoir une base de données à l'échelle de l'entreprise basée sur un modèle de données spécifique, mais indépendante des détails physiques de l'implémentation. La conception logique doit contenir que des tables correctement normalisées (1FN, 2FN, 3FN, FNBC) et nécessite également la définition des domaines d'attributs et des contraintes appropriées dans le cas des SGBDR.

1.2.4 Modélisation physique

La modélisation ou la conception physique est le processus de détermination de l'organisation du stockage de données et des caractéristiques d'accès aux données afin d'assurer l'intégrité, la sécurité et les performances du système. Il s'agit de la dernière étape du processus de conception de base de données. Les caractéristiques de stockage sont déterminées en fonction des types de périphériques pris en charge par le matériel, le type de méthodes d'accès aux données prises en charge par le système et le SGBD. Dans cette étape, il faut choisir les bonnes structures d'optimisations physiques : vues matérialisées, partitionnement de données, indexes, etc. La conception physique pourrait devenir un travail très technique qui affecte non seulement l'accessibilité des données dans le(s) périphérique(s) de stockage, mais aussi la performance du système.

1.2.5 Déploiement et maintenace

Le résultat des phases de conception de base de données est une série d'instructions détaillant la création de tables, d'attributs, d'index, de contraintes de sécurité, de règles de stockage, des déclencheurs (triggers) et des fonctions de hachage d'une architecture centralisée ou distribuée, etc. Dans cette phase, toutes ces spécifications de conception doivent être mises en œuvre. L'administrateur teste, évalue et ajuste la base de données pour s'assurer qu'elle fonctionne comme prévu. L'administrateur doit aussi effectuer des activités de maintenance dans la base de données. Certaines des activités de maintenance périodique comprennent la sauvegarde, la restauration, la modification des attributs et tables, la génération de statistiques, l'audit de la sécurité, etc.

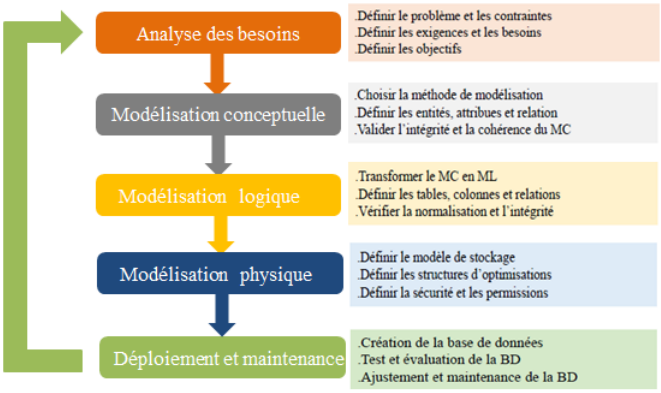


FIGURE 1.1 – Phases du cycle de vie de BD

1.3 Outils d'assistanse durant le cycle de vie de BD

Plusieurs outils ont été proposés durant le cycle de vie de la base de donné afin de faciliter leur compréhension et pour bien exploiter le traitement des données, ces outils ont été injectés dans chaque phase de ce cycle (la phase conceptuelle (PC), la phase logique (PL) et la phase physique (PP)).
Nous présentons ci-dessous les outils assistants les plus connus durant chaque phase de cycle de vie de BD.

1.3.1 Outils assistants pour la phase conceptuelle (PC)

Un ensemble d'outils ont été proposés durant la phase conceptuelle afin d'unifier et de bien représenter les informations collectées dans la phase d'élaboration de cahier de charge, parmi ceux les plus utilisés nous citons : entité-association (E/A) proposé par Peter Chen en 1976 [3], Unified Modeling Language (UML), Express, etc...

1.3.2 Outils assistants pour la phase logique (PL)

Un ensemble d'outils ont été proposés durant cette phase logique afin de modéliser les traitements effectués.Nous pouvons citer : ERwin, AMC Designer [1], BPwin [2] et Rational Rose [3] d'IBM.

1.3.3 Outils assistants pour la phase physique (PP)

Comme des travaux au niveau physique nous pouvons trouver les advisors qui ont pour but d'aider les administrateurs des BD (DBAs) pendant leurs tâches de la sélection des structures d'optimisation telles que les vues matérialisées et les index. Ces travaux sont : Index Tuning Wizard, Tuning Advisor dans Microsoft SQL Server[5], PARINDA pour PostgreSQL [6].

La figure suivante illustre les outils cités précédemment.

1. http ://help.sap.com/poweramc
2. http ://www.bpmmicro.com/downloads/
3. http ://www-03.ibm.com/software/products/fr/enterprise

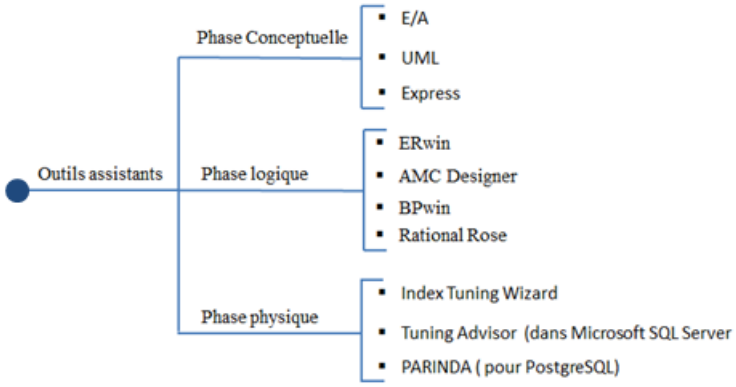


FIGURE 1.2 – Outils assistants durant le cycle de vie de BD

1.4 Architecture du système des BD

Au début de l'informatique, l'approche dominante de la gestion des BD était centralisée, les données sont gérées par un seul calculateur. Au milieu des années 80, la notion de BD répartie (distribuée) est apparue, engendrée par des développements remarquables dans les divers domaines tels que les réseaux de communication, et sur les mini et les micros ordinateurs. Donc nous distinguons dans la base de données une variante d'architectures qui sont utilisées dans différents domaines, parmi ces architectures nous citons :

1.4.1 Bases de données centralisées

La BD dans un système centralisé est stockée sur une seule machine, les utilisateurs y accèdent via des terminaux. L'avantage de cette approche se résume dans sa simplicitée. Cependant, elle est inadéquate pour le traitement de grandes quantités des données.

1.4.2 Bases de données réparties ou distribuées

Une BD distribuée est une BD dont certaines portions de données sont stockées sur plusieurs endroits physiques. Le traitement est réparti ou répliqué entre différents points d'un réseau. Les bases de données distribuées peuvent être homogènes ou hétérogènes. Dans le cas d'un système de base de données distribuée homogène, tous les emplacements physiques fonctionnent avec le même hardware et tournent sous le même système d'exploitation et les mêmes applications de bases de données. Au contraire, dans le cas d'une BD distribuée hétérogène, le hardware, les systèmes d'exploitation et les applications de bases de données peuvent varier entre les différents endroits physiques.

1.4.3 Bases de données fédérées

Contrairement à une BD répartie, une BD fédérée ne supporte pas de schéma global. La notion de la BD fédérée consiste à relâcher la contrainte des couplages forts des BD réparties. Il peut y avoir plusieurs BD sur un site gérée chacune par un SGBD indépendant. La motivation essentielle de cette approche est d'assurer l'autonomie totale des différentes BDs et de privilégier leur administration, leur gestion et leur manipulation indépendante. [7]

1.4.4 Bases de données parallèles

Une BD parallèle est un ensemble de données réparties sur des nœuds d'une machine parallèle. Cela permet d'exécuter en parallèle des requêtes relationnelles et par conséquent d'augmenter les performances.

1.4.5 Bases de données cloud

Dans ce cadre, elle est optimisée ou directement créée pour les environnements virtuels. Il peut s'agir d'un Cloud privé, d'un Cloud public ou d'un Cloud hybride. Les bases de données Cloud offrent plusieurs avantages comme la possibilité de payer pour la capacité de stockage et la bande passante en fonction de l'usage. Par ailleurs, il est possible de changer l'échelle sur demande. Ces bases de données offrent aussi une disponibilité plus élevée.

1.5 Historique des SGBD

Les premiers SGBD sont réellement apparus à la fin des années 60. La première génération de SGBD est marquée par la séparation de la description des données et de la manipulation par les programmes d'application. Elle coïncide aussi avec l'avènement des langages d'accès navigationnels. Cette génération a été dominée par les SGBD TOTAL, IDMS, IDS 2 et IMS 2. Elle traite encore aujourd'hui une partie importante du volume de données gérées par des SGBD.

La deuxième génération de SGBD a grandi dans les laboratoires depuis 1970, à partir du modèle relationnel. Elle vise non seulement à enrichir, mais aussi à simplifier le SGBD externe afin de faciliter l'accès aux données pour les utilisateurs. En effet, les données sont présentées aux utilisateurs sous forme de relations entre domaines de valeurs, simplement représentées par des tables. Les recherches et mises à jour sont effectuées à l'aide d'un langage non procédural standardisé appelé SQL (Structured Query Language). Celui-ci permet d'exprimer des requêtes traduisant directement des phrases simples du langage naturel et de spécifier les données que l'on souhaite obtenir sans dire comment y accéder. C'est le SGBD qui doit déterminer le meilleur plan d'accès possible pour évaluer une requête. Cette deuxième génération reprend, après les avoir fait évoluer et rendus plus souples, certains modèles d'accès de la première génération au niveau du SGBD interne, afin de mieux optimiser les accès. Les systèmes de deuxième génération sont commercialisés depuis 1980. Ils représentent aujourd'hui l'essentiel du marché des bases de données. Les principaux systèmes sont ORACLE, INGRES, SYBASE, INFORMIX, DB2 et SQL SERVER. Ils supportent en général une architecture répartie, au moins avec des stations clients transmettant leurs requêtes à de puissants serveurs gérant les bases.

La troisième génération a été développée dans les laboratoires depuis le début des années 80. Elle commence à apparaître fortement dans l'industrie avec les extensions objet des systèmes relationnels. Elle supporte des modèles de données extensibles intégrant le relationnel et l'objet, ainsi que des architectures mieux réparties, permettant une meilleure collaboration entre des utilisateurs concurrents. Cette troisième génération est donc influencée par les modèles à objets, intégrant une structuration conjointe des programmes et des données en types, avec des possibilités de définir des sous-types par héritage. Les systèmes objet-relationnels tels Oracle 8, DB2 Universal Database ou Informix Universal Server sont les premiers représentants des systèmes de 3e génération. Tous ces systèmes tentent de répondre aux besoins des nouvelles applications (multimédia, Web, bureautique, télécommunications, etc.).

Quant à la quatrième génération, elle est déjà en marche et devrait mieux supporter Internet et le Web, les informations mal structurées, les objets multimédias, l'aide à la prise de décisions et l'extraction de connaissances à partir des données. Finalement, l'évolution des SGBD peut être perçue comme celle d'un arbre, des branches nouvelles naissant, mais se faisant généralement absorber par le tronc, qui grossit toujours d'avantage.

1.6 Architecture d'un SGBD

Un SGBD est composé de trois niveaux qui autorisent la manipulation de données, garantissent l'intégrité des données et optimisent l'accès aux données.

1.6.1 Niveau interne (physique) :

Il définit la façon selon laquelle sont stockées les données et les méthodes pour y accéder. Il regroupe les services de gestion de la mémoire secondaire. Il s'appuie sur un système de gestion de fichiers pour définir la politique de stockage ainsi que le placement des données. Cette politique est définie en fonction des volumes de données traitées, des relations sémantiques entre les données ainsi qu'en fonction de l'environnement matériel disponible. Il est tout à fait possible de répartir les données sur différents supports de stockages distribués sur un réseau. Le niveau physique est donc responsable du choix de l'organisation physique des fichiers ainsi que de l'utilisation de telle ou telle méthode d'accès en fonction de la requête. Ce niveau doit également assurer le partage des ressources, la gestion de la concurrence et des pannes.

1.6.2 Niveau conceptuel :

Il définit l'arrangement des informations au sein de la base de données. Il correspond à la vision des données générale indépendante des applications individuelles et de la façon dont les données sont stockées. Dans le cas des SGBD relationnels, il s'agit d'une vision tabulaire où la sémantique de l'information est exprimée en utilisant les concepts de relation, attributs et des contraintes d'intégrité. Le niveau conceptuel est défini à travers le schéma conceptuel.

1.6.3 Niveau externe :

Il définit les vues des utilisateurs. Il représente l'interface entre le SGBD et les utilisateurs, il regroupe toutes les possibilités d'accès aux données par les différents utilisateurs. Ces accès sont éventuellement distants,ils peuvent se faire via différents types d'interfaces et langages plus ou moins élaborés. Ce niveau détermine le schéma externe qui contient les vues des utilisateurs sur la base de données c'est à dire le sous-ensemble de données accessibles ainsi que certains assemblages d'information et éventuellement des informations calculées.

La figure suivante résume ce qu'a été exprimé précédemment :

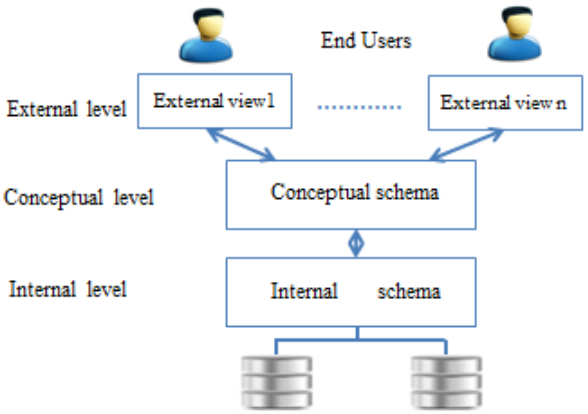


FIGURE 1.3 – Architecture d'un SGBD

1.7 Composantes d'un SGBD

Un SGBD est un composant software qui se compose de plusieurs éléments, la figure 1.4 illustre l'architecture générale d'un système de gestion de base de données typique. Cette architecture montre les principales composantes d'un SGBD, qui sont : le compilateur (DDL, DML), l'optimiseur de requêtes, la gestion de transaction et le gestionnaire de stockage.

1.7.1 Compilateur :

Il constitue l'interface entre l'utilisateur et le moteur d'exécution. Il permet de vérifier la syntaxe des requêtes avant de les diriger vers le moteur d'exécution. Le compilateur DDL permet de créer et de modifier l'organisation des données dans la BD, c'est-à-dire définir les schémas (les tables, les attributs et les contraintes). Le compilateur DML permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les BD.

1.7.2 Optimiseur des requêtes :

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).

1.7.3 Gestion de transaction :

Une transaction est une unité de mise à jour composée de plusieurs opérations successives qui doivent être toutes exécutées ou pas du tout. Le SGBD doit garantir les fameuses propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité) des transactions. Outre l'atomicité mentionnée, les transactions effectuent des accès concurrents aux données qui doivent être contrôlés afin d'éviter les conflits entre lecteurs et écrivains. Cela nécessite d'isoler les mises à jour dont les effets doivent par ailleurs être durables. En conséquence, la gestion de transactions mélange

intimement les problèmes de fiabilité et de reprise après panne avec les problèmes de concurrence d'accès. Les problèmes de sécurité qui recouvrent la confidentialité sont aussi connexes. Une transaction est donc composée d'une suite de requêtes dépendantes à la base qui doivent vérifier les propriétés d'atomicité, de cohérence, d'isolation et de durabilité, résumées par le vocable ACID.

- **Atomicité :** Une transaction doit effectuer toutes ses mises à jour ou ne rien faire du tout. En cas d'échec, le système doit annuler toutes les modifications qu'elle a engagées. L'atomicité est menacée par les pannes de programme, du système ou du matériel, et plus généralement par tout événement susceptible d'interrompre une transaction en cours.
- **Cohérence :** La transaction doit faire passer la base de données d'un état cohérent à un autre. En cas d'échec, l'état cohérent initial doit être restauré. La cohérence de la base peut être violée par un programme erroné ou un conflit d'accès concurrent entre transactions.
- **Isolation :** Les résultats d'une transaction ne doivent être visibles aux autres transactions qu'une fois la transaction validée, afin d'éviter les interférences avec les autres transactions. Les accès concurrents peuvent mettre en question l'isolation.
- **Durabilité :** Dès qu'une transaction valide ses modifications, le système doit garantir qu'elles seront conservées en cas de panne. Le problème essentiel survient en cas de panne, notamment lors des pannes disques.

1.7.4 Gestionnaire de stockage :

Il permet d'effectuer les lectures et les écritures sur les données persistantes sur les supports de stockages, dans le but de répondre aux requêtes traitées par le moteur d'exécution.

La figure suivante illustre ses principales composantes.

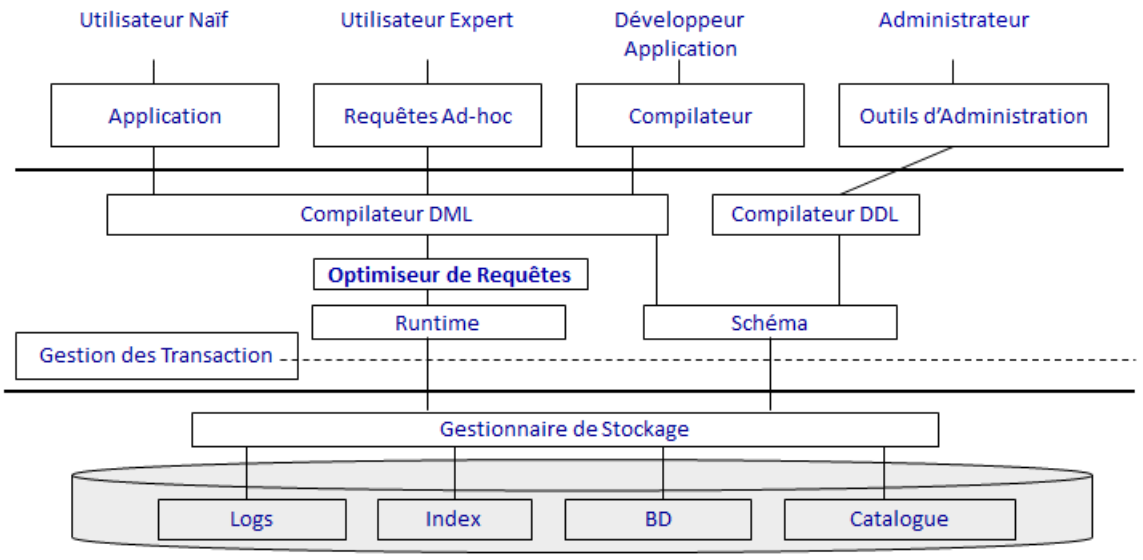


FIGURE 1.4 – Composantes d'un SGBD

Il est composé aussi de nombreux programmes, parmi lesquels le moteur, le catalogue, le processeur de requêtes, le langage de commande et un ensemble d'outils.

— **Le moteur de base de données :** est le cœur du SGBD, il manipule les fichiers de la base de données, transmet les données depuis et vers les autres programmes, et vérifie la cohérence et l'intégrité des données.

— **Le catalogue :** est le magasin qui contient la description de l'organisation de la base de données, les listes de contrôle d'accès, le nom des personnes autorisées à manipuler la base de données et la description des règles de cohérence (contraintes). Selon les modèles de SGBD ces informations peuvent être modifiées en utilisant le langage de commande, ou alors à l'aide d'une interface graphique.

— **Le processeur de requête :** exécute les opérations demandées. Selon les modèles de SGBD, ces opérations peuvent être formulées dans un langage de commande, ou à l'aide d'une interface graphique du type QBE (Query by Example, en français requête par l'exemple).

— **Les outils du SGBD :** servent à créer des comptes rendus (reports), des écrans pour la saisie des informations, importer et exporter les données vers la base de données, et manipuler le catalogue. Ces outils sont utilisés par l'administrateur de bases de données (DBA) pour effectuer des sauvegardes, des restaurations de données, autoriser ou interdire l'accès à certaines informations, et effectuer des modifications du contenu de la base de données -création, lecture, modification et suppression d'informations, abrégé *C*RUD (anglais *c*reate, *r*ead, *u*pdate, *d*elete). Ces outils servent également à surveiller l'activité du moteur et effectuer des opérations de tuning.

1.8 Synthèse sur la formulation des requêtes SQL et leur optimisation

Suite à la grande évolution des modèles conceptuels (ER, UML, Semantic), logiques (Relationnel) , physiques (column store, Row store), les plateformes de déploiement et la complexité des requêtes SQL, la conception des systèmes de gestion des bases de données est devenue plus difficile et complexe. Ainsi, suite à l'évolution de la technologie informatique, la conception de la BD est l'autre difficile. Cependant, de nombreux praticiens n'ont aucune formation en conception de bases de données plus particulièrement au niveau de la phase physique et ne sont pas familiarisés avec les techniques d'optimisation des requêtes SQL et le déploiement de bases de données.

Nous explorons le champ d'optimisation physique, nous trouvons plusieurs travaux qui ont mis l'accent sur le cycle de vie actuel de la BD, ils se déroulent sur les trois phases principales suivantes : la phase conceptuelle (PC), la phase logique (PL) et la phase physique (PP), parmi ces travaux il y a ceux qui ont proposé un processus de conception des BD automatique via un générateur de code qui prend en compte les dépendances entre la conception des trois phases citées précédemment [1] et qui augmente les interdépendances entre les phases du cycle de vie [2] [3].

D'autres travaux correspondent aussi à la couche physique on cite ceux de l'apprentissage du langage SQL, qui visent d'aider les étudiants à apprendre le SQL et les enseignants dans la présentation de leurs cours. Dans ce sens, plusieurs Frameworks ont été développés comme ceux de Huda Shuaily et al. [8], de Kristin Annabel et al. [9], de Alberto Abello et al.[10], et de Andrew Yost et al. [11]. Il y a également des travaux qui ont mis le point sur la complexité de l'optimisation de requêtes (pour plus de détail voir [7]).

L'optimisation des requêtes prend en compte celle basée sur le coût, cette dernière repose sur l'ordre de sélectivité et l'énumération des jointures. Les travaux qui ont focalisé sur l'optimisation des requêtes sont nombreux comme les travaux de Surajit Chaudhuri [12], de Saurabh Gupta et al. [13] et de Herodotos Herodotou et al[14]. En revanche, les travaux cités ont mal concentré sur l'apprentissage de cette expérience par l'ensemble d'apprenants, ils n'ont pas pensé sur la réflexion pour faciliter cette connaissance.

La figue suivante représente une carte conceptuelle sur l'ensemble de travaux cités auparavant.

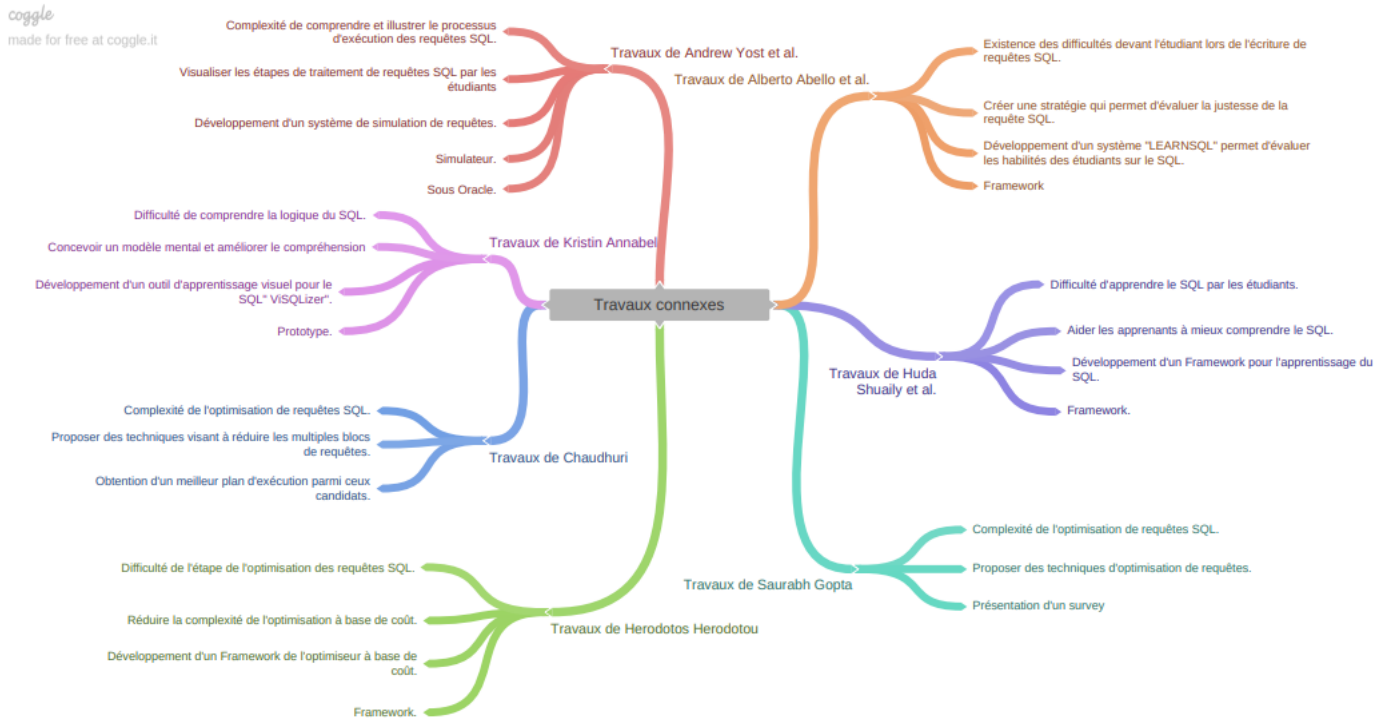


FIGURE 1.5 – Synthèse de quelques travaux de l'optimisation des requêtes SQL.

1.9 Conclusion

Dans ce premier chapitre nous avons présenté des concepts et des généralités sur les bases de données et les systèmes de gestion de base de données, nous avons vu les différentes phases qui constituent le cycle de vie de la BD, ses types, et un ensemble de concepts liés au SGBD tels que son architecture, et ses composantes. Nous avons cité quelques outils assistants durant le cycle de vie de la BD et nous avons mis le point sur quelques travaux qui concerne l'écriture de requêtes SQL et leur optimisation.

Les informations et les données requises à partir de ce chapitre nous aident énormément à entamer notre travail concernant le chapitre suivant sur le processus d'exécution des requêtes SQL et notamment l'optimisation des requêtes.

Chapitre 2

Optimisation des requêtes SQL

« Life is precious as gold that loses its time loses a treasure ... »
—Henri Loevenbruck

2.1 Introduction

Les analyses de données font partie du quotidien des professions ; ces données sont stockées dans un dépôt (BDD) qui représente une unité essentielle et cœur des systèmes d'information des entreprises.
Les bases de données relationnelles conçues suivant le modèle relationnel défini par Edgar Codd dont les fondations théoriques sont solides, et manipulées en utilisant l'algèbre relationnelle pour pouvoir organiser et accéder à l'ensemble de données ; cependant, il est difficile de modéliser un domaine directement sous une forme de base de données relationnelle. Une modélisation intermédiaire est généralement indispensable, Le modèle entités-associations (E/A) ou le diagramme de classe UML permettent une description naturelle du monde réel à partir des concepts du modèle E/A.
Sur le plan pratique la base de données peutêtre créée et interrogée à partir d'un langage opérationnel, ce dernier est considéré comme un langage d'accès normalisé à la base de données relationnelle qui permet les recherches et les mises à jour. Ce langage s'appelle SQL « Structured Query Lanquage » qui sert à exploiter des bases de données à travers les parties qui les composent : (1) LMD « Langage de Manipulation de Données (2) LDD « Langage de définitions de données » (3) LCD « Langage de Contrôle de Données ».

2.2 Processus d'exécution des requêtes SQL

Le traitement et l'optimisation des requêtes SQL ont toujours été l'un des éléments importants de la technologie des bases de données. Ces composantes traitent 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.
Une requête SQL est soumise dans son traitement à un ensemble enchainé d'étapes afin d'obtenir les résultats souhaités. La figure 4.1 présente le processus classique de traitement des requêtes qui comprend les phases d'analyse, d'optimisation, de génération de code, et d'exécution. En général, le processus s'exécute en deux étapes : une étape de compilation qui recouvre souvent les phases de l'analyse jusqu'à la génération de code, et une étape d'exécution. Les fonctions principales du traitement de requêtes sont :

2.2.1 Analyse de requête

Une requête exprimée dans un langage déclaratif tel que SQL [DD93], est analysée syntaxiquement et sémantiquement. Le résultat de cette phase est une représentation interne de la requête souvent sous forme d'un arbre algébrique (*plan de requête*) dont les noeuds sont les opérateurs comme la sélection, la projection, la jointure, etc.., et les arcs représentent la dépendance de données entre les noeuds.

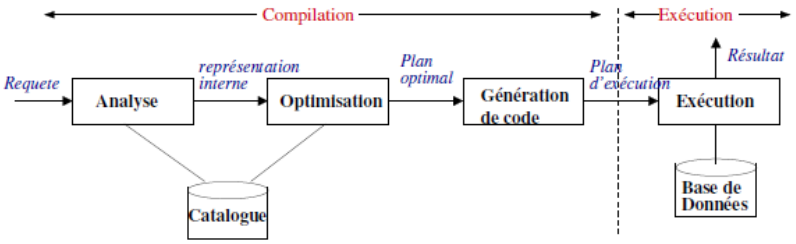


FIGURE 2.1 – Processus de traitement des requêtes.

2.2.1.1 Plan de requête

Un plan de requête représente un schéma pas à pas des opérations d'accès aux données d'un système de gestion de base de données SQL, ce plan constitue d'un ensemble de nœuds correspondant à un ensemble d'opérateurs algébriques. Dans le plan logique des requêtes, chaque nœud est associé à un algorithme qui sera utilisé pour évaluer l'opérateur algébrique relationnel correspondant (voir figure 4.2), qui peut être implémenté par plusieurs façons.
La plupart des systèmes de SGBD commerciaux ont mis en oeuvre au moins :

— Trois variantes des algorithmes de jointure : jointure à boucles imbriquées, jointure tri-fusion et jointure de hachage hybride.

— Deux variantes de scan (balayage séquentiel et indexé).

— Variante de tri comme (radix sort, bitonic merge sort).

— Variante de d'agrégation (hash-based, sort-based).

Nous fournissons un exemple de code dans l'Algorithme 1 qui démontre l'algorithme de jointure par boucles imbriquées.

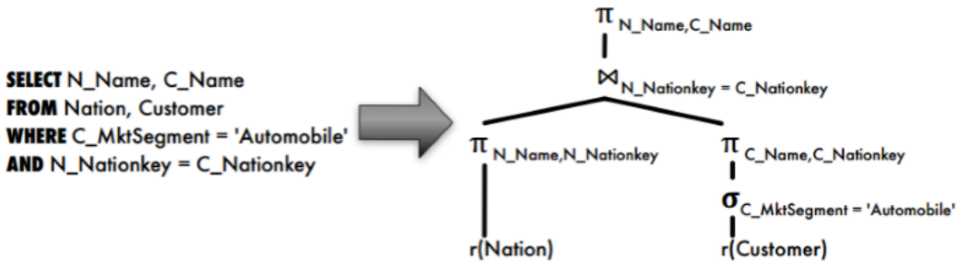


FIGURE 2.2 – Exemple de plan de requête SQL.

Algorithm 1 Algorithm of block-nested-loops join.

Input : - Relation `A`, JoinAttribute `jA`, Relation `B`, JoinAttribute `jB`;
Output : Relation `C`.

```
1  // Load as many pages as possible begin
2    foreach Block aBlock in A do
3      // Load page by page block size = page size
4      foreach Block bBlock in B do
5        foreach Tuple a in aBlock do
6          foreach Tuple b in bBlock do
7            if a.jA = b.jB then
8              //Keys match join tuples
               c := join(a,b) C.insert(c);
9            end
10         end
11       end
12     end
13   end
14   return C
15 end
```

FIGURE 2.3 – Algorithme de jointure par deux boucles imbriquées

2.2.2 L'optimisation de plan de requête

Étant donné une requête, il existe plusieurs expressions de calcul équivalentes. Ces expressions sont examinées afin de choisir la meilleure, appelée plan optimal [4], ce dernier sera évalué par la suite afin de répondre à la requête d'origine.

2.2.3 Génération de code

L'objectif de cette phase est de générer les appels des méthodes, les codes pour pouvoir évaluer le plan optimal. Le résultat de cette phase est le code exécutable, appelé *plan d'exécution.*

2.2.4 Execution du plan d'execution

Les plans générés par la phase précédente sont exécutés pour produire les résultats de la requête d'origine.
La phase d'analyse est considérée comme l'interface entre le système et l'utilisateur. La fonction

4. Nous utilisons le terme plan optimal pour désigner une (meilleure) solution retenue sans forcément être la solution la plus optimisée.

d'analyse dépend du langage déclaratif utilisé. En revanche, la phase de génération de code dépend de l'implémentation du système, des opérateurs, des fonctions, etc. Quant à l'optimisation et à l'exécution, elles exploitent l'aspect ensembliste des données en utilisant les techniques plus ou moins complexes pour garantir l'efficacité du traitement des requêtes. Ces deux aspects sont les fonctions les plus importantes dans les évaluateurs de requêtes sur lesquelles la suite de ce chapitre se focalise.

2.3 Optimisation des requêtes SQL

L'optimisation de la requête est le processus général qui consiste à choisir le moyen le plus efficace d'exécuter une instruction SQL, son objectif est de trouver le plan optimal des requêtes en entrée (ou requêtes utilisateurs). Il s'agit de réécrire la requête, si nécessaire, de choisir l'ordre d'exécution des opérateurs de la requête et les algorithmes implémentant ces opérateurs.
Le choix du plan optimal s'appuie souvent sur une estimation de coût en exploitant les connaissances sur les données interrogées et sur l'environnement d'exécution. La figure 2.4 montre la vue globale du processus d'optimisation qui se compose de deux étapes : une pour la *réécriture* et l'autre pour la *planification* [15].

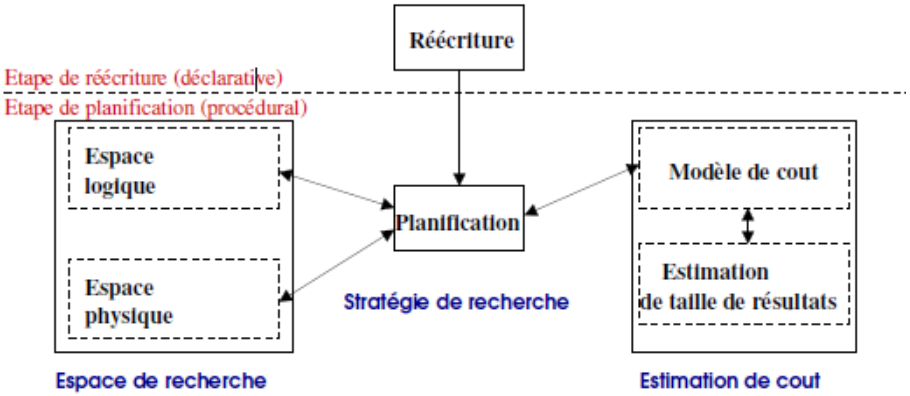


FIGURE 2.4 – Architecture de l'optimiseur.

2.3.1 Étape de réécriture :

C'est une étape déclarative, vise à transformer la requête en entrée en une ou plusieurs requête(s) équivalente(s) de sorte à obtenir une représentation canonique tout en générant un arbre logique. Elle s'appuie sur les déclarations statiques (e.g. la définition des vues) et sur les caractéristiques de la requête (e.g. l'imbrication des (sous-)requêtes), sans considérer les caractéristiques spécifiques de l'implémentation du SGBD.
Cette phase comporte un aspect sémantique pouvant aller jusqu'à la prise en compte des règles d'intégrité, et un aspect syntaxique concernant le choix d'une forme canonique, celle-ci comprend la mise sous forme normale des critères et l'affectation d'un ordre fixé pour les opérateurs algébriques.

2.3.2 Étape de planification :

C'est une étape procédurale, elle est susceptible d'examiner les plans équivalents et de choisir le meilleur plan qui sera exécuté par la suite. Cette phase a pour rôle d'ajouter les annotations à l'arbre logique obtenu précédemment et ça, pour avoir un plan d'exécution. Elle s'appuie sur un modèle de coût pour minimiser le temps nécessaire à l'exécution d'un arbre. La planification peut être décomposée en trois éléments principaux qui sont : (i) *l'espace de recherche*, (ii) *l'estimation du coût* et (iii) *la stratégie de recherche.*

Le but des ces deux phases précédentes est de générer un arbre optimal et de choisir les meilleurs algorithmes pour exécuter chaque opérateur et l'arbre dans son ensemble. Pour cela, il faut optimiser simultanément :
- le nombre d'entrées-sorties ;
- le parallélisme entre les opérations ;
- le temps de calcul nécessaire.

Dans ce qui suit ; en présente les trois éléments de planification :

2.3.2.1 *Espace de recherche :*

Étant donné une requête, il peut exister plusieurs plans équivalents (voir figure 2.5) , chacun représentant une façon d'évaluer la requête que le système peut considérer. Ces plans constituent l'espace de recherche de la requête composée de l'espace logique et de l'espace physique.
La génération de cet espace, étant une des tâches de l'optimiseur [5] ,il se base sur les propriétées algèbriques(e.g. la commutativité, l'associativité) pour l'espace logique et utilise les propriétés physiques de l'implémentation (e.g. les index, les algorithmes) pour l'espace physique. Elle consiste donc à appliquer successivement différentes opérations sur le(s) plan(s) initial et intermédiaires[15].

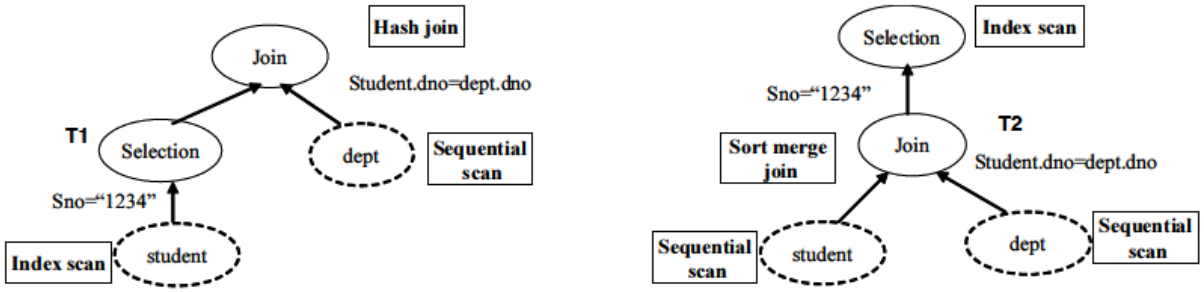


FIGURE 2.5 – Plans d'exécution équivalents.

— **Probléme de sélection de plan optimal :** Une requête SQL est souvent représentée sous forme d'un ou plusieurs arbres algébriques dont les feuilles sont des relations et les non-terminaux des opérateurs.
L'optimisation consiste à trouver un meilleur plan (plan optimal) d'une requête SQL parmi un ensemble de plans candidats et selon des critères et des statiques spécifiés, ce qui cause le problème de selection d'un arbre optimal qui sera défini par : - pour n operateurs, le nombre de plan d'exécution possible est égale $n!$ dans sa complexité égale à $O(n!)$. Le

5. logiciel de base de données intégré qui détermine la méthode la plus efficace pour une instruction SQL d'accéder aux données demandées.

problème de génération du plan d'exécution optimal devient alors NP-complet [16].

Le problème de selection de plan optimal peut être formalisé comme suit :

- Soit Q une requête à optimiser :
- Procédure :
  1. Enumérer tous les plansP1,...,Pm pour chaque requête(notons que chaque requête possède un ensemble d'opérations O1,...,Ok)
  2. Pour chaque plan Pj
     - Pour chaque opération Oi du plan Pj, énumérer les routines d'accès
     - Sélectionner la routine ayant le coût le moins élevé.

     Alors nous pouvons définir les fonctions de coût pour chaqunes d'elles comme suit :

$$\textbf{Coût}(P_i) = \sum_{i=1}^{k} \min(O_i) \tag{2.1}$$

$$\textbf{Coût}(Q) = \sum_{h=1}^{m} Cout(P_h) \tag{2.2}$$

2.3.2.2 *Estimation de coût :*

Le choix du meilleur plan d'une requête se base souvent sur l'estimation de coût des plans dans son espace de recherche. Le coût peut s'exprimer en terme de temps d'exécution (du lancement de l'exécution de la requête jusqu'à l'obtention de son résultat), de consommation des ressources (e.g. communication, mémoire, CPU, etc.) ou encore le coût économique. La mesure du coût reflète effectivement l'objectif d'optimisation dont les plus courants sont de minimiser, soit la consommation de ressources, soit le temps de réponse.
D'une manière générale, l'estimation de coût s'appuie sur un modèle de coût dont les principaux facteurs à prendre en considération sont (i) le coût de communication, et (ii) le coût du traitement de données (à distance et locale). Ces coûts dépendent aussi de la taille et de la cardinalité des données traitées (i.e. données en entrée et intermédiaires) qui sont estimées par les formules mathématiques en se basant sur les informations nécessaires [6] stockées dans le catalogue (i.e. méta-données) du système[15].

2.3.2.3 *Stratégie de recherche :*

La stratégie de recherche spécifie la manière dont l'optimiseur examine l'espace de recherche, ces stratégies sont utilisées par l'optimiseur pour explorer l'espace des plans d'exécution afin de déterminer un plan de coût proche du minimum possible. On trouve deux types de stratégies qui sont :

- ***Stratégies énumératives :*** Cette stratégie possède deux sous stratégies dont la première est une *stratégie exhaustive* et la deuxième *stratégie par augmentation.*

6. e.g. la taille et la cardinalité des données en entrée, la sélectivité, etc.

— ***Stratégies aléatoires :*** Cette stratégie explore aléatoirement l'espace des plans, elle contient d'autres sous stratégies dans lesquelles nous trouvons :

  — ***amélioration itérative :*** elle tire au hasard n plans et essaie de trouver pour chacun d'eux le plan optimal le plus proche (par exemple par descente des restrictions et projections, et choix des meilleurs index).L'optimum des plans sera finalement sélectionné.
  — ***Stratégies simulées :*** Elles procèdent à partir d'un plan que nous tentons de l'optimiser en appliquant des transformations successives. Les transformations retenues améliorent ce plan exceptées quelques-unes introduites afin d'explorer un espace plus large avec une probabilité variante.

  — ***Stratégies génétiques :*** Elles visent à fusionner deux plans pour obtenir un troisième.

La figure 2.6 illustre ces différentes stratégies :

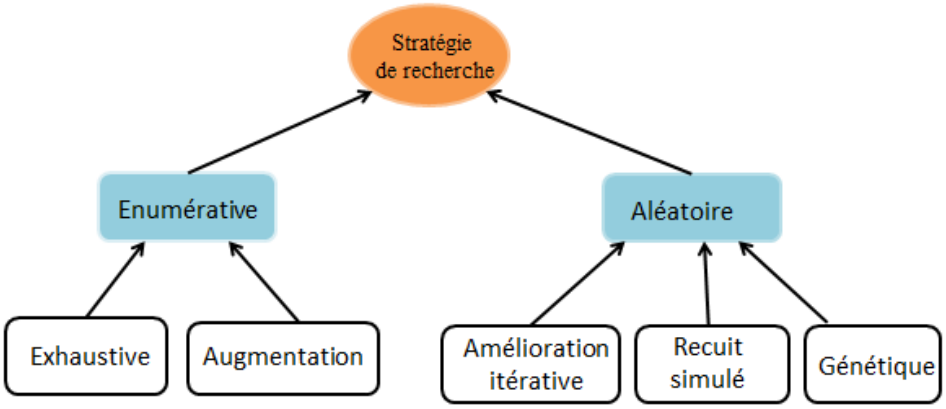


FIGURE 2.6 – Principale stratégie de recherche.

Plusieurs stratégies ont été proposées sous forme d'algorithmes d'optimisation, dont la plupart regroupent aussi les aspects liés à la génération de l'espace de recherche et à l'estimation de coût. A titre d'exemple, nous pouvons citer les algorithmes basés sur la programmation dynamique, les algorithmes basés sur le parcours aléatoire, et ceux basés sur les règles.

L'objectif de ces algorithmes est de rechercher le plan optimal de la requête utilisateur en réduisant le coût de la phase d'optimisation [7] dans le but d'améliorer la performance globale de l'évaluation de requête. Ces algorithmes se différencient par leur complexité (polynomial vs combinatorial) et/ou par la nature du parcours de l'espace de recherche (déterministe vs aléatoire, heuristique vs systématique, constructive vs transformative).

7. Ce coût peut être mesuré par le temps de l'optimisation et/ou par les ressources nécessaires pour l'optimisation, etc.

Ces algorithmes peuvent être regroupés en deux grandes approches, une basée sur la construction de plan (*bottom-up*) et l'autre appuyée sur la transformation de plan (*top-down*) comme sillustre la figure 2.7. Nous présentons ci-dessous le principe de ces deux approches en introduisant quelques heuristiques qui pouvent être utilisées pour améliorer la recherche du plan optimal.

— ***Approche par construction (bottom-up) :*** Suivant cette approche, l'énumération de plans d'une requête débute par les plans les plus simples (i.e. les plans d'accès ou les méthodes d'accès de chacune des entrées de la requête). A partir de ces premiers plans, l'algorithme construit des plans de plus en plus complets en combinant ceux qui ont été construits précédemment (la figure 2.7 (a)).

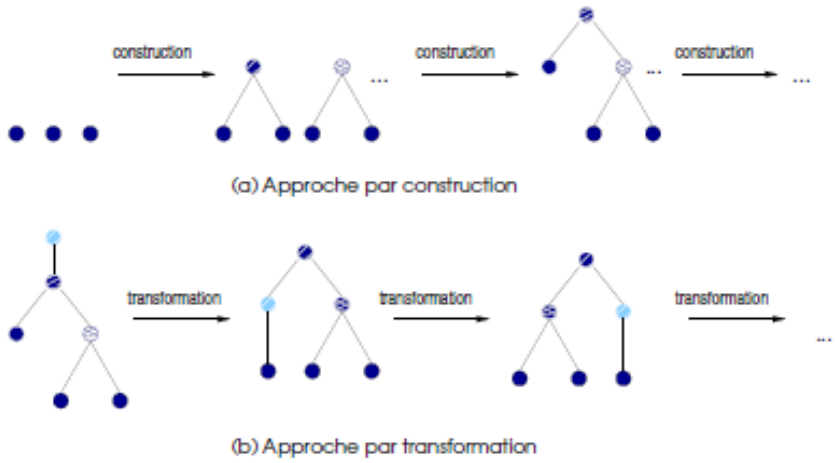


FIGURE 2.7 – Illustration des stratégies de recherche.

L'algorithme le plus représentatif de cette approche est proposé par P. Selinger et al [17], dans le contexte du système R. Le principe de cet algorithme est de réduire dynamiquement l'espace de recherche par l'itération sur le nombre des opérateurs de jointure du plan de requête. Par conséquent, l'optimisation s'effectue localement et elle ne peut pas garantir que le plan optimal retenu à la fin du processus soit le plan le plus optimisé. Cependant, elle permet de réduire le coût de l'optimisation en limitant le nombre des plans générés à chaque étape.

L'algorithme s'exécute en trois phases comme suit :

1. examiner les plans d'accès de chacune des entrées de la requête : tous les plans d'accès possibles sont énumérés et seul celui le *moins coûteux* de chaque entrée est conservée pour l'étape suivante.
2. (*Itération*) énumérer les plans en considérant toutes les jointures possibles entre deux entrées, puis les jointures de trois entrées, etc., jusqu'à l'obtention des jointures de n entrées de la requête à optimiser. A chaque étape, seulement les plans les moins coûteux sont conservés.
3. compléter les plans obtenus de la phase précédente avec les opérateurs de sélection, de projection, etc.

— ***Approche par transformation :*** A la différence de l'approche précédente, l'optimisation par transformation consiste à appliquer successivement les transformations possibles sur les plans complets de la requête afin d'énumérer tous les plans équivalents avant d'en choisir le meilleur pour son exécution. La figure 2.7 (b) illustre le principe de cette stratégie.

Les algorithmes de cette approche ont été proposés initialement dans le contexte de SGBD extensibles. Dans cette approche, les heuristiques consistent sur l'ordonnancement des transformations à appliquer. Quelques heuristiques les plus utilisées sont « sélection d'abord », « évitement les produits cartésiens. », « diminution des constituants » (i.e. réaliser les projections aussi tôt que possible) etc. Ces heuristiques visent à réduire la taille et la cardinalité des résultats intermédiaires.

D'autres algorithmes utilisent les heuristiques pour explorer l'espace de recherche par un parcours aléatoire (random walks). Nous trouvons différentes variantes de cet algorithme telles que Simulated Annealing, Iterative Improvement Two-Phase Optimization.

2.4 Les modes d'optimisation

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, l'optimiseur repose dans son traitement sur des approches principales plus utilisée (voir figure 2.8), et une approche secondaire rarement utilisée. La première approche se base sur un ensemble de règles, comme par exemple, descendre les sélections au plus bas du plan (Push Down Selection) afin de réduire la taille du résultat des opérations. Cette approche est désignée sous le nom d'approche dirigée par des règles (RBO : Rule-based Optimisation), celle-ci s'avérait insuffisante face à l'augmentation de la taille des schémas de base de données impliquant un nombre de plus en plus important de tables (relations) et à la diversité des algorithmes d'optimisation et des exigences.

L'autre approche dite dirigée par des modèles de coût (CBO : Cost-based Optimisation) elle est venue pour compléter la précédente. Cette approche vise à évaluer le coût des plans d'exécution afin de choisir celui ayant le coût minimal. La troisième approche est basée sur la réutilisation des plans d'exécution.

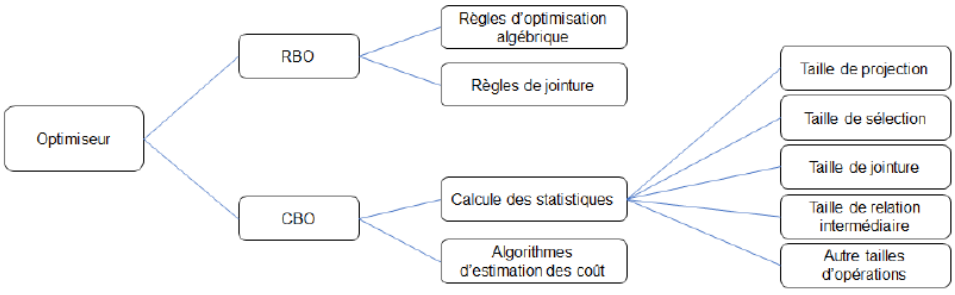


FIGURE 2.8 – Les deux modes d'optimisation.

Nous présentons ci-dessous un tableau qui récapitule ces trois approches avec une comparaison entres elles.

Stratégies d'optimisation	**Avantages**	**Inconvénients**
A base de règles	facile à mettre en œuvre	Néglige tous les paramètres physiques de la BD
A base de coût	considère la sélectivité de requêtes. fiable.	Nécessite les statistiques et les formules Elle est lente.
Réutilisation des plans d'exécution	Sélection des plans futurs. Valeur ajoutée à l'optimiseur. Décharger l'optimiseur classique. Fournir des plans d'exécution de qualité à terme des requêtes.	Travailler dans son plus haut niveau

TABLE 2.1 – Comparaison des stratégies de l'optimisation.

2.4.1 Optimiseur basé sur les règles RBO

Un optimiseur basé sur des règles est un optimiseur qui applique simplement un ensemble de règles à une instruction SQL au lieu de regarder des estimations de coûts, afin de déterminer le meilleur moyen d'exécuter cette instruction SQL.

2.4.2 Optimiseur basé sur les coûts CBO

Un optimiseur basé sur les coûts examinera toutes les manières possibles ou les scénarios dans lesquels une requête peut être exécutée et chaque scénario se verra attribuer un «coût» indiquant l'efficacité avec laquelle cette requête peut être exécutée. Ensuite, l'optimiseur basé sur les coûts choisira le scénario présentant le coût le plus bas et exécutera la requête à l'aide de ce scénario, car c'est le moyen le plus efficace d'exécuter la requête.

Les optimiseurs basés sur les coûts doivent utiliser certaines statistiques qu'ils collectent à partir de la base de données. Parmi les types de statistiques utilisées par les optimiseurs basés sur les coûts, citons le nombre de valeurs uniques d'une colonne indexée ou même le nombre de lignes d'une table.

Le CBO à un ensemble d'étapes à effectuer que allons décrire par ci-dessous.

2.4.2.1 Les étapes effectuées par CBO

Le CBO dans leur traitement effectue un ensemble d'étapes pour choisir le plan optimal attendu par l'utilisateur, ces étapes sont :

1. L'optimiseur génère un ensemble de plans potentiels pour l'instruction SQL en fonction des chemins d'accès disponibles et des hints

2. 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

proportionnellement 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, le CPU et la mémoire.

3. L'optimiseur compare les coûts des plans et choisit celui avec le coût le plus bas.

2.4.2.2 Les composants du CBO

Le CBO dans son traitement comprend trois principaux composants Transformateur de requête, Estimateur, Générateur de plan.

La figure 2.9 montre le fonctionnement de ces composants et la collaboration entres eux, au début une requête analysée (à partir de l'analyseur) est entrée dans le transformateur de requête. La requête transformée est ensuite envoyée à l'estimateur. Les statistiques sont extraites du dictionnaire, puis la requête et les estimations sont envoyées au générateur de plan qui va renvoyer le plan à l'estimateur ou envoyer le plan de requête au générateur de source de lignes.

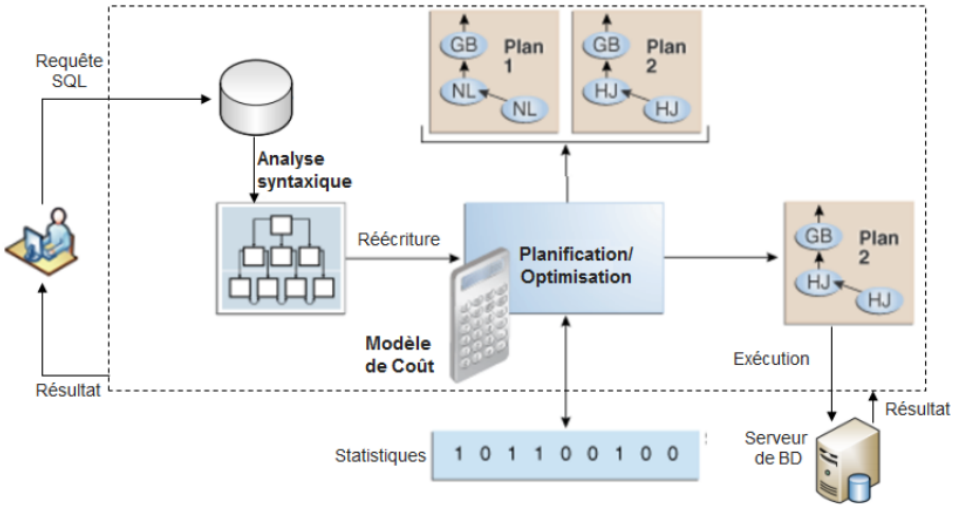


FIGURE 2.9 – Composants de l'optimiseur basés sur les coûts.

Nous détaillerons par la suite ces composants et nous montrons leur fonctionnement

1. ***Transformateur de requête :*** L'entrée du transformateur de requête est une requête analysée, qui est représentée par un ensemble de blocs de requêtes qui sont imbriquées ou liés entres eux. L'objectif principal du transformateur de requête est de déterminer s'il est avantageux de changer la forme de la requête afin qu'elle permet de générer un meilleur plan de requête.

2. ***Estimateur :*** c'est le composant essentiel du CBO qui détermine le coût global d'un plan d'exécution [8] donné, L'estimateur utilise trois mesures différentes pour déterminer le coût :

   — ***Sélectivité :*** C'est un pourcentage de lignes dans l'ensemble des lignes sélectionnées par la requête, La sélectivité est liée à un prédicat de requête, tel que, where ou une combinaison de prédicats qui devient plus sélective à mesure que la valeur de la sélectivité approche de 0 et moins sélective (ou plus non sélective) à mesure que la valeur se rapproche de 1.

   — ***Cardinalité :*** La cardinalité est le nombre de lignes renvoyées par chaque opération dans un plan d'exécution, cette entrée cruciale est commune à toute les fonctions de coût, elle est utilisée pour obtenir un plan optimal. L'estimateur peut déduire les cardinalités à partir des statistiques de table ou le déduire en tenant compte des effets de prédicats ou des opérations.

   — ***Coût :*** Le coût est une mesure numérique interne qui représente l'utilisation estimée des ressources pour un plan. Le coût est spécifique à une requête dans un environnement d'optimisation. Pour estimer le coût, l'optimiseur prend en compte des facteurs tels que : Ressources système, y compris les estimations d'E/S, de CPU et de mémoire, de nombre estimé de lignes renvoyées (cardinalité), taille des ensembles de données initiles.

3. ***Générateur de plan*** Le générateur de plans explore différents plans pour un bloc de requête en essayant différents chemins d'accès, méthodes de jointure et ordres de jointure.

2.5 Les structures d'optimisation

Il existe une large panoplie de structure d'optimisation supportée par les systèmes de gestion de base de données commerciaux, ces structures contenant plusieurs techniques d'optimisation qui ont été proposées dans la littérature, nous pouvons classer ces techniques en deux catégories principales : les techniques redondantes et les techniques non redondantes, la figure suivante (2.10) montre une classification des principales techniques d'optimisation.

8. Décrit une méthode d'exécution recommandée pour une instruction SQL.

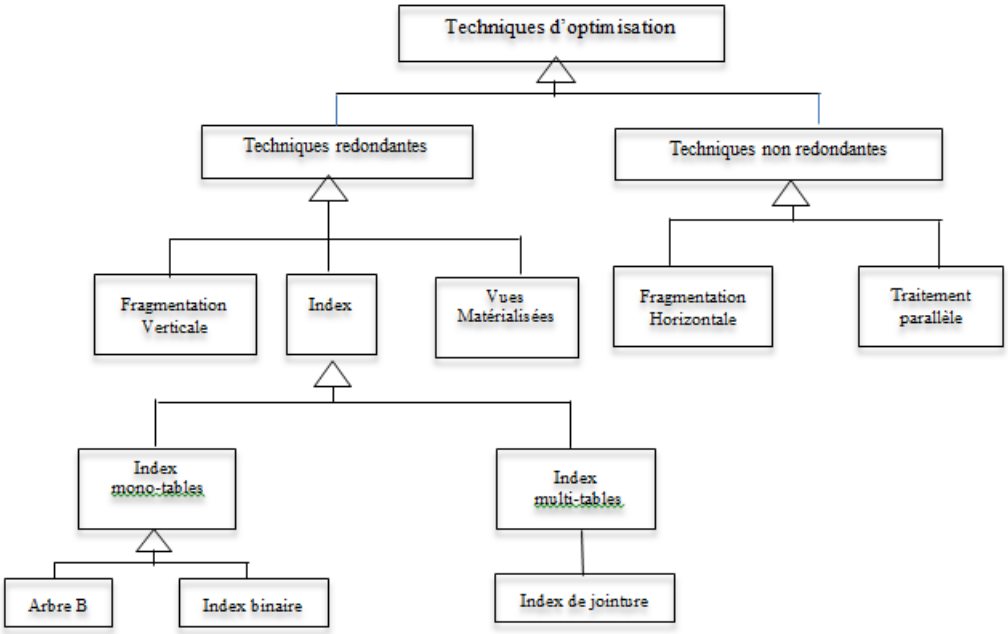


FIGURE 2.10 – Techniques d'optimisation.

2.5.1 Les techniques redondantes :

Les techniques redondantes optimisent les requêtes, mais exigent un coût de stockage et de maintenance. Cette catégorie regroupe les index, les vues matérialisées, la fragmentation verticale, etc.. . .

1. ***Les 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.

2. ***Les vues matérialisées*** Une vue matérialisée est une table contenant les résultats d'une requête. Les vues améliorent l'exécution des requêtes en pré-calculant les opérations les plus coûteuses comme la jointure et l'agrégation, 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 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. ***La fragmentation :*** La fragmentation est une technique, permettant l'optimisation de performances des requêtes et d'éviter le balayage de grandes tables, elle consiste à diviser une relation en plusieurs parties appelées fragment qui peuvent être non disjointes, dont le but est de réduire le temps d'exécution des requêtes. Nous distinguons deux type de fragmentation verticale qui représente une des techniques d'optimisation redondantes, est horizontalle comme techniques non redondantes.

   — ***Fragmentation verticale (VH) :*** La fragmentation verticale est l'une des techniques d'optimisation redondante, elle permet de diviser une relation (table) verticalement en plusieurs sous relations appelées fragments verticaux qui sont des projections appliquées à la relation, dans chacune va comporter un sous ensemble d'attributs de la relation initiale. La fragmentation verticale 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 à accéder. Son inconvénient est qu'elle requiert des jointures supplémentaires lorsqu'une requête accède à plusieurs fragments.

2.5.2 Les techniques non redondantes :

Les techniques d'optimisation non redondante n'impliquent pas une surcharge de stockage et de maintenance. Cette catégorie regroupe la fragmentation horizontale, le traitement parallèle, etc. . .

1. ***La fragmentation horizontalle (FH) :*** Est une technique d'optimisation considérant comme l'une des structures d'optimisation dans le cadre des bases de données relationnels. Cette fragmentation consiste à diviser une relation R en sous-ensembles de n-uplets appelés fragments horizontaux, chacun étant défini par une opération de restriction appliquée à la relation. Les n-uplets de chaque fragment horizontal satisfait une clause de prédicat. Nous distinguons deux types de fragmentation horizontale :

   — FH primaire définie sur une table de dimension en fonction de ses propres attributs.
   — FH dérivée définie sur la table des faits en fonction des dimensions fragmentées. La fragmentation dérivée est adaptée au contexte de bases de données relationnelles.

2. ***Le traitement parallèle :*** Le traitement parallèle des requêtes consiste à répartir leur déroulement dans l'espace (nœuds ou processeurs d'exécution) et dans le temps (ordonnancement) par une utilisation judicieuse des ressources disponible [18]. On cherche ainsi à réduire le temps de réponse moyen (satisfaction des utilisateurs) et le temps de réponse de chaque requête prise indépendamment (satisfaction de chaque utilisateur en particulier).

   Il existe différentes formes de parallélisme :
   — Parallélisme inter-requêtes.
   — Parallélisme intra-requêtes.
      • Parallélisme inter-opérateur.
      • Parallélisme intra-opérateur.

2.6 Les directives d'optimisation (Hint)

L'optimiseur généralement peut être subi par un tuning qui l'oblige à modifier le plan d'exécution, parmi les méthodes de ce tuning nous trouvons les hints, qui se plaçent dans la requête sous forme de commentaires afin d'obtenir les meilleures performances possibles. Nous commençons par la suite de présenter cette méthode.

2.6.1 Hint :

Les hints sont des instructions que nous pouvons insérer dans nos ordres SQL pour influencer l'optimiseur à choisir certainnes structures. Dans certains cas l'optimiseur peut ne pas prendre le meilleur chemin, du moins à notre goût. Nous pouvons alors l'influencer en insérant des hints dans l'ordre SQL.

2.6.2 Classification des hints

Le domaine d'optimisation a connu une grande variété de directives d'optimisation qui sont classées en plusieurs catégories :

- ***Hint pour les approches et objectifs d'optimisation :*** Les directives décrites dans cette section permettent de choisir entre les approches d'optimisation et les objectifs. Nous trouvons dans cette section :
  - All_ROWS.
  - FIRST_ROWS.
- ***Hint pour chemin d'accès :*** Chaque indication décrite dans cette section suggère un chemin d'accès pour une table. Nous trouvons dans cette section :
  - PLEIN.
  - GRAPPE.
  - HACHER.
- ***Hint pour operateur de jointe :*** Chaque indication décrite dans cette section suggère une opération de jointure pour une table. Nous trouvons
  dans cette section :
  - USE_NL.
  - USE_MERGE.
  - USE_HASH
- ***Hint pour les ordres de jointure :*** Les directives dans cette section suggèrent des ordres de jointure :
  - DE PREMIER PLAN.
  - COMMANDÉ.

2.7 Positionnement de notre solution

Dans le contexte des bases de données, le traitement des requêtes présente une étape sensible et très importante au même temps, en revanche ce processus implique des étapes corrélées et très complexes qui ont amplifié l'importance des optimisations des requêtes, celle-ci nous a ramené à chercher un moyen pour faciliter la compréhension et la manipulation de ce processus, dont le but de minimiser le coût de réponse des requêtes envoyées à la base de donnée et d'avoir des résultats avec une optimisation des performances efficaces (e.g coût de stockage, coût de maintenance).
Par exploration de la litterature, peu de travaux s'interessent à l'apprentissage et à l'enseignement des techniques de modelisation des bases de données(conceptuel, logique, physique, déploiement).
A notre connaissance il n'ya pas de travail qui traite la problématique et qui facilite la compréhension de l'optimiseur des requetes SQL dans un SGBD relationnel.
Notre solution qui sera présentée dans le chapitre 4 est considérée comme une initiative pour expliciter et conceptualiser l'optimiseur des requetes SQL d'un point de vue structurel et processural.

2.8 Conclusion

Dans ce chapitre nous avons présenté le processus général d'exécution d'une requête SQL ainsi que les différentes phases qui les constituent. Nous avons également mis le point sur la phase d'optimisation qui représente le cœur de ce processus en présentant et en classifiant les différentes techniques d'optimisation des requêtes. Dans le chapitre suivant nous allons présenter notre contribution qui concentre sur notre solution proposée.

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

3.1 Introduction

Après avoir présenté la partie état de l'art dans les deux chapitres précédents, nous avons remarqué que le processus d'exécution des requêtes SQL est trop difficile à comprendre, plus précisément l'étape de l'optimisation qui constitue une étape indispensable dans les SGBD.
Chaque travail de ceux que nous avons exprimé auparavant a donné une solution de sa guise pour aider les apprenants à mieux savoir le langage, et chaque travail à travaillé sur le domaine de l'optimisation de requêtes SQL par des diverses manières.

Dans ce chapitre nous allons illustrer notre solution et l'approche suivie pour l'atteindre, tout en basant sur des modèles qui vont nous amener à bien montrer cette solution, et aussi nous allons aborder le fondement théorique derrière cette approche.

3.2 Description de notre problème

Dans cette partie, nous décrivons les problèmes rencontrés au sein de la visualisation et la détermination du processus d'exécution des requêtes SQL, plus particulièrement la détermination de l'étape de l'optimisation de requêtes SQL (mode CBO).

Notre problème dans sa globalité s'inscrit sur la difficulté de comprendre et d'illustrer le processus d'exécution des requêtes SQL (écriture de la requête SQL, son optimisation, utilisation des hints et la visualisation du plan d'exécution d'une requête SQL donnée), ce problème a porté un ensemble de sous problèmes que nous avons essayé de les traiter tous afin d'avoir une meilleure compréhension de ce processus. Nous distinguons trois sous problèmes qui sont :

— ***Formulation de requête SQL :*** elle porte sur la nécessité d'assister une formulation d'une requête SQL à partir d'une requête écrite en langage naturel et c'est à l'apprenant à pouvoir la formuler en SQL, ce qui présente une difficulté car cette transformation nécessite un ensemble de règles à suivre pour avoir une expression SQL bien exprimée équivalente à celle écrite en langage naturel.

— ***Visualisation du processus de requêtes :*** La visualisation de ce processus représente une étape ambigue avec l'absence des outils qui montrent le traitement réel de ce processus, ainsi que la difficulté d'exprimer ses étapes qui comportent beaucoup de détails ; ces étapes nécessitent des recherches approfondies afin de les bien maitriser. Ce processus nécessite une vision plus détaillée de chacune de ses étapes pour bien les comprendre et les visualiser.

— ***Amélioration des performances des requêtes SQL :*** Ce sous problème porte sur la réécriture de la requête SQL pour influencer le plan d'exécution de requête SQL et forcer l'optimiseur à choisir un plan optimal par l'utilisation des hints (directives), et faire exploiter ces dernies nécessitent l'implémentation des algorithmes.

La figure suivante illustre les points à considérer lors de compréhension du processus d'exécution des requêtes SQL.

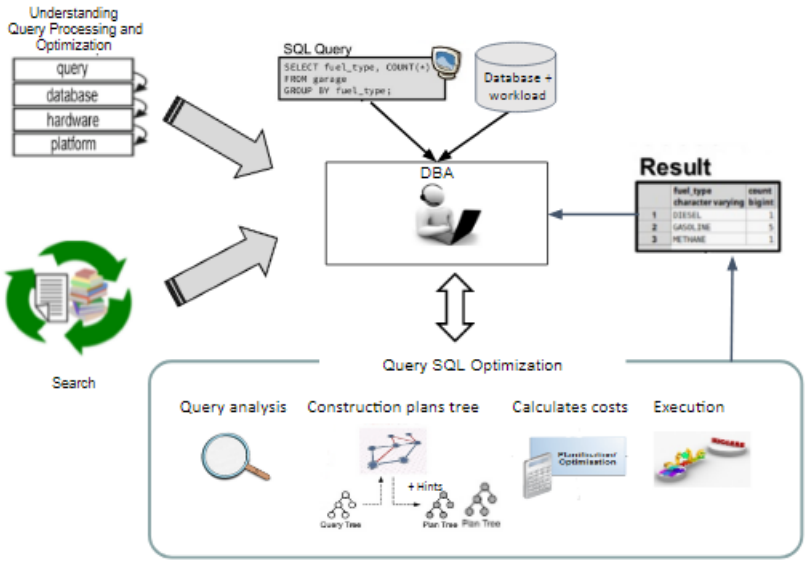


FIGURE 3.1 – Illustration de difficulté du processus de l'optimisation par un utilisateur.

3.3 Exemple de motivation

Considérons un scénario réel dans lequel un administrateur de base de données (DBA) souhaite optimiser une requête SQL. Un optimiseur de requête typique s'appuie sur des informations sur le matériel et sur des métadonnées pour trouver un plan d'exécution de moindre coût.

Dans la figure ci-dessous nous décrivons un exemple tiré du schéma de référence TPC-H (TPC-H benchmark schema) [9], dans lequel les DBA doivent optimiser le temps de réponse total d'une requête en prenant en compte leurs coûts d'E/S et du processeur.

Le système en cours de conception est décrit par plusieurs paramètres liés à la base de données, aux requêtes et à la plateforme. Les paramètres de la BD sont les suivants : schéma relationnel, stockage orienté ligne, aucune compression de données et stratégie de traitement de

9. https ://docs.snowflake.net/manuals/user-guide/sample-data-tpch.html

requêtes en pipeline. Les paramètres de requêtes considérés sont : type de requêtes OLAP sans exécution simultanée. La plateforme a comme paramètres : architecture de déploiement centralisée, mémoire principale en tant que périphérique de stockage principal et disque dur comme un périphérique de stockage secondaire. Le DBA peut utiliser des directives (hints) et des structures d'optimisation pour déterminer quel opérateur physique doit être utilisé afin d'obtenir un plan d'exécution minimal.

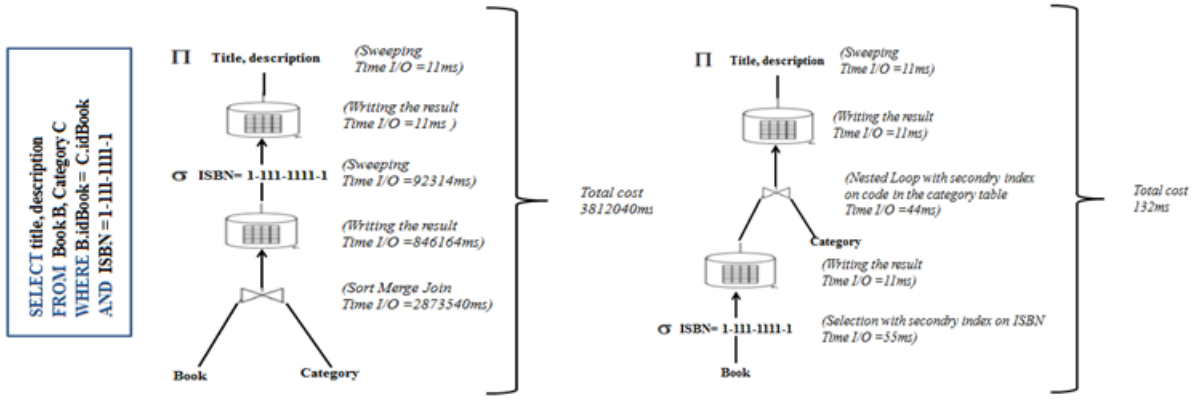


FIGURE 3.2 – Exemple du plan d'exécution d'une requête sélectionnée.

La figure 3.2 montre que la jointure de boucle imbriquée (nested loop join) a été remplacée par la jointure de hachage (hash join) et que (full scan) a été remplacé par (index scan). Les performances de la requête améliorent le temps de CPU et d'E/S du disque (temps d'exécution = 3812,40 s 0.132s), comme le montre la figure 3.3. Ces résultats permettent de conclure qu'il y a un impact sur le coût des requêtes, le faite de choisir des indicateurs tels que les opérations de jointure et les structures physiques comme les index.

3.4 Fondement théorique de notre approche : « Taxonomie de Bloom »

Dans notre approche nous avons basé sur la taxonomie de Bloom comme un fondement théorique. Cette taxonomie était proposée par le chercheur Benjamin Bloom qui a fait émerger une classification des niveaux de pensées importants dans le processus d'apprentissage.

Ces niveaux sont : **la connaissance, la compréhension, l'application, l'analyse, l'évaluation et la création.** Ils sont classés en allant du simple au complexe, du concret à l'abstrait.

Dans cette partie nous allons présenter une figure qui propose une taxonomie d'apprentissage sur l'optimisation de requêtes SQL.
En premier lieu, un apprenant doit connaitre les concepts de base sur l'optimiseur de requêtes SQL, puis une clarification du domaine est effectuée afin de conceptualiser ce domaine par un métamodèle qui va être représenté en détail dans la section 3.7. Les étapes « Knowledge of optimizer based concepts » et « Domain clarification » correspondent aux étapes « mémoriser et comprendre » dans la taxonomie de Bloom.

L'étape d'analyse de la taxonomie de Bloom correspond à notre BPMN (Business Process Modeling and Notation) qui représente la vue processurale de notre approche, ce BPMN englobe plusieurs modules (voir section 3.8), tel que chacun d'eux exprime une suite de tâches. Une formulation de requête SQL, une visualisation du plan et une gestion de transaction constituent les principaux modules de ce BPMN.

Pour évaluer le comportement de l'apprenant, le tuteur peut voir la traçabilité sauvegardée dans un fichier de surveille qui porte le nombre de tentatives, le type d'erreurs (erreurs syntaxiques, erreurs sémantiques et erreurs d'analyse). Cette évaluation correspond à l'étape «évaluer» de la taxonomie de Bloom .

La figure suivante montre l'inclusion de la taxonomie de Bloom dans l'apprentissage du SQL et l'optimisation de requêtes.

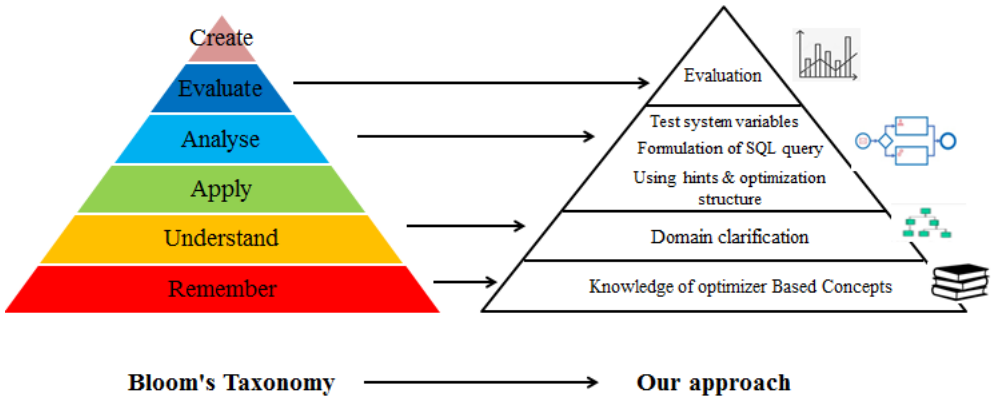


FIGURE 3.3 – Fondement théorique de notre approche.

3.5 Vue d'ensemble de notre approche

Derrière chaque travail dominant dans un domaine spécifique une approche qui le présente, dans cette section nous présentons notre approche, qui comporte trois zones principales, dont la première zone est dédiée aux DBAs, où les caractéristiques et les besoins du DBA sont définis, la deuxième zone est dédiée pour exprimer le plan de la requête. La troisième et la dernière zone est consacrée pour améliorer le plan de requête en se basant sur des hints et des structures d'optimisation.

La figure suivante montre cette vue d'ensemble.

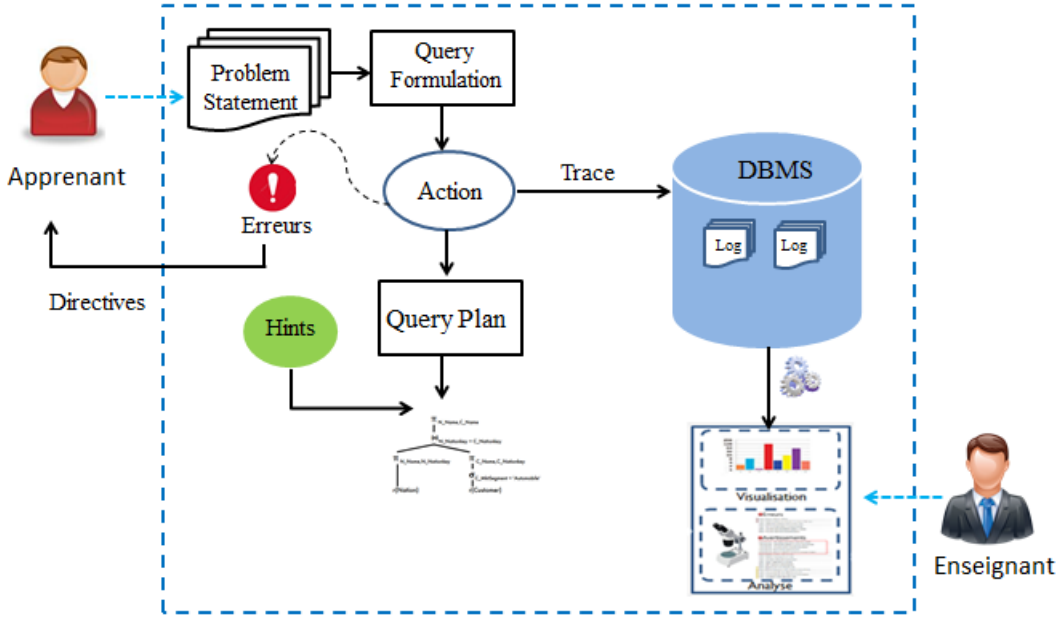


FIGURE 3.4 – Vue globale de notre approche.

3.6 Formalisation du problème de notre framework

Une requête SQL pour être exécutée elle va suivre un ensemble d'étapes qui constituent un processus, ce dernier ne peut pas être visualisé, sans la présence de certains critères, qui font les éléments corps pour le montrer.
La formalisation générale du processus d'exécution de requêtes SQL est définie comme suit : étant donné :

- **Entrées :**
  - Un schéma de base de données BD.
  - Un ensemble de requêtes E = $Q1$, $Q2$,... Qn.
  - Un ensemble de directives (hints) D= $H1$, $H2$,... Hn.
- **Sorties :**
  - Un plan d'exécution optimal.
  - Un fichier log.
  - Une estimation du coût <CPU, I/O>.
- **Objectifs :**
  - Réduire le coût de la requête SQL entrante.
  - Utiliser les règles pour la formulation de la requête SQL.
  - Exploiter bien les hints utilisés.

  Ces objectifs en général servent à visualiser le traitement réel de la requête SQL, en exploitant bien les caractéristiques de chaque entrée.

3.7 Structure de notre framework

Les différents outils, applications et Frameworks développés ont une structure qui les définissent et les représentent, quel que soit le domaine dont ils ont été développés.
Dans ce qui suit, nous allons présenter la structure de notre Framework tout en commençant par une modélisation de domaine et ceci est fait par l'élaboration des méta-modèles qui montrent la relation entre les différentes parties qui constituent cette structure.

3.7.1 La méta-modélisation

La méta-modélisation est l'activité de construire des méta-modèles. Dans l'informatique, la méta-modélisation se définit comme la mise en évidence d'un ensemble de concepts pour un domaine particulier (Barais, 2007). Parmi ces concepts, il y a le modèle qui représente un phénomène particulier du monde réel, le méta-modèle qui est une abstraction mettant en évidence les concepts utilisés pour définir le modèle et le méta-méta-modèle qui a une forme d'abstraction d'ordre supérieur composée de concepts génériques permettant de définir des méta-modèles (voir Figure ci-dessous).

Un méta-modèle est une « définition formelle » d'un modèle qui aide à le comprendre et qui facilite le raisonnement sur sa structure, sa sémantique et son usage. De la même manière, il est nécessaire d'avoir un méta-modèle pour interpréter un modèle, il est nécessaire d'interpréter un méta-modèle, par une description du langage dans lequel il est écrit.

Un langage de modélisation conceptuel peut servir, dans la plus part des cas comme un langage de méta-modélisation. Par exemple, le langage UML peut être considéré comme un langage de modélisation ou encore un langage de méta-modélisation [19].

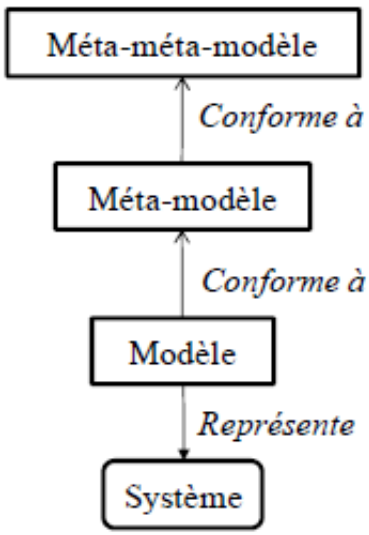


FIGURE 3.5 – Notions de base de la méta-modélisation

3.7.2 Les méta-modèles de notre Framework

Les représentations graphiques sont parmis les moyens les plus indispensables pour bien déterminer les éléments corps d'un outil quelconque.

Cette section est destinée à présenter notre contribution en se basant sur des méta-modèles. La figure3.6 représente les éléments corps de notre méta-modèle, dont l'élément racine est la classe « QueryPlanner », et les classes centrales sont : Query, Database, Operation, Platform, CostModel, Hints, OptimizationStructure.

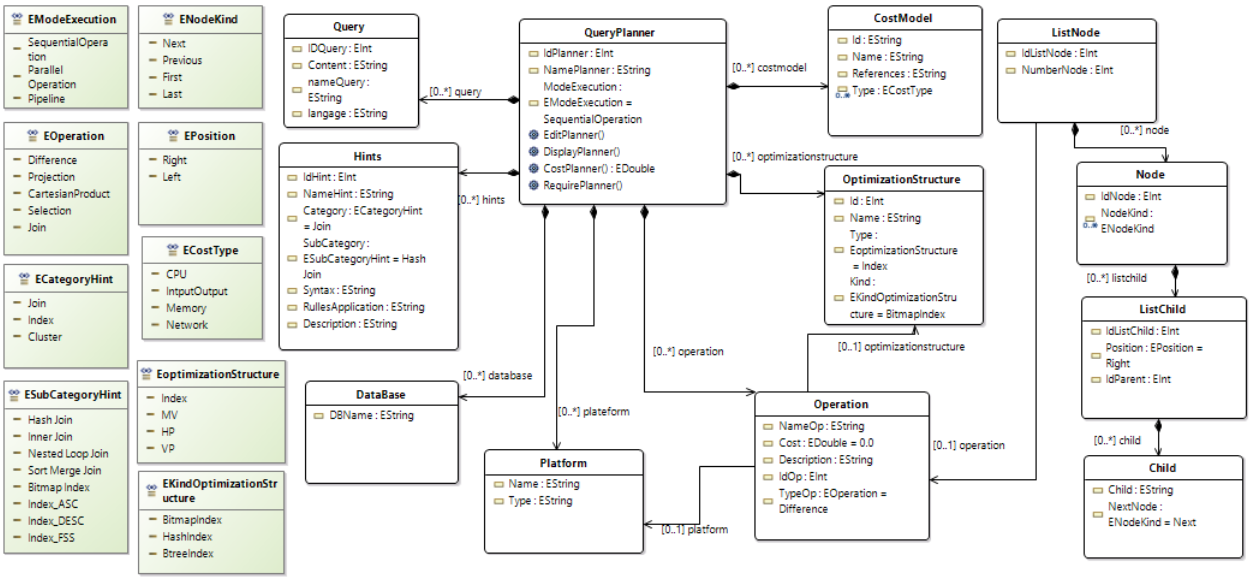


FIGURE 3.6 – Méta-modèle de QueryPlanner.

Ci-dessous, nous détaillons les méta-modèles de notre Framework.

- **Query :** Une instance de la classe requête comporte un prédicat qui peut être textuel, binaire, ou logique, et une opération (join, sélection, produit cartésien. . . etc.) qui peut être binaire ou unaire.

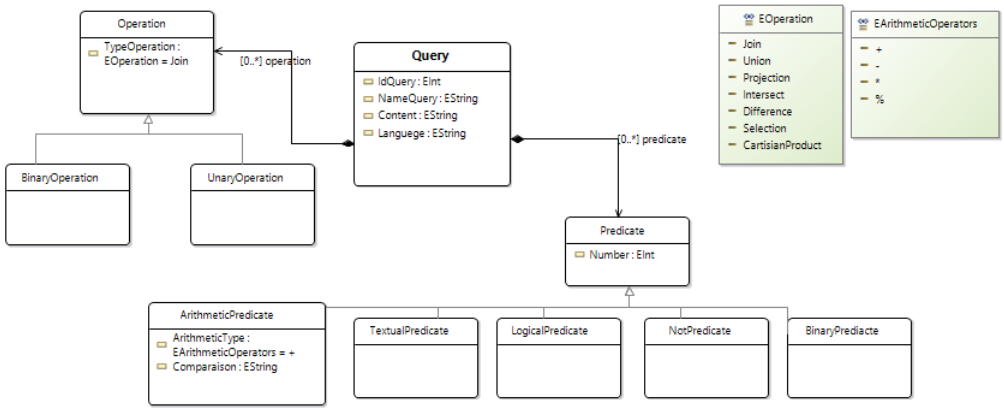


FIGURE 3.7 – Méta-modèle de requête.

- **DataBase :** Une instance de la base de données se compose d'entités conceptuelles et leurs attributs. De plus, les liens entre eux sont aussi représentés par des associations.

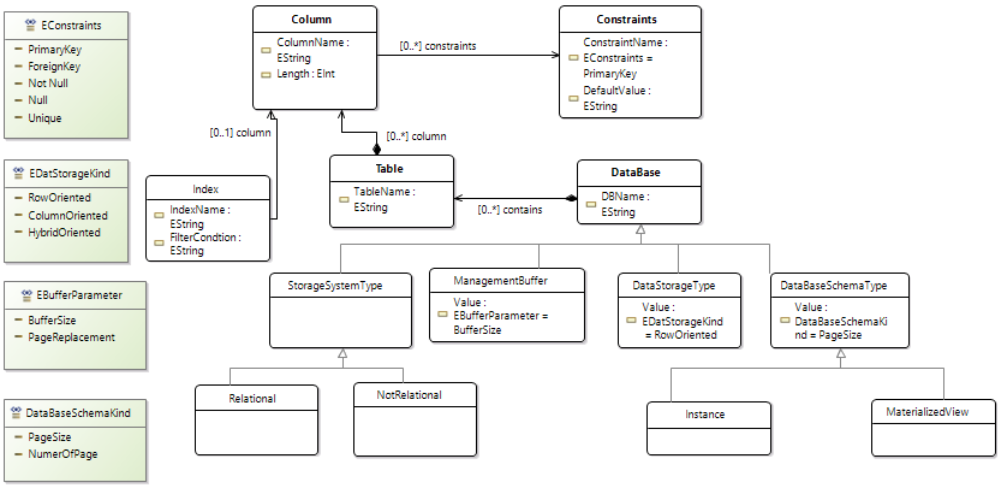


FIGURE 3.8 – Méta-modèle de la base de données.

- **Operation :** Une instance de la classe opération représente les opérateurs physiques sous forme de nœuds. Chaque nœud a un nombre de fils allant de zéro à n, ces nœuds fils peuvent être organisés sous la forme d'une liste(ListChild), qui contient l'identifiant du nœud parent, la classe (ListNode), c'est elle qui est responsable à la gestion des nœuds.

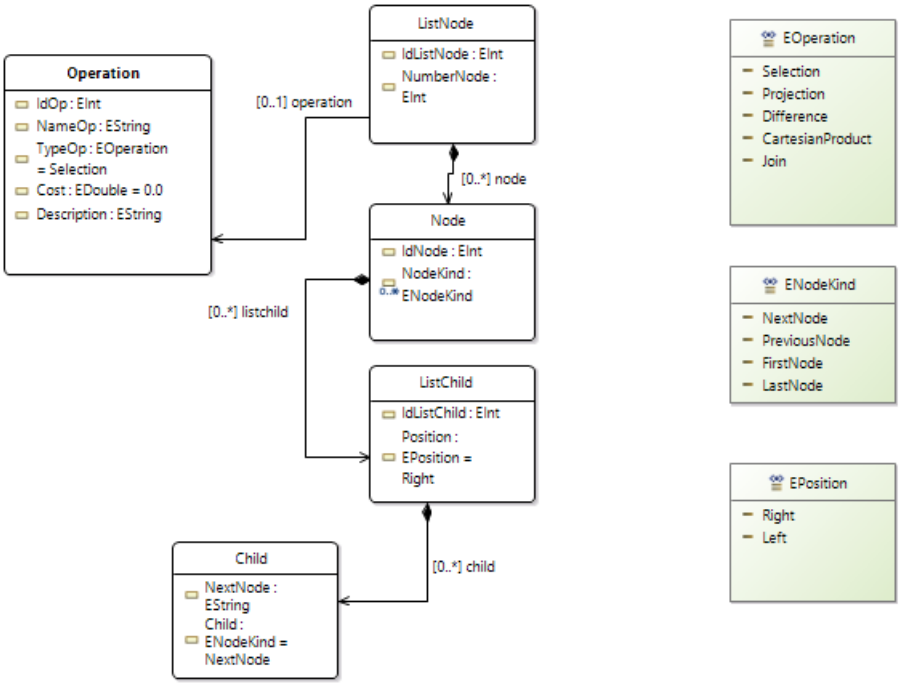


FIGURE 3.9 – Méta-modèle de la classe Operation.

- **Hints :** Une instance de cette classe est caractérisée par un identifiant, un nom, une syntaxe et une catégorie qui peut être représentée par : join, index ou cluster, cette instance est aussi caractérisée par un autre attribut appelé SubCategoryType qui peut prendre une valeur de type E SubCategoryType à savoir : Hash join, Bitmap index. . . etc.

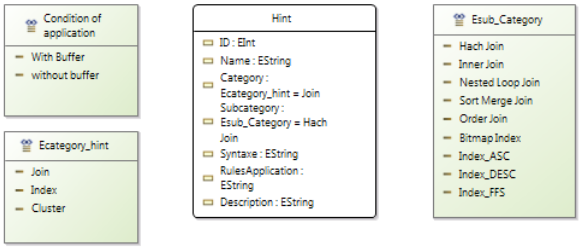


FIGURE 3.10 – Méta-modèle de la classe Hints.

- **CostModel :** CostModel est présenté en proposant un langage de CostDL dédié au domaine du modèle de coût [20]. Une instance de CostModel se compose d'une métrique (instance de classe Metric), un contexte (instance de la classe contexte)et une fonction du coût (instance de la classe CostFunction). Elle se caractérise également par un nom et des références qui indiquent les papiers scientifiques présentent le modèle du coût. Chaque instance de la classe CostModel a également au moins un type de coût. Une formule mathématique d'un modèle du coût qui est soutenue d'une manière structurée grâce à la classe CostFunction.

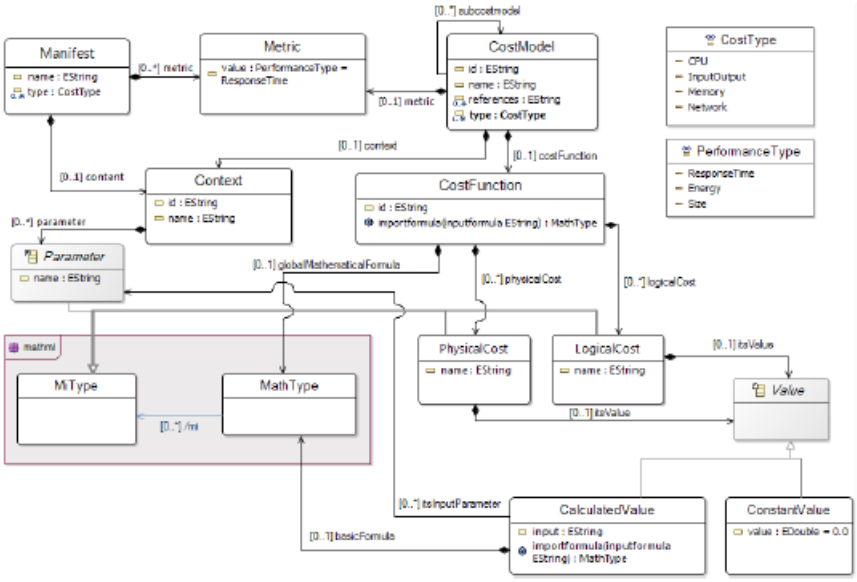


FIGURE 3.11 – Méta-modèle de la classe CostModel.

- **Platform :** Une instance de la classe plateforme inclut l'architecture de BD, qui peut être (centralisée, distribuée, parallèle ou dans le Cloud).
  Ce package présente le modèle du coût avec le type de l'architecture du système (Shared memory, Shared disk, Shared Nothing), il représente également les paramètres de device utilisé par la plateforme déployée comme StorageDevice, ProcessingDevice, CommuicationDevice).

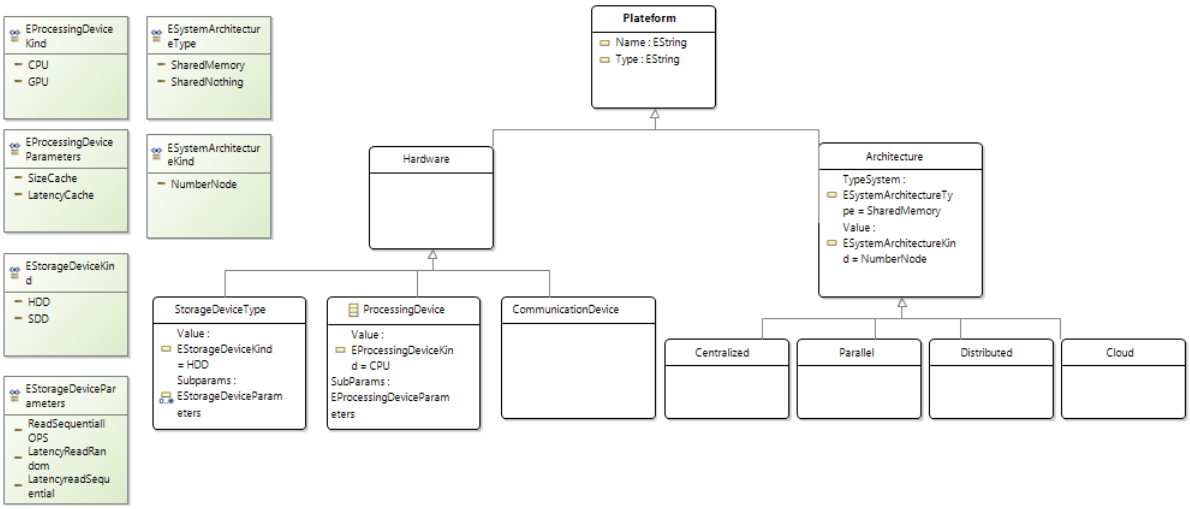


FIGURE 3.12 – Méta-modèle de la classe plateforme.

3.8 Vue d'ensemble du processus de notre framework

Le processus de notre framework est montré par une représentation graphique appelée BPMN (Business Process Modeling Notation), comme illustre la figure 3.13.

Dans ce modèle, les différents volets du BPMN représentent les étapes de conception et de raffinement dans le plan de requête. Dans chaque étape, différentes entrées et sorties de données sont utilisées pour la transformation du plan de requête.

Ce processus est conçu comme une approche descendante comprend trois grandes phases : la formulation de la requête, la visualisation du plan d'exécution et la gestion de transaction. Dans lequel, la première étape détermine l'analyse de la requête SQL entrante et l'utilisation des directives lors de sa formulation. Dans l'étape suivante, l'utilisateur est devant le choix d'une configuration par défaut ou par Hint (Customizing), dans cette dernière les variables vont être changées et de même la structure d'optimisation et la visualisation du plan de requête entrante. Cette dernière va être représentée comme un ensemble de transactions, ou elle peut être validée ou annulée suivant le cas.

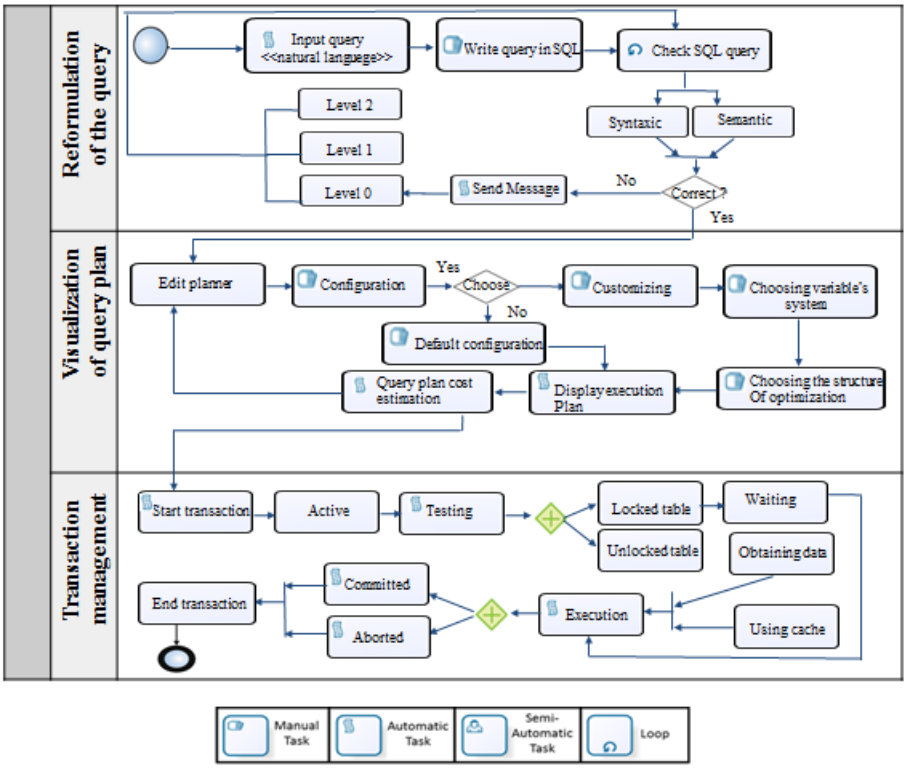


FIGURE 3.13 – Vue d'ensemble du processus de notre framework.

3.8.1 Description des modules de notre BPMN

- **La formulation de la requête SQL :** Dans cette étape, une requête entrante en langage naturel va être transmise par l'utilisateur à une requête SQL bien formulée. A ce stade-là, une analyse syntaxique et sémantique est associée plus un ensemble de directives (directives de syntaxe, de sémantique et d'analyse) est donné par niveaux à l'utilisateur

afin de lui permettre d'écrire une requête SQL juste et valide.

La figure suivante montre le traitement correspond à ce module.

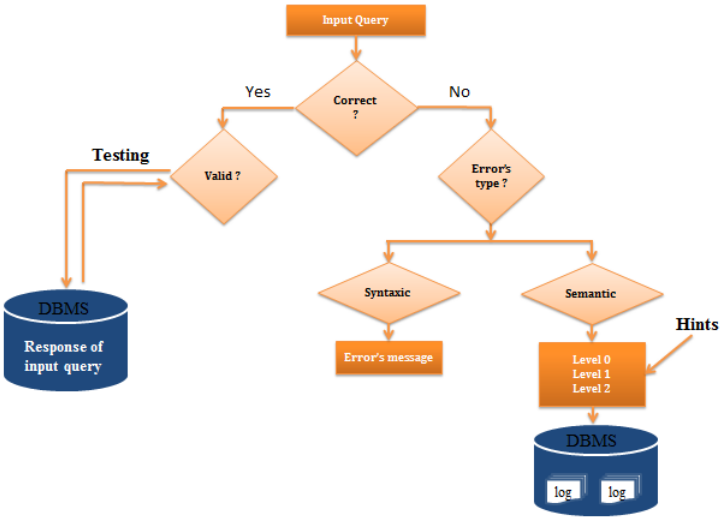


FIGURE 3.14 – Explication du module formulation de la requête SQL.

— **Directives de la formulation SQL** La catégorie des erreurs donne la possibilité de classer chaque tentative de l'apprenant dans une catégorie. Afin d'aider l'apprenant et augmenter l'interactivité avec notre système proposé. Nous avons proposé un algorithme qui affiche des directives (HINT).Ces directives sont affichés selon le type d'erreurs et le nombre de tentatives. (Voir tableau 3.1).

Niveau de directives	Description
Niveau 1	Directives en langage SQL
Niveau 2	Directives en langage naturel
Niveau 3	Directives contient le type d'erreur

TABLE 3.1 – Niveaux de directives.

— **Classification des erreurs** Nous avons classé les types d'erreurs relatives aux tentatives de la formulation de requête SQL dans la figure 3.15, ces types d'erreur sont :

  — **Erreurs d'analyse :** Ce type d'erreurs lié à la difficulté à analyser le fonctionnement des opérateurs et leur ordre par un apprenant.

  — **Erreurs syntaxiques :** Ce genre d'erreurs montre si une requête est conforme à la grammaire SQL ou pas, c'est-à-dire si cette requête respecte cette grammaire.

  — **Erreurs sémantiques :** Elles montrent si une requête SQL est correcte syntaxiquement. Cette erreur est apparue quand par exemple nous référencions un attribut ou bien une table qui n'appartient pas au dictionnaire de base de données.

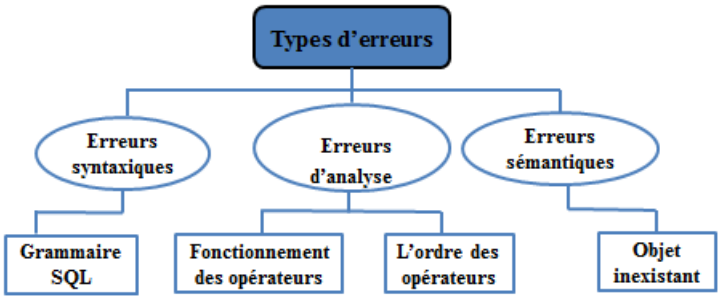


FIGURE 3.15 – Organigramme des types d'erreurs.

3.8.2 La visualisation du plan de la requête SQL

Dans ce module un plan d'exécution de la requête SQL entrante va être affiché. L'utilisateur va choisir une configuration parmi celles existantes (configuration par défaut ou par des directives), cet utilisateur va choisir aussi les variables du système et la structure d'optimisation (vue matérialisée, index, partitionnement. . . etc.). la figure 3.16 illustre ce module.

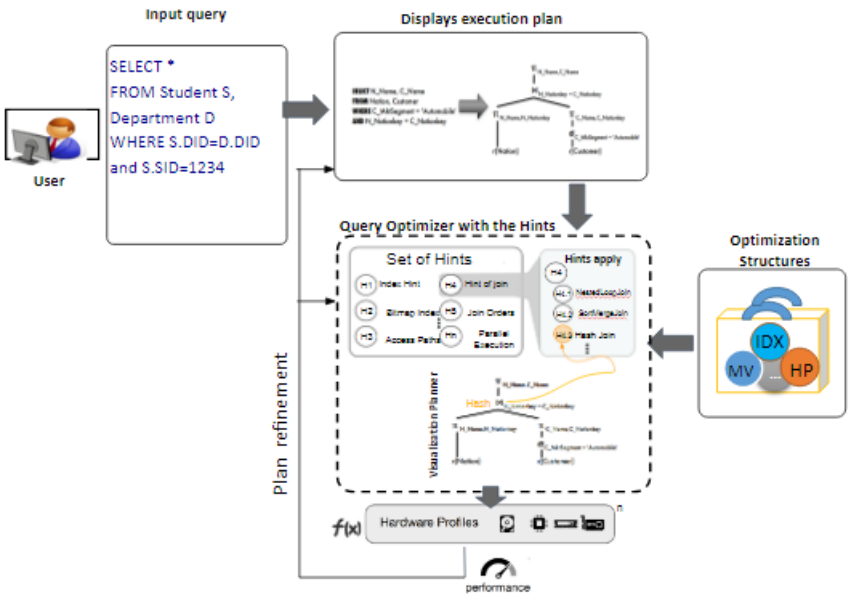


FIGURE 3.16 – Explication du module planner.

3.8.3 Gestionnaire de transaction

Dans cette étape une modification est portée sur des variables du système tels que l'auto-commit et le cache, ce dernier qui peut prendre une des trois valeurs suivants (ON, OFF ou à la demande), et une analyse sont faits par l'utilisateur. A ce stade-là, un verrouillage ou un déverrouillage des tables est met afin d'avoir l'impact sur le coût de la requête SQL entrante. Ce module permet l'interactivité avec le SGBD.

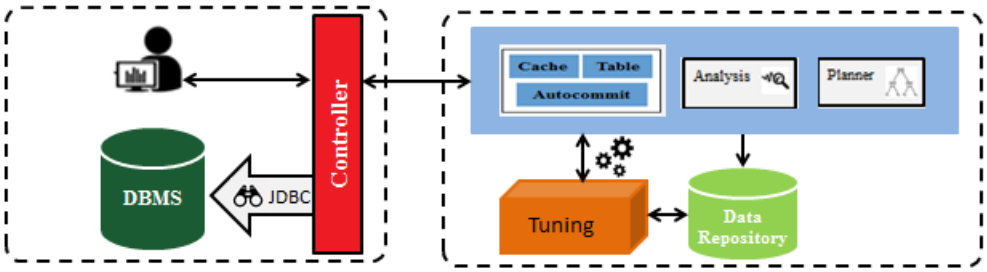


FIGURE 3.17 – Explication du module gestionnaire de transaction.

3.8.4 Exemple d'instanciation

Dans cette section, nous présentons un exemple d'instanciation de notre système, ou en montrant des exemples sur la sélection de la requête, l'utilisation des directives (hints), la visualisation du plan d'exécution de cette requête et puis un scénario de la gestion de transaction. En les montrant dans la figure suivante.

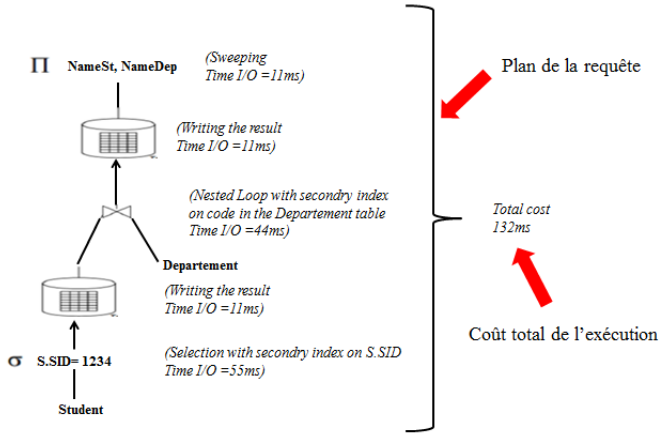
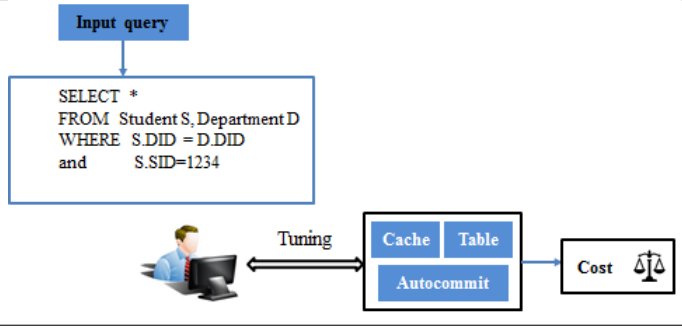
Etape	Exemple
Sélection de la requête	SELECT NameSt, NameDep FROM Student S, Departement D WHERE S.SID= D.SID AND S.SID= 1234
Utilisation des hints	SELECT NameSt, NameDep /* USE Nested Loop join */ FROM Student S, Departement D WHERE S.SID= D.SID AND S.SID= 1234
Visualisation du plan d'exécution de requête	
Gestionnaire de transaction	

TABLE 3.2 – Exemple d'instanciation de notre framework.

3.9 Conclusion

Ce chapitre est organisé en différentes sections, dans le but de comprendre notre approche, et mettre en évidence l'ensemble des étapes suivies pour bien eclaircir notre approche.

Dans ce chapitre nous avons vu une analyse, une formalisation du problème sous forme d'entrées, sorties et objectifs. Nous avons présenté une vue globale de notre approche et une représentation graphique qui a met le point sur notre Framework, telles que des méta-modèles et un BPMN. Nous présentons dans le chapitre suivant la solution que nous avons proposé pour développer notre Framework, et nous présentons aussi les différents digrammes UML qui font partie de la modélisation conceptuelle de ce Framework.

Chapitre 4

Implémentation et la mise en oeuvre de notre application.

« Genius consists in realizing in mature age a great youthful thought. »
— Alfred de Vigny

4.1 Introduction

Après la présentation de notre contribution et la démarche suivie pour l'obtention du but établi précédemment, nous illustrons dans ce chapitre les différentes technologies utilisées pour le développement de notre outils, ainsi, nous présentons la conception de notre outils afin de figurer le travail fait, et d'illustrer les principales fonctionnalités réalisées.

4.2 Présentation des technologies de développement utilisées

Cette section est consacrée pour la présentation des différents outils utilisés pour le développement de notre application (Webdev17, MySQL, Ecore, Entrepris Architect. . . etc).

FIGURE 4.1 – les technologies de développement utiliser.

4.2.1 Langage WLangage

WLangage est le langage utilisé pour le développement de notre outil, il est considéré comme un langage de programmation de quatrième génération qui ne peut être manipulé qu'avec les outils PC SOFT [10] .Ce langage dispose des fonctions habituelles des langages de programmation, et la raison pour laquelle on a choisis ce langage c'est que sa manipulation est faisable, elle réduit une grande quantité du code nécessaire par l'élimination de la phase de codage des interfaces, car elle permet la définition et le test de ses interfaces dans un mode 100% WYSIWYG [11] , y compris pour la définition avancée des contrôles (champs). Bien entendu, le WLangage permet l'accès et la modification pour tout ce qui a été créé.

FIGURE 4.2 – Logiciel webdev.

4.2.2 Serveur MySQL

Pour la gestion de base de données de notre application nous avons choisis MySQL (My Structured Query Language), qui désigne un serveur de bases de données relationnelles, distribué sous licence libre GNU (General Public License) dans la plupart du temps elle est intégrée dans la suite des logiciels LAMP [12] qui comprend un système d'exploitation(Linux), un serveur web (Apache) et un langage de script (PHP).
MySQL stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de cet ensemble. Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête. Les tables du MySQL sont généralement manipulées et interrogées à travers des requêtes écrites en SQL pour récupérer des informations rapidement sur ce serveur.

10. Société française d'édition de logiciels basée à Montpellier, elle est spécialisée dans les environnements de développement professionnels, en particulier les ateliers de génie logiciel.
11. L'acronyme de What You See Is What You Get qui signifie "ce que vous voyez est ce que vous avez".
12. C'est un acronyme désignant un ensemble de logiciels libres permettant de construire des serveurs de site web.

4.2.3 Navicat for mysql

Navicat for mysql, représente un puissant outil d'administration et de développement de base de données MySQL, qui peut convertir XML, CSV, MS Excel Et MS Access formats de données à MySQL bases de données, éliminant la saisie des données de temps et les erreurs qui l'accompagnent. Les autres grandes fonctions utiles comprennent Assistant Importation et Exportation, support Unicode, HTTP / SSH Tunnel, la planification des lots de travail, la synchronisation des données, transfert de données, le constructeur visuel de requêtes, et le constructeur de rapport visuel. Il dispose également d'importer des données à partir d'ODBC et lots ordonnancement des tâches.

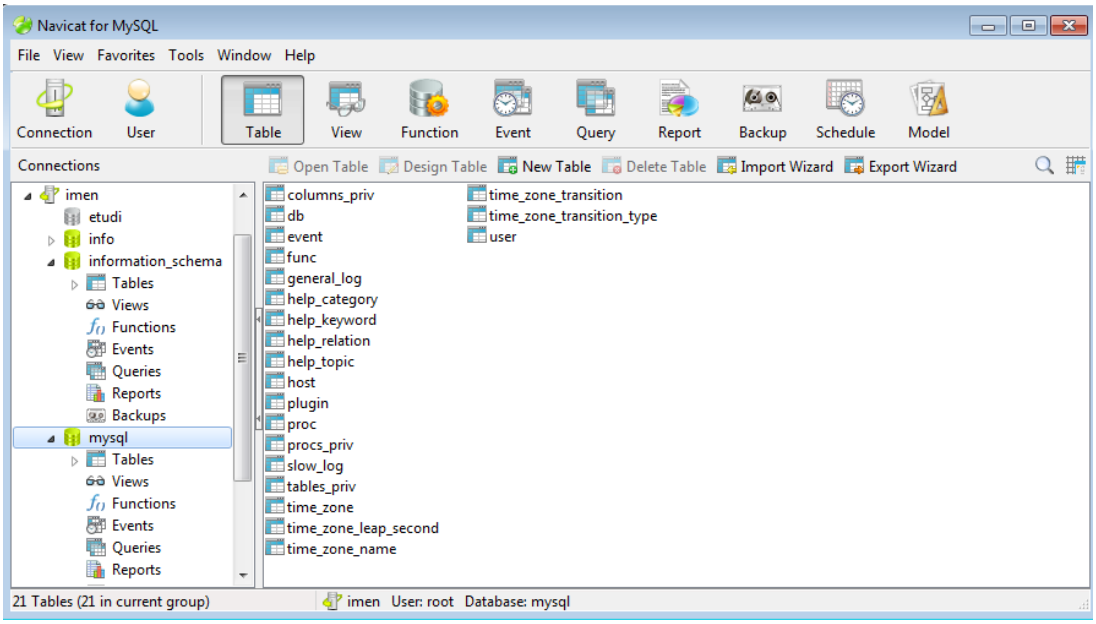


FIGURE 4.3 – Navicat for MySQL.

4.2.4 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 inclut un métamodèle Ecore pour décrire les modèles et le support d'exécution pour les modèles.

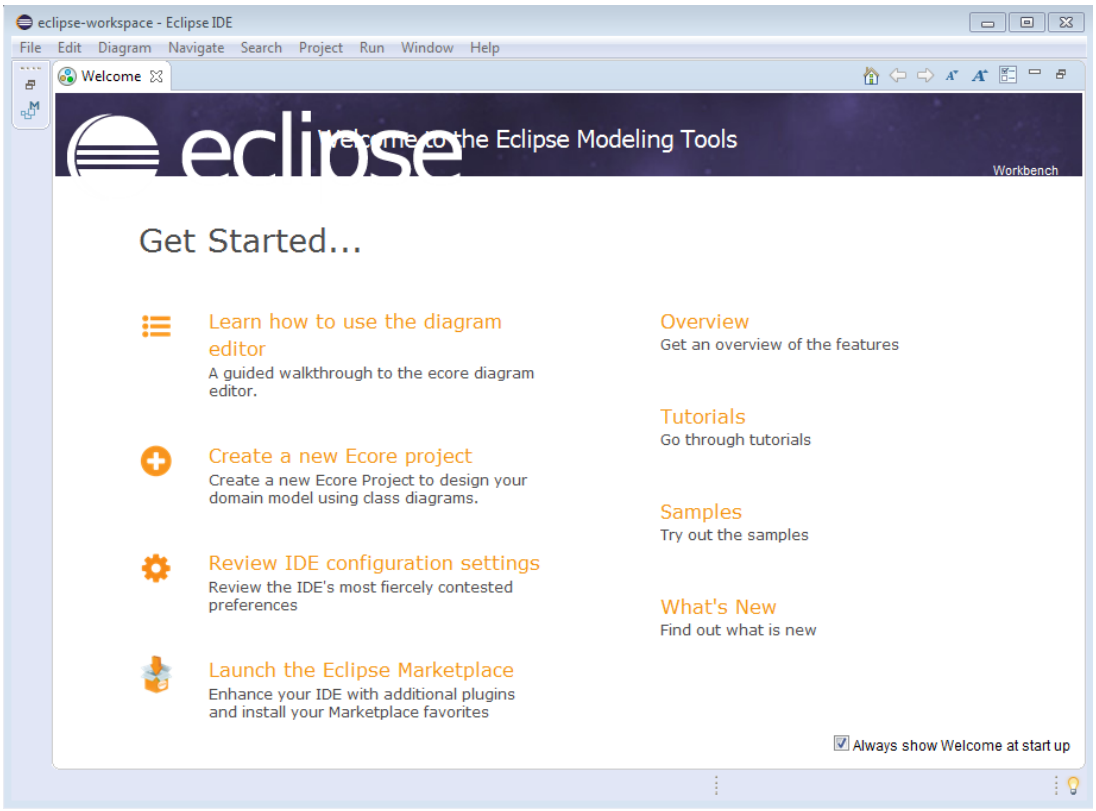


FIGURE 4.4 – Eclipse Modeling Tools.

4.2.5 Entreprise Architect

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

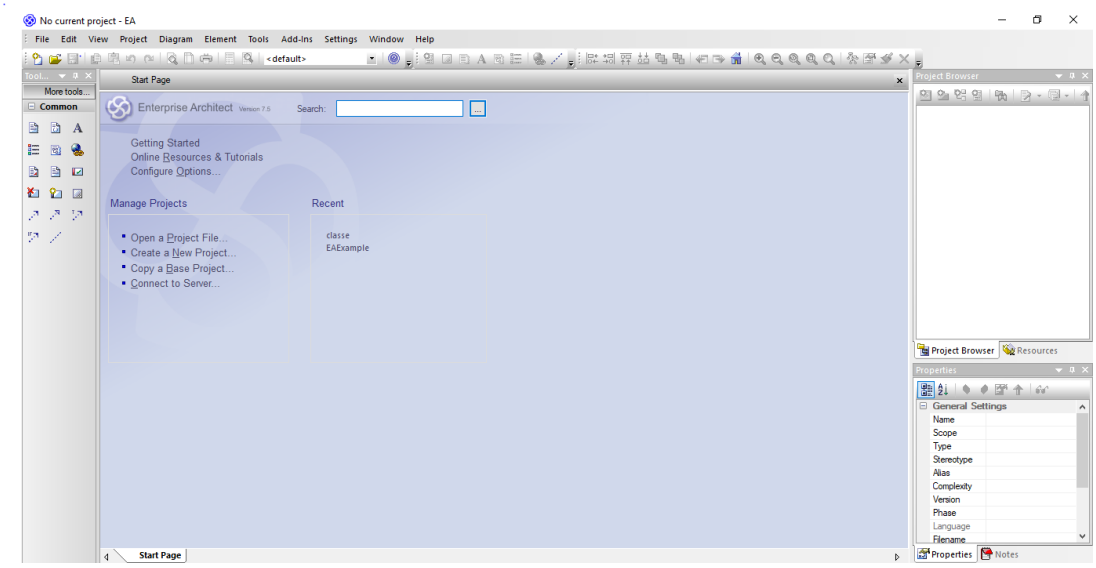


FIGURE 4.5 – Entreprise Architect.

4.3 Présentation de notre outil

4.3.1 Objectifs

Notre outil sert principalement à réaliser une plateforme éducative et interactive, spécialisée à l'apprentissage du processus d'execution des requêtes SQL, en particulier la phase d'optimisation en mode CBO des requêtes, afin d'offrir l'opportunité aux apprenants de maîtriser ce processus et de bien comprendre la phase d'optimisation. Nos principaux objectifs sont :

- Assister les apprenants et les professionnels lors de l'étape de l'optimisation à base de coût des requêtes SQL.
- Offrir aux apprenants une image mentale sur le déroulement de ce processus.
- Aider les apprenants à comprendre l'optimisation des requêtes SQL à travers la visualisation du plan de cette requête.
- Offrir une vue transparente sur les aspects liés à l'optimiseur des requêtes SQL.
- Faire face aux difficultés de la formulation des requêtes SQL.
- Donner aux apprenants l'opportunité d'exercer et de comprendre d'une manière interactives les transactions avec la possibilité de visualiser l'état des variables système, les tables, cache. Etc.
- Offrir aux enseignants la possibilité de suivre et d'évaluer le comportement des apprenants (Nombre tentative, nombre d'erreur, type d'erreur) lors de la résolution du problème (requête).

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 les principales fonctionnalités réalisées.

4.3.2.1 La modélisation du système

Parmi les diagrammes UML largement connus par les informaticiens, nous citons :

- *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 et il montre la distribution des composants logiciels sur la base d'unités d'exécution.

  Notre mécanisme supporte plusieurs acteurs (utilisateur Naïf, utilisateur Expert, Apprenant, Enseignant) mais nos acteurs principaux sont, l'apprenant qui va utiliser la plateforme dans le but d'apprentissage, et l'enseignant dans le but de suivre et d'évaluer le comportement des apprenants.

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

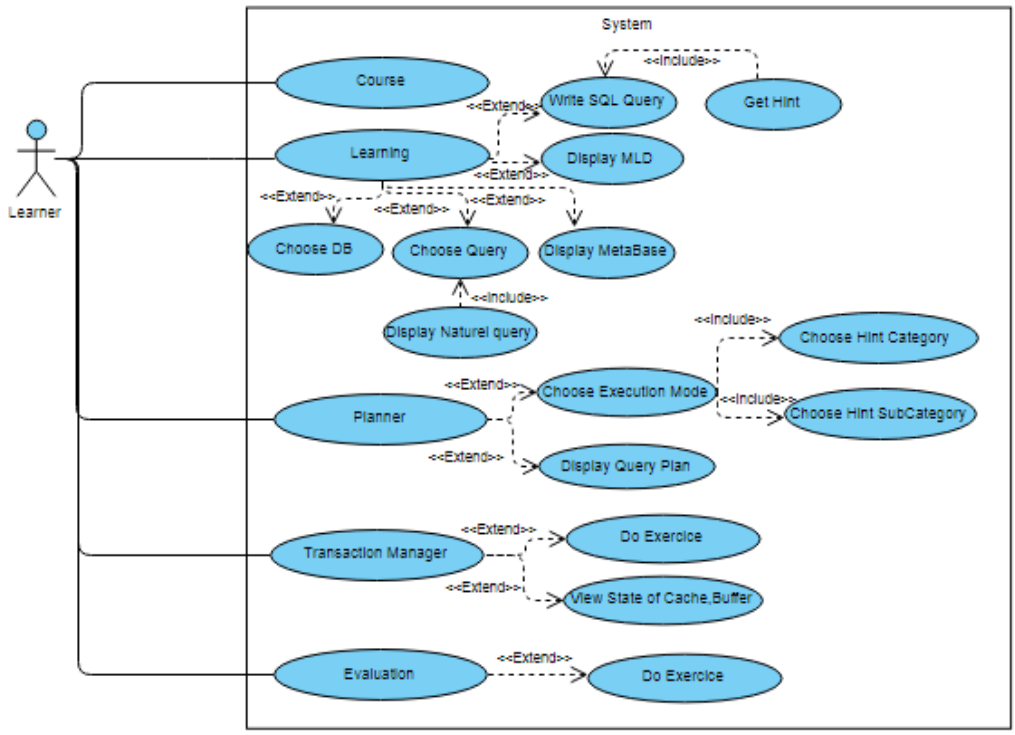


FIGURE 4.6 – Diagramme de cas d'utilisation.

(b) **Diagramme des 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 parmi nos diagrammes de séquence :

— **Accéder au volet Learning :** L'apprenant commence par choisir la base de données(BD) pour le traitement, ensuite il doit effectuer un choix de question voulue résoudre, parmi un ensemble de requêtes stockées dans une BD après ces opérations le système charge la requête sélectionnée en langage naturel pour qu'elle soit résolue par l'apprenant, en cas d'erreurs lors de la réponse sur la requête choisie le système affiche des hints par niveaux pour aider l'apprenant, en second lieu l'apprenant peut accéder à plusieurs fonctionnalités de ce système telles que, l'affichage du schéma de BD sélectionnée, l'affichage des données supplémentaires (metabases) de cette dernière et il peut avoir un exemple sur la formulation d'une requête pour donner à l'apprenant une vision globale sur la structure générale d'une requête.

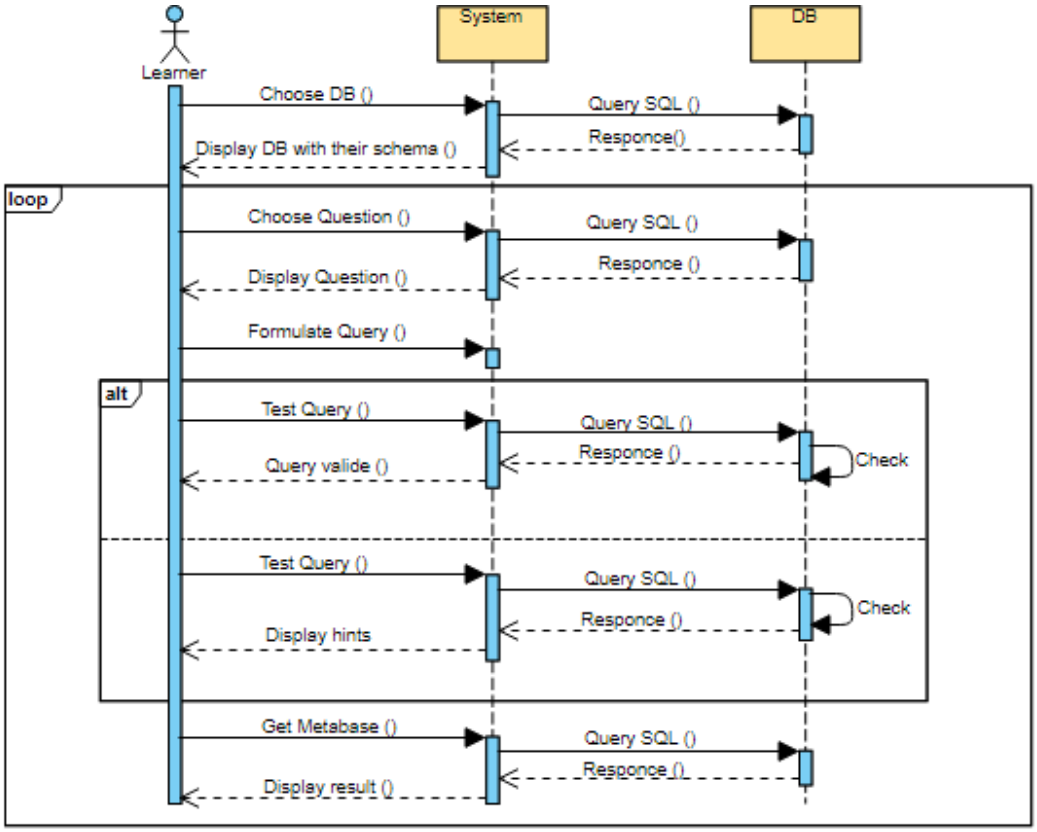


FIGURE 4.7 – Diagramme de séquence : Learning.

— **Accéder au volet Planner :** après que la requête soit valide, l'apprenant passe à visualiser le plan d'execution de la requête formulée, en premier lieu il doit choisir la configuration voulue (Default configuration, Customize), après que le système applique la configuration sélectionnée, l'apprenant est invité à choisir le mode d'execution (Rule Base, Cost Base), le mode cost base est le mode par défaut du système, en second lieu l'apprenant à la possibilité de visualiser le plan de la requête ainsi il peut faire du tuning et revisualiser le plan de la requête écrite après certain changement fait par l'apprenant à savoir la modification des variables système... etc.

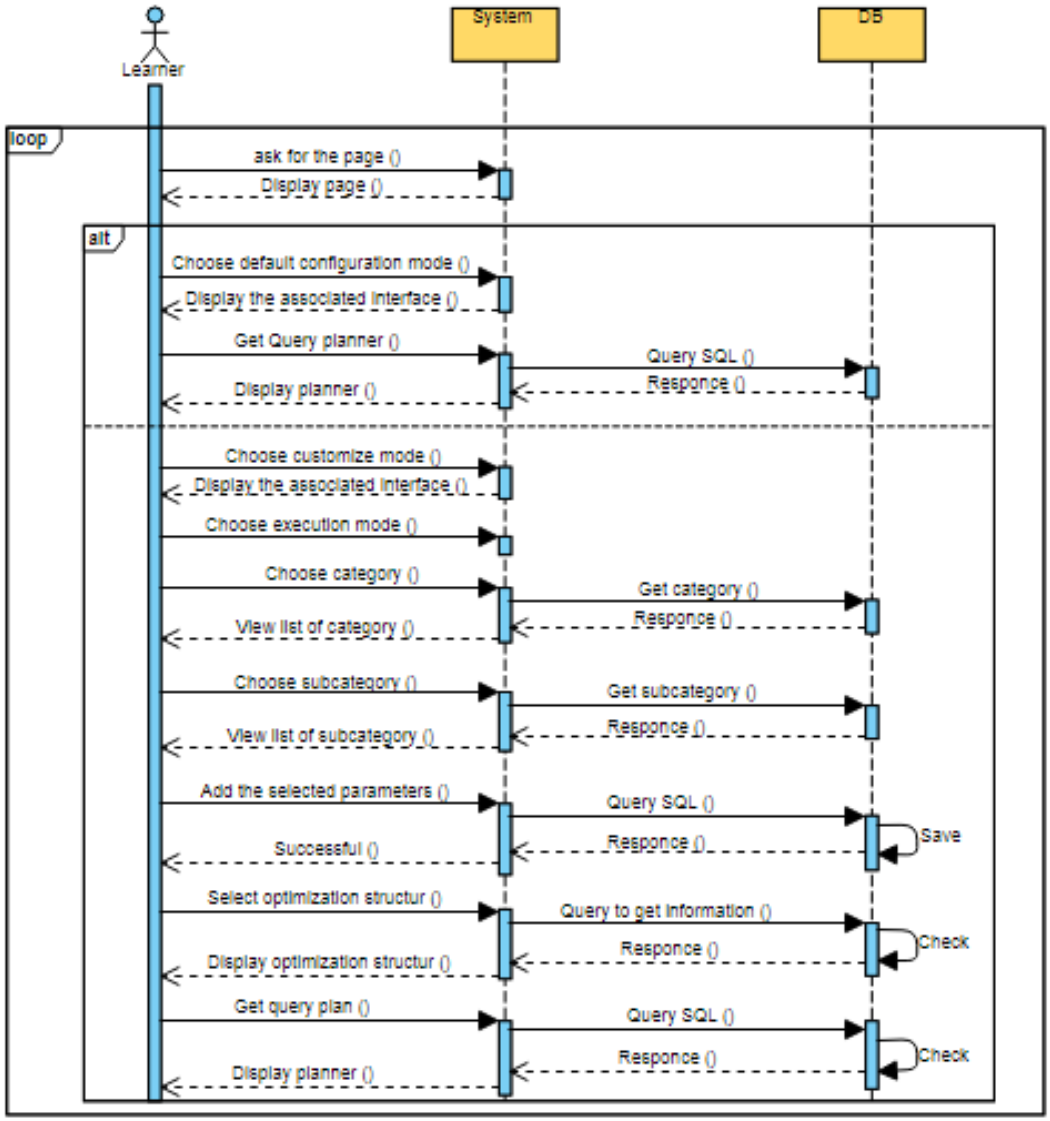


FIGURE 4.8 – Diagramme de séquence : Planner.

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

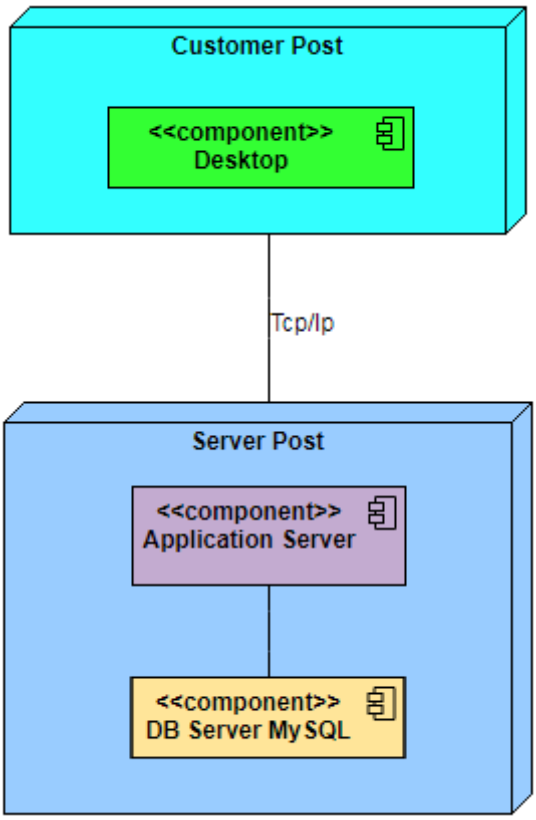


FIGURE 4.9 – Diagramme de déploiement.

Passons maintenant à la deuxième partie, concernant la présentation du framework réalisé nous montrons quelques fonctionnalités de cet outil en prise d'écrans.

4.3.2.2 Présentation des interfaces

Nous présentons dans cette partie un ensemble des captures d'écran qui illustrent les fonctionnalités principales de notre framework.

— **Interface Principale :** Nous commençons notre démonstration par une vue globale de l'interface de notre framework qui se compose de plusieurs volets enchainés afin d'offrir un meilleur apprentissage pour l'apprenant sur le processus d'exécution de requêtes SQL plus particulièrement l'étape de l'optimisation. Pour évaluer et suivre le comportement de cet apprenant à partir des traces sauvegardées.
Cette interface affiche au début une variation d'informations concernant l'outil développé, plus d'autres volets qui sont : Catalogue, Learning, Planner, Transaction_Management, Traceability.

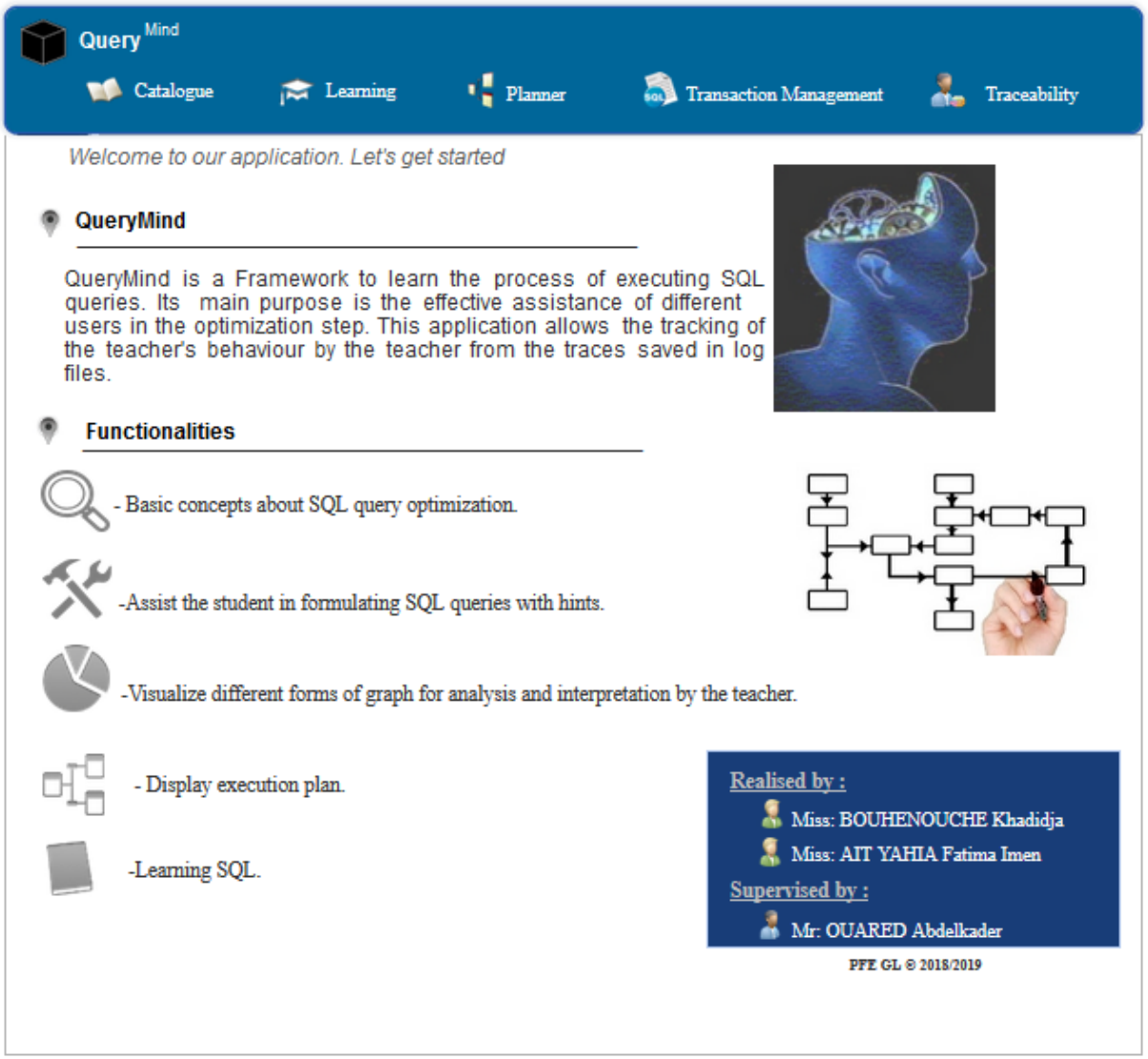


FIGURE 4.10 – Interface principale.

— **Volet "Catalogue" :** Ce volet a pour but d'offrir aux apprenants un support éducatif, grâce à lui l'apprenant est capable d'acquérir un ensemble de principes de base sur la requête SQL, son optimisation et son exécution, ce volet lui permet aussi de comprendre et d'approfondir les concepts de base acquis.

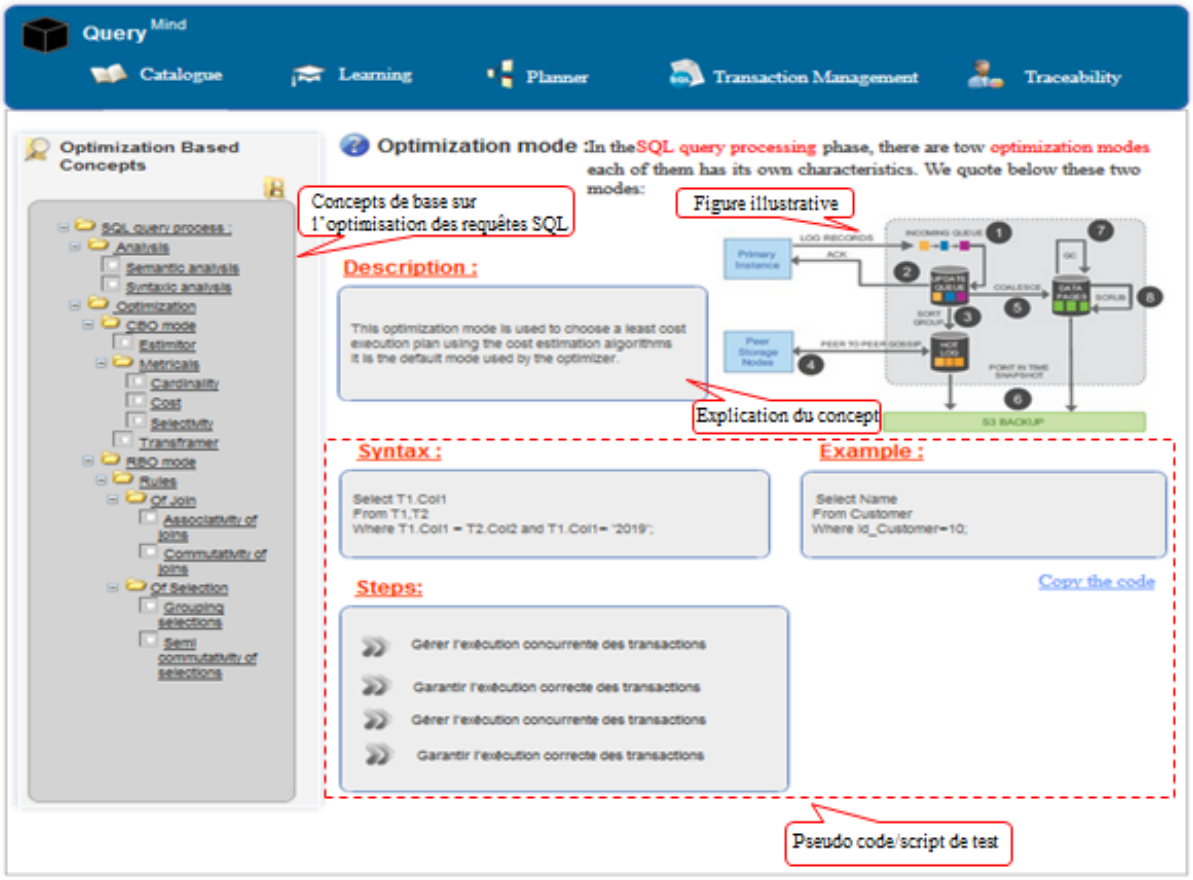


FIGURE 4.11 – Interface "Catalogue".

— **Volet "Learning" :** Dans cette partie du framework l'apprenant commence à exercer ce qu'il avait appris auparavant, il commence par la selection de la base de données sur laquelle aura lieu le traitement, ensuite il va choisir la requête parmi un ensemble de requêtes proposées.
Puis l'apprenant peut répondre à cette requête, qui sera dirigé par des directives en cas d'erreurs, pour garantir un meilleur apprentissage par degré d'absorption. L'apprenant peut aussi accéder à plusieurs fonctionnalités telles que l'affichage de la métabase de la base de données sélectionnée, retour en arrière à tout moment, et finalement l'apprenant peut exécuter sa requête afin d'afficher le résultat.

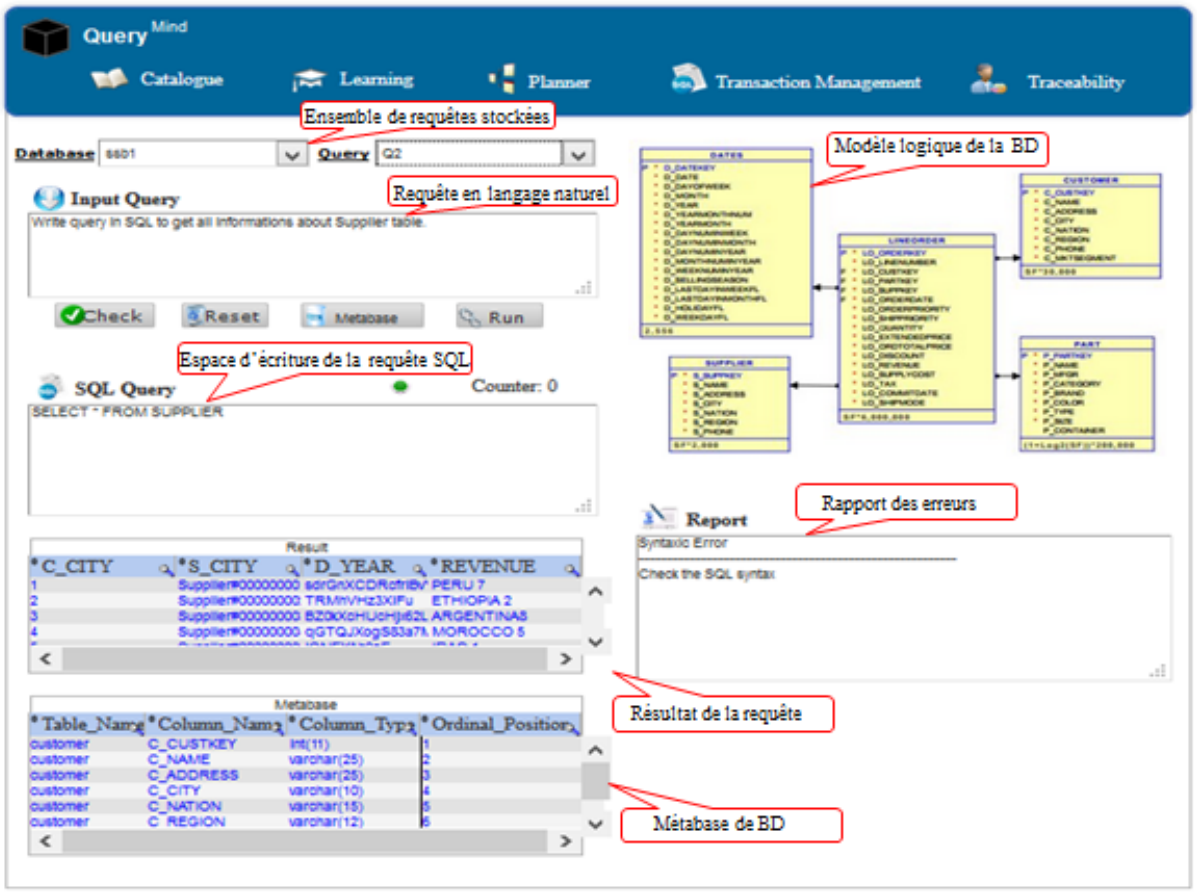


FIGURE 4.12 – Interface "Learning".

— **Volet "Planner" :** Ce volet offre à l'apprenant la possibilité de visualiser le plan de la requête écrite auparavent ainsi son coût, et leur temps d'execution. Après un certain tuning (choix du mode d'execution, modification des variables système, taille buffer), l'apprenant est capable de visualiser le changement du coût du plan d'execution de la requête.

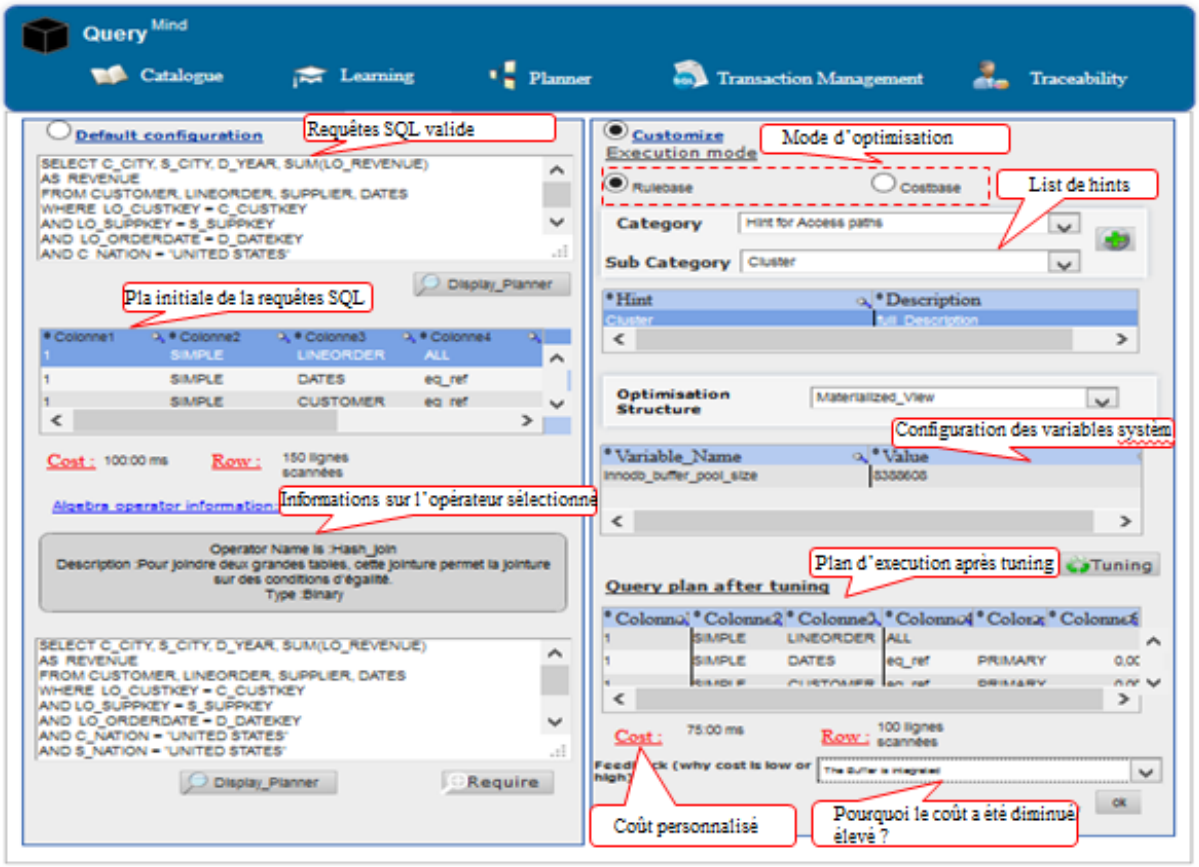


FIGURE 4.13 – Interface "Planner".

— **Volet "Transaction Management" :** Dans ce volet l'apprenant est capable de bien saisir la notion de transaction et comment se déroule son traitement tout ça c'est à travers la visualisation des changements dans le cache, buffer et l'état des tables (Lock, Unlock) lors de l'execution de la transaction, l'apprenant peut aussi dérouler un exemple, dont il va manipuler certaines variables comme : Rollback, Savepoint. De même l'apprenant peut voir la solution du script choisi.

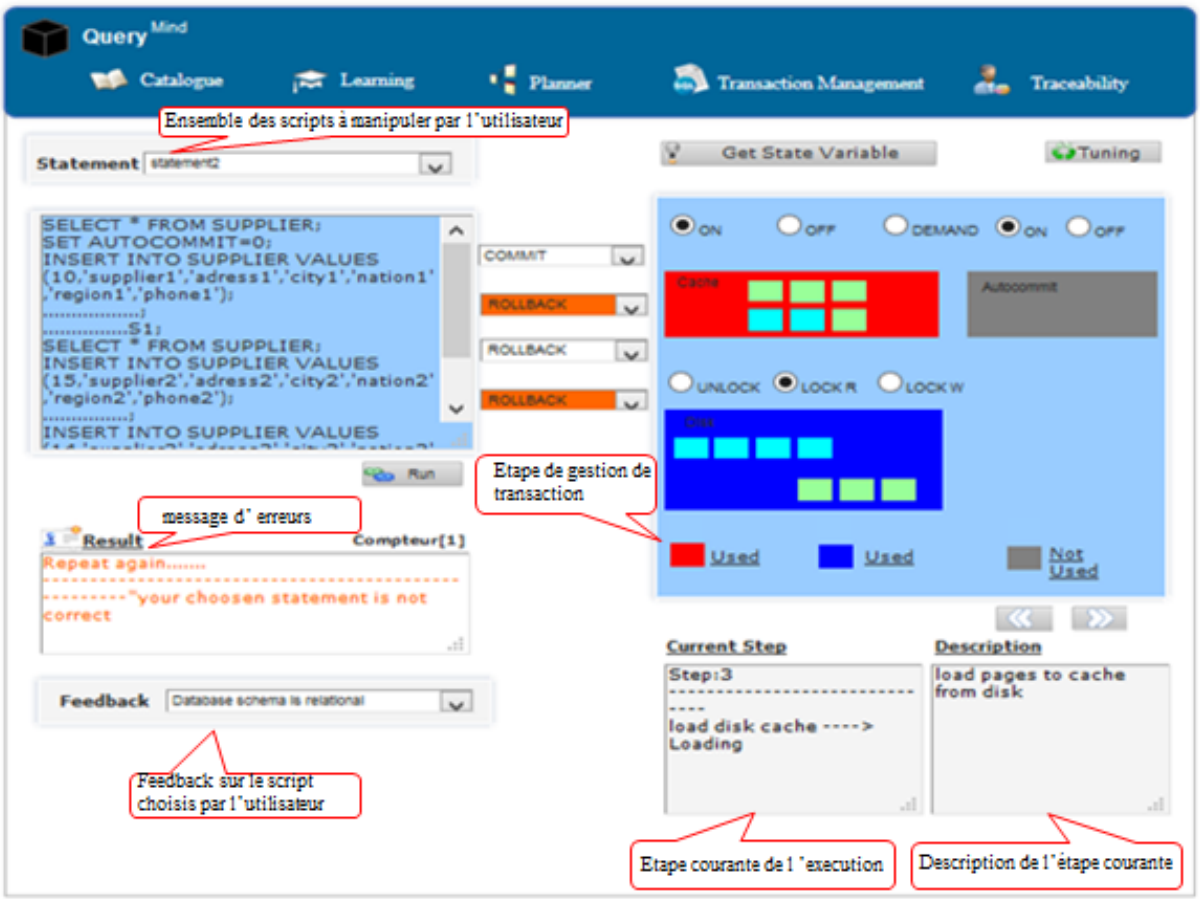


FIGURE 4.14 – Interface "Transaction Management".

— **Volet "Traçabilité" :** Ce volet montre plusieurs graphes comme type d'erreur, temps d'execution de la requête, ces graphes offrent la possibilité à l'enseignant de suivre et de garder une trace sur le comportement de l'apprenant.

FIGURE 4.15 – Interface "Traçeability".

4.4 Conclusion

Dans ce chapitre nous avons commencé en premier lieur par présenter les différentes technologies utilisées pour l'implémentation de notre framework, en second lieu nous avons montré notre outil tout en commençant par la modélisation et en terminant par des captures d'écrans illustratives de notre framework.

Troisième partie

Annexes

Chapitre 5

Gestion du Projet

"If you tell people where to go, but not how to get there, you'll 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 l'état d'avancement du projet. Dans ce chapitre, nous détaillons la méthode que nous avons suivi pour bien 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, et nous terminons enfin par une étude de risques pour notre 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 et nous metons l'accensent sur les différents concepts ambigus afin de les résoudre. La démarche à suivre doit donc nous permettre de s'adapter rapidement 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.

1. **Analyse préliminaire des besoins** : cette étape nécessite l'identification des fonctionnalités principales du système nécessaires pour la proposition d'un premier prototype.
2. **Développement du prototype** : construction du prototype
3. **Évaluation du prototype** le prototype est évalué par l'encadreur pour valider s'il correspond aux besoins exigés.

4. **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 5.1

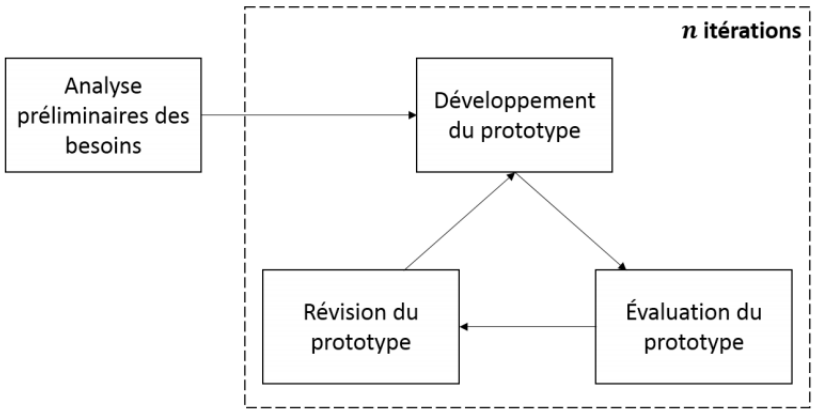


FIGURE 5.1 – Les quartes étapes du cycle de vie prototypage

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.*" [21]. 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 khaldoun. 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 sommes basées sur une méthode de travail qui nécessite le contact hebdomadaire avec nôtre encadreur. Ce contact se traduit par les différents types des réunions qui sont présentées dans le tableau suivant.

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.

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

Table 5.1 – Le contact hebdomadaire avec l'encadreur

5.3.2 livrables

Les livrables que nous allons énumérer dans cette section ont 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 à notre encadrant d'être à jour avec les différentes fonctionnalités proposées,les livrables intermédiaires sont ceux les suivants.

1. le plan des chapitres.
2. Architecture générale du prototype.
3. Astuces pour le dévelopment du prototype.

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.

Numéro	Nom du risque	Catégorie
1	Le niveau d'anglais	Technique
2	Compréhension technique	Technique
3	Conjoncture du pays	Humain
4	Diversité des outils	Technique

Table 5.2 – Risques liés à notre projet

5.3.3.1 *Niveau d'anglais*

Étant donné, que la plupart des publications scientifiques sont rédigées en anglais. Ces publications demandent un bon niveau technique d'anglais pour les biens assimiler. Actions

à mettre en œuvre :

— 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 exploité durant la préparation du mémoire à demander un temps assez long dont nous avons appris la maitrise de ces derniers. les deux mois qui ont précédé ont été consacrés à l'apprentissage des outils.
Actions à mettre en œuvre :

— prévoir au préalable les outils a exploités durant votre travail pour s'adapter.

5.3.3.3 Conjoncture du pays

Une situation inhabituelle (des manifistations,..ect) ont été notées ces derniers mois, ce qui a provoqué des perturbations au niveau des universités ceci est répercutée sur le travail et son avancement.

5.4 Rédaction de rapport du PFE

La rédaction du mémoire été divisée en deux grandes périodes. la premiére du septembre jusqua'a la fin de mois de mars cette periode est pour l'état de l'art, elle à été concacrée pour l'analyse du domaine et la délimitation du cadre de travail, cette phase a été rédigée au fur et à musure avec la rédaction du mémoire. La deuxième phase a commencé au début du mois de mars et elle a été étalée jusqu'au mois de juin, elle a été concacré pour tous ce qui est conception et implémentation du notre prototype.

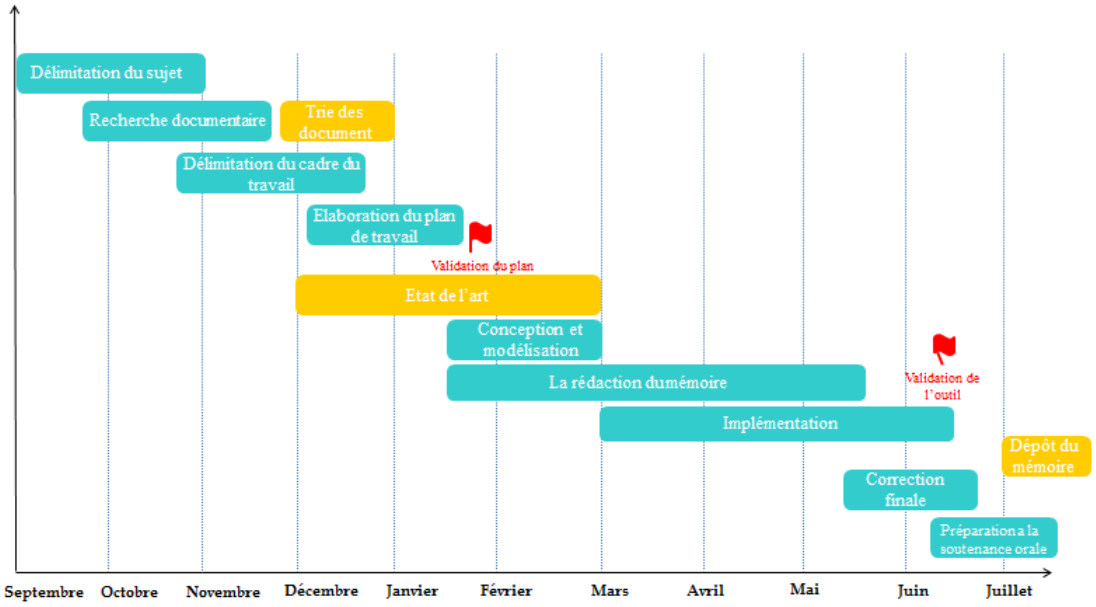


FIGURE 5.2 – Diagramme Gantt : synthèse de déroulement du projet

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 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 les manifestations qui ont été rencontrées et ses perturbations au niveau du travail, 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.

Quatrième partie

Conclusion et perspectives

Chapitre 6

Conclusion générale et Perspectives

« Whoever wants to reach the top of a mountain must take risks by climbing the steep walls. »
— Djamel Fadel

6.1 Conclusion

Dans ce projet de thèse nous avons abordé la problématique de la difficulté d'assimiler le processus d'exécution de requêtes SQL par un étudiant dans le milieu d'enseignement et d'apprentissage ou bien par un DBA chargé par l'administration et le tuning de BD dans un milieu professionnel. Notre objectif consiste à fournir une image mentale qui facilite la compréhension de ce processus d'exécution. Pour répondre à cet objectif, nous sommes tombées dans une approche d'apprentissage basée sur la taxonomie de bloom (voir la figure 3.3 dans le chapitre 3). Cette piste de recherche idéale est encapsulée dans quatre étapes principales qui sont : La connaissance, la compréhension, l'analyse et l'évaluation.

Dans la première étape, nous avons analysé le domaine de l'optimiseur de requête relationnel (par ex MySQL, PostgreSQL) afin d'offrir des concepts de base sur cet optimiseur. La deuxième étape est la conceptualisation du domaine par un métamodèle. **Les deux étapes précédentes correspondent au niveau (1) et (2) dans la taxonomie de Bloom (mémoriser, comprendre).**

La troisième étape est la vision processurale de notre approche, qui consiste à proposer un BPMN (Business Process Modeling an Notation) qui décrit l'usage de notre métamodèle ainsi que le traitement de la requête SQL depuis sa formulation jusqu'à son exécution. Ce BPMN inclut plusieurs tâches à savoir le test de la requête, la vérification de la syntaxe et de la sémantique de la requête et la manipulation des variables système (cache, buffer, tables...). **Cette étape correspond au niveau (4) dans la taxonomie de Bloom (analyser).**

Les actions effectuées par l'apprenant sont stockées dans un fichier de surveille qui indique l'action, le type d'erreur, et les tentatives. Ce fichier permet au tuteur d'apprécier le degré d'assimilation de ce processus par les étudiants, de relier son comportement aux connaissances théoriques et de fournir un feedback pertinent. Et plus précisément c'est comment

donner à l'apprenant la possibilité de s'autoévaluer ou être évalué. **Cette étape correspond au niveau (5) dans la taxonomie de Bloom (évaluer).**

En terme applicatif, nous avons développé un outil prototype sous forme d'une plateforme web pour montrer la faisabilité de l'approche proposée.

De nombreuses perspectives de recherche peuvent être envisagées. Nous présentons succinctement celles qui nous paraissent être les plus intéressantes :

- Evaluer l'usabilité de l'application (apprentissage, nombre erreur, mémorisation etc.).
- Généraliser l'application sur d'autres SGBD (PostgreSQL, Oracle... etc.).
- Augmenter l'interactivité entre l'application et l'utilisateur en ajoutant des modules qui offrent l'image mentale de processus d'exécution des requêtes SQL, par exemple l'utilisation des modules de simulation multimédia.
- Etre utilisée comme un support d'aide à l'apprentissage et l'enseignement pour les apprenants dans le module de bases de données avancées.
- Consolider les efforts de ce projet dans un papier de conférence.

Bibliographie

[1] S. Manegold, P. A. Boncz, and M. L. Kersten. Optimizing database architecture for the new bottleneck : memory access. volume 9, pages 231246, 2000.

[2] M. Rosenmuller, N. Siegmund, H. Schirmeier, J. Sincero, S. Apel, T. Leich,O. Spinczyk, and G. Saake. Fame-dbms : tailor-made data management solutions for embedded systems. In : *Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data management, pages 1-6. ACM, 2008.*

[3] T. Zaschke, S. Leone, T. Gmunder, and M. C. Norrie. Optimising conceptual data models through proffling in : *object databases. In International Conference on Conceptual Modeling, pages 284-297. Springer, 2013.*

[4] P. P.-S. Chen. « The entity-relationship modeltoward a uni$_e$*dviewofdata*˘.*In* : *Readings in arti$_c$ialintelligenceanddatabases.Elsevier*, 1988, *p.*98111(*cf.p.*26).

[5] S. Chaudhuri and V. Narasayya. Autoadmin what-if index analysis utility. In : *ACM SIGMOD Record, 27(2) :367378, 1998.*

[6] C. Maier, D. Dash, I. Alagiannis, A. Ailamaki, and T. Heinis. Parinda : an interactive physical designer for postgresql. In *Proceedings of the 13th International Conference on Extending Database Technology, pages 701704. ACM, 2010.*

[7] A. R. Da Silva. Model-driven engineering : A survey supported by the uniffed conceptual model. Computer Languages, Systems Structures, 43 :139155, 2015.

[8] Huda Al Shuaily, Karen Renaud,« A Framework for SQL Learning : Linking Learning Taxonomy, Cognitive Model and Cross Cutting Factors » In : *World Academy of Science, International Journal of Computer and Systems Engineering, Vol :10, 2016 , No 9 p 1-7*

[9] Kristin Annabel Torjussen Folland,« viSQLizer : An interactive visualizer for learning SQL » In : *Norwegian University, vol.2,2016 p. 1-159.*

[10] Alberto Abell Elena Rodrez, « LEARN-SQL :Automatic Assessment of SQL Based on IMS QTI Specification » in *Universitat Politica de Catalunya, Universitat Oberta de Catalunya, p 1-2*

[11] Andrew Yostt,Paull Wagnerr, « Developing a Dynamic Visualization of The Oracle Query Execution Process » In : *University of Wisconsin-Eau Claire, p.1-5.*

[12] Surajit Chaudhuri. An Overview of Query Opimization In : *Relationnal Systems. In Proc. of ACM PODS,1998.*

[13] Saurabh Gupta et al. A Survey on Query Processing and Optimization In : *Relationnal Database Managment System. Internationnal Journal of Latest Trends in Engineering and Technology(IJLTET).*

[14] Herodotos Herodotou and Shivnath Babu, « Profiling What if Analysis, and Cost based Optimization of MapReduce Programs ».In : *VLDB Endowment, Vol. 4, No. 11.*

[15] Tuyet-Trinh Vu. Une approche pour la construction dluateurs adaptables de requs. Autre [cs.OH]. In : *Institut National Polytechnique de Grenoble - INPG, 2005. Frans. <tel-00009482>.*

[16] D. J. Rosenkrantz et H. B. Hunt III. « Processing conjunctive predicates and queries ». In : *Proceedings*

[17] P. Griffiths Selinger et al. « System R : Relational Approach to Database Management ».In : *ACM Transactions on Database Systems, Vol. 1, No. 2, June 1976, pp.*

[18] S. Navathe and M. Ra. « Vertical partitioning for database design : a graphical algorithm ». In : *ACM SIGMOD, pages 440450, 1989.*

[19] Sana Mallouli. M-modsation du Comportement dun Mod de Processus : Une Drche de Construction dun Moteur dExtion. Modsation et simulation. *Universitnth-Sorbonne - Paris I, 2014. Frans.*

[20] A. Ouared, Y. Ouhammou, and L. Bellatreche. « Costdl : a cost models description language for performance metrics in database. » In : *Proceedings of the 21ST IEEE ICECCS. IEEE, 2016.*

[21] Francois.Jacquenet, 2 partie-Processus de developpement Du Logiciel (Software Process) In : *Facults Sciences et Techniques, page 54*