Ordre...../F.S.S.A/UAMOB/2017

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE AKLI MOAND OULHADJE-BOUIRA

Faculté des Sciences et des Sciences Appliquées
Département Informatique

Mémoire de fin d'étude

Présenté par :

HAOUES MESSAOUDA
BACHA SAADA

En vue de l'obtention du diplôme de **Master 02** en **:**

Filière :INFORMATIQUE
Option : ***Ingénierie des Systèmes d'Informatique et du Logiciel (ISIL)***

Thème :

Application de la Méthode DVFS Dans l'Exploitation du Base de Données

Devant le jury composé de :

Nom et prénom	Grade		
Dr. Abbas Akli	MCB	UAMOB	Président
		UAMOB	Encadreur
		UAMOB	Examinateur
		UAMOB	Examinateur

Année Universitaire 2017/2018

Remerciements

Nous tenons à exprimer nos grandes reconnaissances et nos vifs remerciements à notre promoteur M. ABBAS AKLI, pour la qualité de l'encadrement dont il nous a fait bénéficier, en particulier pour son attention, ses conseils et sa disponibilité dans le travail ainsi que son honnêteté.

Nous tenons aussi de remercier vivement les membres de jury d'avoir accepté de juger ce modeste travail.

Enfin, Nous tenons de remercier tous ceux qui de près ou loin ont bien voulu nous encourager et nous aider pour que ce travail puisse être achevé.

Dédicaces

A la mémoire de mes parents, les regrettés,

Et en particulier à ma mère qui m'a soutenue

Durant toute ma vie.

HAOUES MESSAOUDA

Dédicaces

A mes parents qui ont toujours encouragé ma curiosité intellectuelle,

A mon mari et mes filles Chahinez et Wissam,

A toute ma famille et toutes mes amies.

BACHA SAADA

Table des matières

Table des Figures

Liste des abréviations

ACPI:*Advanced Configuration and Power Interface*

AOL:*Approches Orientées Logicielles*

AOM:*Approches Orientées Matériels*

BDD:*Bases de Données*

BF:*Besoin fonctionnel*

BNF:*Besoin non-fonctionnel*

BNFM*: Besoin non-fonctionnel Mesurable*

BNFNM*: Besoin non-fonctionnel Non Mesurable*

COBOL:*Common Business Oriented Language*

DPM:*mémoire non volatile*

DRAM:*Dynamic Random Access Memory*

DVFS:*Dynamic Voltage and Frequency Scaling*

EA-DVFS:*Energy-aware DVFS*

EDeg:*Earliest Deadline with energy guarantee*

EDF:*Earliest Deadline First*

EE:*l'efficacité énergétique*

EEPROM:*mémoire morte effaçable électriquement et programmable*

FBA:*Frame-Based Algorithm*

GPU:*3- Les unités de traitement graphique*

HA-DVFS:*Harvesting-aware Dynamic Voltage and Frequency Selection Algorithm*

LSA:*Lazy Scheduling Algorithm*

MCD:*modèle conceptuel de données*

MLD:*modèle logique de données*

MPD:*modèle physique des données*

NVM:*mémoire non volatile*

OLAP:*Online Analytical Processing*

OLTP:*Online Transaction Processing*

OS:*Operating system*

PVFC:*Processor Voltage/Frequency Control*

SGBD:*Systèmes de Gestion de Bases de Données*

SIG:*Systèmes d'information géographiques*

SSD:*Solid-State Drive*

WCET:*Estimation of Wind Chill Equivalent Temperatures*

Introduction Générale

Jour après jour, nous prenons l'énergie que nous utilisons pour acquise. Elle intervient dans tous les aspects de nos vies et au fur et à mesure que la demande augmente, les ressources traditionnelles se raréfient. La technologie faisant autant partie du problème que de la solution, nous pensons que l'informatique doit s'imposer comme un leader dans la transition énergétique. Ce secteur est un grand consommateur d'énergie et l'industrie est responsable de l'utilisation en électricité de ses systèmes et appareils.

Avec la démocratisation de « l'informatique as a service », il nous faut aussi veiller à réduire la consommation d'énergie par unité de travail.

Il a été estimé que les équipements informatiques utilisent un total de 10% de la capacité mondiale en génération d'électricité. Si tous ces équipements étaient alimentés par une énergie verte et neutre en carbone cela réduirait certainement notre empreinte sur le globe[1].

Les experts du secteur, tels que le SMARTer 2020, rapportent que la consommation des centres de données mondiaux augmenteront de 7% chaque année jusqu'en 2020 [2], en raison du développement d'Internet et de l'augmentation des besoins de stockage des entreprises.

La consommation énergétique d'un SGBD dépend de la base de données qu'il héberge. Plus précisément, elle dépend de son schéma (le nombre de tables et les attributs), sa population (en termes d'instances) et son exploitation via des requêtes. La plus grosse consommation d'un SGBD est due au calcul effectué par ses composantes principales : l'optimiseur de requêtes, le gestionnaire de buffer, le contrôleur de concurrence, le gestionnaire des méthodes d'accès, le gestionnaire de stockage, etc.

Les SGBD sont les plus gros consommateurs d'énergie électrique. Cela est dû au fait qu'ils effectuent des opérations (jointure, union, agrégation, etc.), impliquant des tables ou des structures de données volumineuses qui nécessitent toutes les ressources d'un centre de données (CPU, mémoire, et réseau). Les chercheurs ont donnée l'importance pour l'énergie consommée par les infrastructures informatique on montrant déférente techniques de réduction énergétique. La technique DVFS une de ces techniques qu'est le sujet de notre mémoire.

Pour mieux présenter notre travail, nous l'avons organisé en quatre chapitres :

Dans **le premier Chapitre**, nous présentons un état de l'art sur le cycle de vie de conception des bases de données avancées, avec une focalisation particulière sur la phase physique. Le chapitre décrit aussi le module de traitement de requêtes dans un SGBD relationnel et propose la conception d'un optimiseur de requêtes éco-énergétique.

Dans **le second Chapitre**, nous présentons un état de l'art sur les différentes techniques d'amélioration de l'efficacité énergétique dans les systèmes informatiques, et dans les bases de données et CPU ainsi que les notions de bases sur l'énergie et les modèles de coût.

Le troisième chapitre est dédié à l'approche de conception de notre système et l'usage et fonctionnalités UML et les types du diagramme utilisé en UML.

Dans **le dernier chapitre**, nous décrivons l'environnement technique de développement ainsi que les principales interfaces de notre application.

En fin, nous concluons ce mémoire par **une conclusion générale.**

Ce mémoire comprend également **l'Annexe** qui liste des requêtes utilisées dans notre application pour économiser l'énergie de CPU.

CHAPITRE 1

LA TECHNOLOGIE DES BASES DE DONNÉES

1.1- Introduction :

Les bases de données ont pris aujourd'hui une place essentielle dans l'informatique, plus particulièrement en gestion. Au cours des trente dernières années, des concepts, méthodes et algorithmes ont été développés pour gérer des données sur mémoire ssecondaires ; ils constituent aujourd'hui l'essentiel de la discipline « Bases de Données » (BDD).

Cette discipline est utilisée dans de nombreuses applications. Il existe un grand nombre de Systèmes de Gestion de Bases de Données (SGBD) qui permettent de gérer efficacement de grandes bases de données. De plus, une théorie fondamentale sur les techniques de modélisation des données et les algorithmes de traitement a vu le jour. Les bases de données constituent donc une discipline s'appuyant sur une théorie solide et offrant de nombreux débouchés pratiques.

Dans ce chapitre, nous présentons un état de l'art portant sur la technologie des bases des données, la définition d'une base de données, l'évolution des modèles de données, son cycle de vie de conception et d'exploitation. Nous focalisant particulièrement sur la phase de conception physique et le traitement de requêtes, nous détaillons les différentes techniques et travaux proposés pour chacune.

1.2- Qu'est-ce qu'une base de données ?

Une base de données (*database* en anglais), permet de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité ; celles-ci peuvent être de natures différentes et plus ou moins reliées entre elles. Dans la très grande majorité des cas, ces informations sont très structurées, et la base est localisée dans un même lieu et sur un même support. Ce dernier est généralement informatisé.

La base de données est au centre des dispositifs informatiques de collecte, de mise en forme, de stockage et d'utilisation d'informations. Le dispositif comporte un système de gestion de base de données (abréviation : SGBD) : un logiciel moteur qui manipule la base de données et dirige l'accès à son contenu. De tels dispositifs - souvent appelés base de données - comportent également des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations.

1.3- Évolution des modèles de données

Avant d'entrer dans le vif du sujet, rappelons brièvement que les bases de données relationnelles occupent toujours une part prépondérante du marché, bien que la notion de *Big Data*. Notons que la gestion de ces données souvent semi-structurées dépend davantage du monde NoSQL qui ne nécessite pas de méthodologie de conception à proprement parler (enfin, jusqu'à présent, mais l'avenir me donnera peut-être tort).

L'informatique existait déjà avant les bases de données relationnelles et les serveurs NoSQL, et Apollo 11 a bien été envoyé sur la Lune en juillet 1969 avant qu'E. Codd ne publie ses écrits sur le

modèle relationnel. À l'époque, la conception des fichiers devait se faire avec bon sens, mais depuis, des méthodes plus formelles ont émergé avec de nombreux livres ou manuels volumineux.

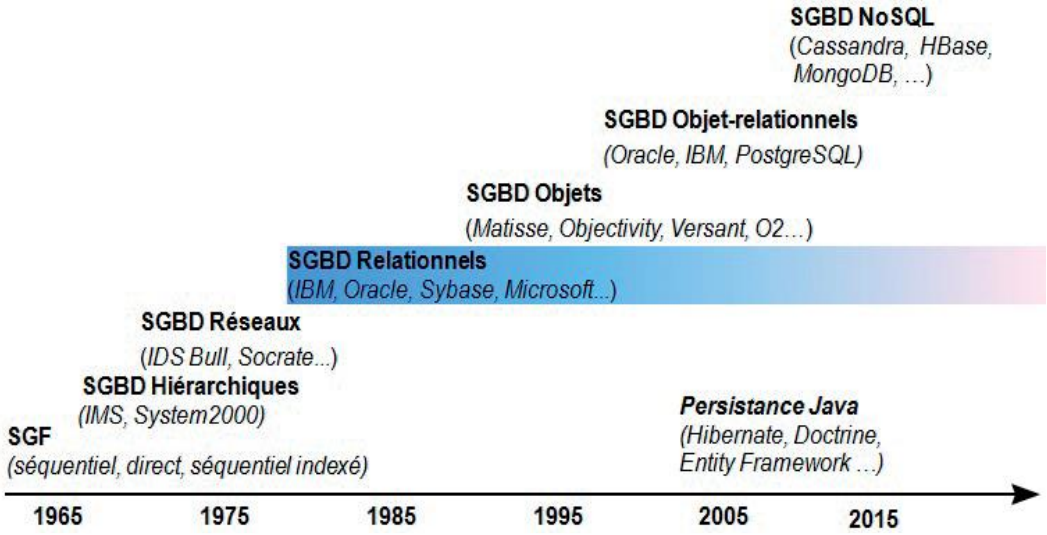

Figure 1. 1 Historique des bases de données

1.3.1- Les fichiers et COBOL

Le stockage des données a commencé dans les années 1960 avec les systèmes de gestion de fichiers et le langage COBOL (*Common Business OrientedLanguage*). Loin d'être dépassé, ce dernier fut le plus utilisé entre 1960 et 1980. En 2002, il permet une programmation de type objet, la gestion des informations Unicode et une intégration avec XML. En 2005, le Gartner Group estimait que COBOL manipulait près de 75 % des données de gestion stockées.

Le principal inconvénient des applications COBOL est la forte dépendance qui existe entre les données stockées et les traitements. En effet, le fait de déclarer dans chaque programme les fichiers utilisés impose une maintenance lourde si la structure d'un fichier doit être modifiée.

De plus, les instructions de manipulation (ouverture, lecture, écriture et modification) sont très liées à la structure de chaque fichier. La structure des fichiers de données s'apparente à celle d'une table (suite de champs de types numériques ou alphanumériques).

1.3.2- Le modèle hiérarchique

Les bases de données hiérarchiques ont introduit un modèle de données du même nom. Il s'agitde déterminer une arborescence de données où l'accès à un enregistrement de niveau inférieurn'est pas possible sans passer par le niveau supérieur. Promus par IBM et toujours utilisés dansle domaine bancaire, les SGBD hiérarchiques souffrent toutefois de nombreux inconvénients.

La figure suivante illustre un modèle hiérarchique de données dans lequel des compagniesaériennes peuvent embaucher plusieurs pilotes. Un pilote peut travailler pour le compte dedifférentes compagnies.

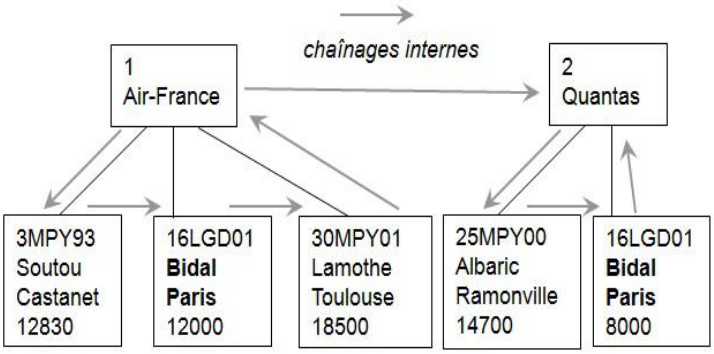

Figure 1. 2 Modèle de données hiérarchique

Les inconvénients récurrents sont toujours la forte dépendance entre les données stockées et les méthodes d'accès. Les chaînages internes impliquent forcément une programmation complexe.

1.3.3- Le modèle réseau

Quelques années plus tard, C. W. Bachman, pionnier dans le domaine de l'informatique, s'est essayé aux bases de données en inventant un modèle brisant cette hiérarchie plutôt arbitraire. Les bases de données réseau étaient nées avec le modèle CODASYL, première norme décidée sans IBM. Bien que résolvant quelques limitations du modèle hiérarchique et annonçant des performances en lecture honorables, le modèle réseau n'est ni plus ni moins qu'une usine à gaz gavée de pointeurs. Pour preuve, plus personne n'utilise de tels SGBD où la dépendance entre les données stockées et les méthodes d'accès existe toujours, et l'évolution d'une base de données est très coûteuse en termes de recompilation de pointeurs.

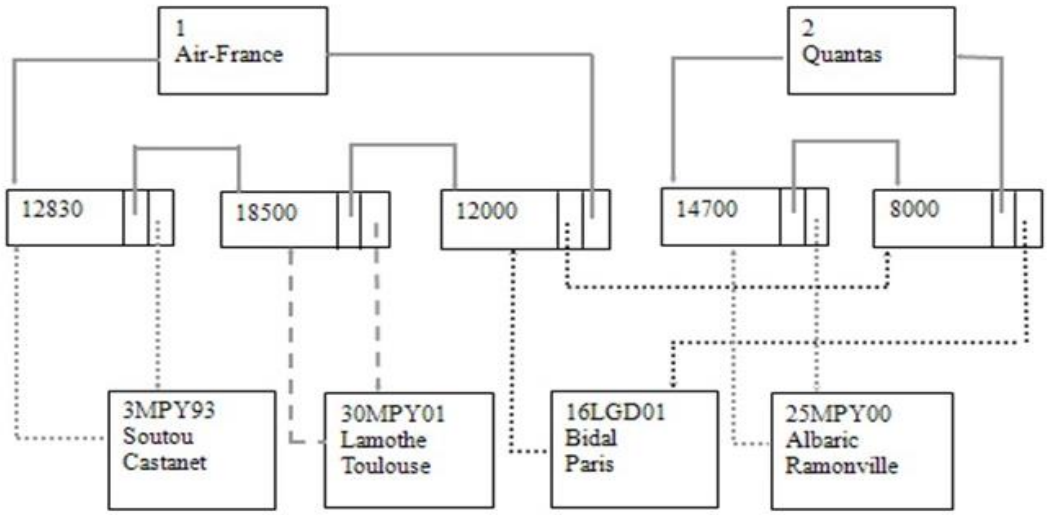

Figure 1. 3 Modèle de données réseau

Soyons honnêtes, le monde ressemble bien à une telle usine à gaz ! Mais pas question de stocker ainsi les données, ce serait bien trop compliqué de concevoir le bon graphe. Le modèle de données se doit d'être plus simple.

1.3.4- Le modèle relationnel :

En 1970, E. Codd publie l'article de référence posant les bases du modèle relationnel [COD70]. D'un seul coup, toutes les limitations des précédents modèles sont résolues. Le but initial de ce modèle était d'améliorer l'indépendance entre les données et les traitements. Cet aspect des choses est réussi et avec ça d'autres fonctionnalités apparaissent :

- Normalisation (dépendances fonctionnelles) et théorie des ensembles (algèbre relationnelle).
- Cohérence des données (non-redondance et intégrité référentielle).
- Langage SQL (déclaratif et normalisé).
- Accès aux données optimisé (choix du chemin par le SGBD).
- Indexation, etc.

Les liens entre les enregistrements de la base de données sont réalisés non pas à l'aide de pointeurs physiques, mais à l'aide des valeurs des clés étrangères et des clés primaires. Pour cette raison, le modèle relationnel est dit « modèle à valeurs ».

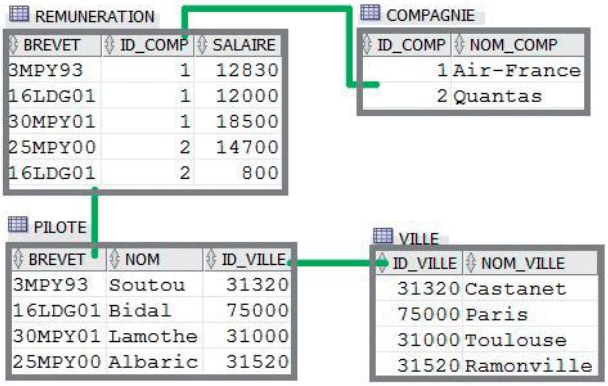

Figure 1. 4 Modèle de données relationnel

La force de ce modèle de données réside dans le fait qu'il repose sur des principes simples et permet de modéliser des données complexes. Le modèle relationnel est à l'origine du succès que connaissent aujourd'hui les grands éditeurs de SGBD, à savoir Oracle, IBM, Microsoft et Sybase dans différents domaines :

- OLTP (Online Transaction Processing) où les mises à jour des données sont fréquentes, les accès concurrents et les transactions nécessaires.
- OLAP (Online Analytical Processing) où les données sont multidimensionnelles (cubes), les analyses complexes et l'informatique décisionnelle.
- Systèmes d'information géographiques (SIG) où la majorité des données sont exprimées en 2D ou 3D et suivent des variations temporelles.

1.3.5- Les modèles NoSQL :

Depuis le graphe ci-dessous, plus le modèle est complexe, moins le système est apte à évoluer rapidement en raison de la montée en charge.

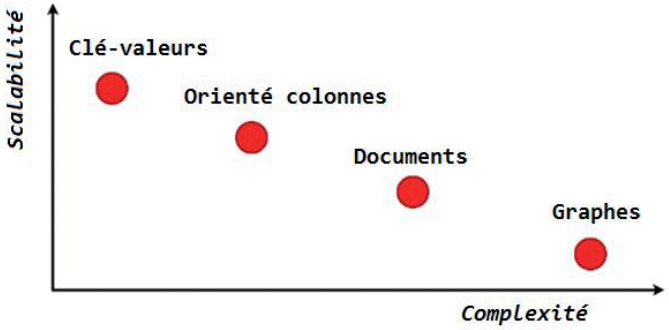

Figure 1. 5 Modèles de données du NoSQL

1.3.6- Modèle de données clé-valeurs

Le mode de stockage du modèle clé-valeurs (key-value) s'apparente à une table de hachage persistante qui associe une clé à une valeur (de toute nature et de type divers, la clé 1 pouvant référencer un nom, la clé 2 une date, etc.). C'est à l'application cliente de comprendre la structure de ce blob. L'intérêt de ces systèmes est de pouvoir mutualiser cette table sur un ou plusieurs serveurs. Les SGBD les plus connus sont Memcached, CouchBase, Redis et oldemort(LinkedIn).

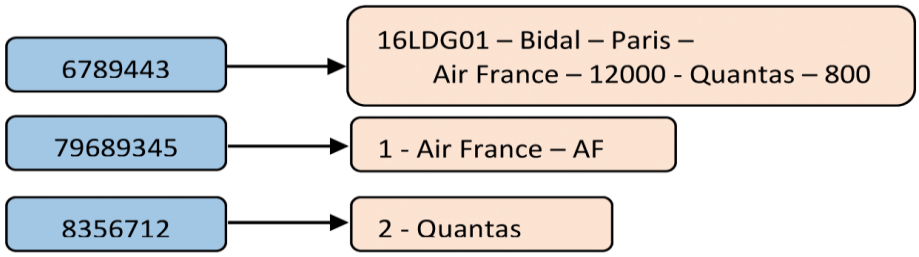

Figure 1. 6 Modèle de données clé-valeurs

1.3.7- Modèle de données orienté colonnes

Le modèle orienté colonnes ressemble à une table dénormalisée (sans la présence de NULL,toutefois) dont la structure est dynamique. Les SGBD les plus connus sont HBase (implémentation du BigTable de Google) et Cassandra (projet Apache qui reprend à la fois l'architecture de Dynamo d'Amazon et BigTable).

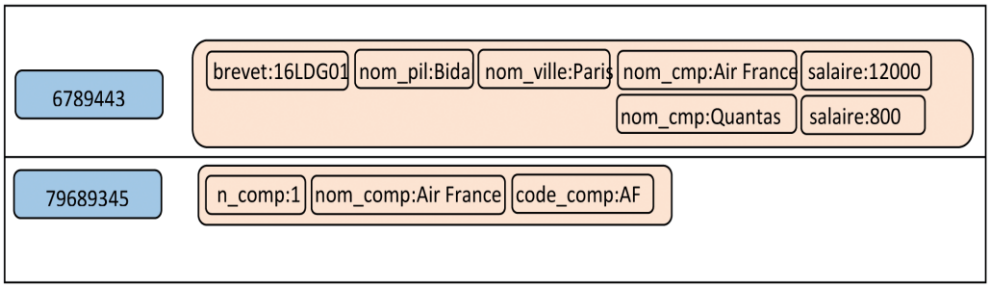

Figure 1. 7 Modèle de données orienté colonnes

1.3.8- Modèle de données orienté documents

Le modèle orienté documents est toujours basé sur une association clé-valeur dans laquelle la valeur est un document (JSON généralement, ou XML). Les implémentations les plus populaires sont CouchDB (Apache), RavenDB, Riak et MongoDB.

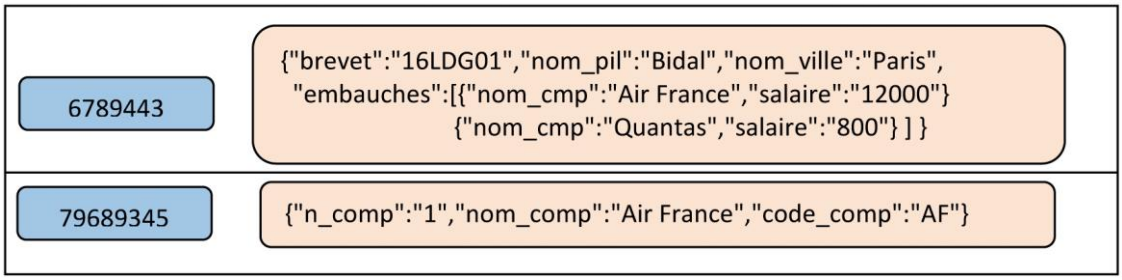

Figure 1. 8 Modèle de données orienté documents

1.3.9- Modèle de données orienté graphes

Le modèle de données orienté graphes se base sur les nœuds et les arcs orientés et éventuellement valués. Ce modèle est très bien adapté au traitement des données des réseaux sociaux où on recherche les relations entre individus de proche en proche. Les principales solutions du marché sont Neo4j (Java) et FlockDB (Twitter).

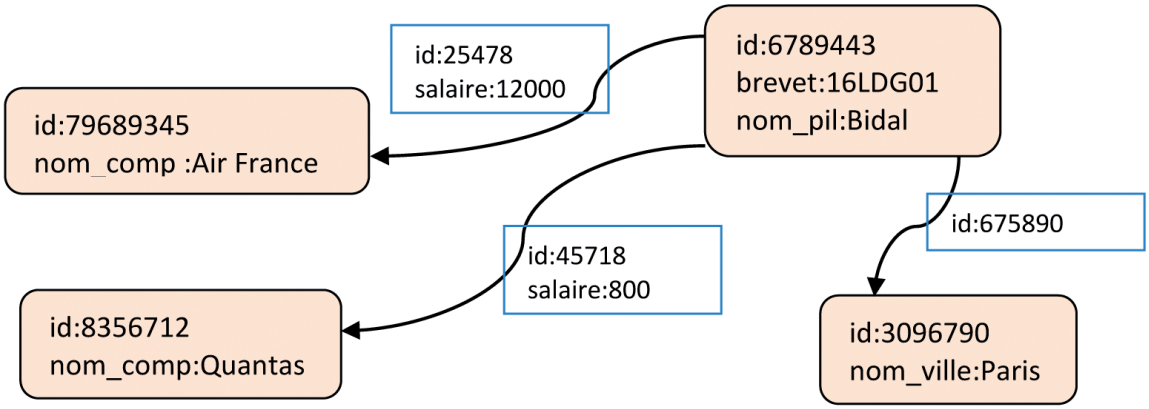

Figure 1. 9 Modèle de données en graphes

Afin d'intégrer l'énergie dans les BDs, nous proposons de les étudier suivant deux dimensions :

(1) le cycle de vie de la conception et (2) le traitement des requêtes dans un SGBD relationnel. Nous Détaillons chacune de ces dimensions dans les prochaines sections.

1.4- Conception et cycle de vie :

Une des tâches essentielles des développeurs de bases de données est la conception de leurs schémas.

La conception de base de données est le processus de production d'un modèle de données détaillé de la base de données. Ce modèle de données contient tous les choix de conception logique et physique nécessaires et les paramètres de stockage physique nécessaires pour générer une conception dans un langage de définition de données, qui peut ensuite être utilisé pour créer une base de données. La

représentation doit être juste pour éviter les erreurs sémantiques, notamment dans les réponses aux requêtes. Elle doit aussi être complète pour permettre le développement des programmes d'application souhaités. Elle doit enfin être évolutive afin de supporter la prise en compte rapide de nouvelles demandes

La démarche de conception et de cycle de vie d'une base de données s'effectue par des abstractions successives et itératives, en descendant depuis les problèmes de l'utilisateur vers la base dedonnées finale [3]. Nous proposons de distinguer cinq étapes :

1.4.1- Analyse des besoins :

L'objectif général de l'analyse des besoins (étude initiale) consiste à : analyser la situation de l'entreprise, définir les problèmes et les contraintes, fixer les objectifs et définir la portée et les limites de la BD par rapport à l'organisation, les fonctionnalités, le matériel, etc. La phase d'analyse du cycle de vie est, en fait, une vérification approfondie des besoins des utilisateurs. Le résultat est une spécification textuelle de ces besoins.

Une bonne analyse des besoins est fondamentale pour une conception réussie de la base de données. Les besoins se répartissent généralement en deux types : besoins fonctionnels (BF) et besoins non-fonctionnels (BNF) tel que représenté dans la Figure 2.3.

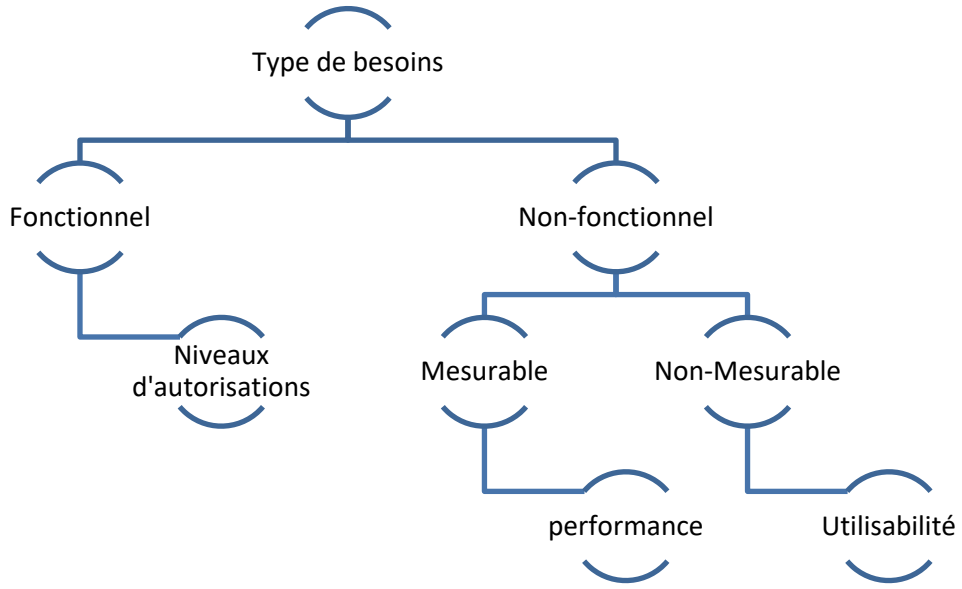

Figure 1. 10 classification des types de besoins

1.4.1.1- Besoin fonctionnel :

Le BF est défini comme étant tout besoin qui spécifie ce que le système doit faire. En d'autres termes,un BF décrira un comportement particulier du fonctionnement du système lorsque certaines conditionssont valides, par exemple : « envoyer un courrier électronique lorsqu'un nouveau client s'inscrit ». Desexemples des BFs dans une base de données incluent : gestion de transactions, authentification, niveauxd'autorisation, exigences de certification, données historiques etc.

1.4.1.2- Besoin non-fonctionnel :

Le BNF est définie comme étant tout besoin qui spécifie comment le système effectue une certainefonction. En d'autres termes, un BNF décrira comment un système doit se comporter et quelles limitesil y a sur safonctionnalité. Les BNFs spécifient généralement les attributs ou caractéristiques de qualitédu système, par exemple : « les données modifiées dans une base de données doivent être mises à jourpour tous les utilisateurs qui y accèdent en moins de 2 secondes ». Par exemple, citons : performance,temps de réponse, consommation d'énergie, évolutivité, disponibilité, fiabilité, la sécurité, utilisabilité,etc.

En outre, les BNFs peuvent être classés en deux catégories : BNF mesurable (BNFM) et BNFnon-mesurable (BNFNM) .

- **Besoin non-fonctionnel Mesurable BNFM :**

Les BNFM sont des besoins qui peuvent être mesurés sur une échelle métrique définie par l'utilisateur.

La métrique est définie par l'utilisateur car pour certains besoins, tel que la performance, il existe plusieurs métriques pour la quantifier. Les mesures peuvent être réalisées automatiquement par un ordinateur comme le temps de réponse, ou manuellement par un dispositif de mesure comme la consommation d'énergie.

- **Besoin non-fonctionnel Non Mesurable BNFNM :**

Cette classe contient des besoins qui ne peuvent être décrits que qualitativement à l'aide d'une échelle ordinaire ou nominale, c'est-à-dire qu'il n'existe aucune métrique à partir de laquelle nous pouvons extraire des mesures quantifiables. Par exemple, considérer l'utilisabilité.

1.4.2- Modélisation conceptuelle :

Le fruit du design conceptuel est un modèle conceptuel dedonnées(MCD) qui décrit graphiquement (par une abstraction) et textuellement les principales entités, attributs, relations et contraintes du monde réel de la façon la plus réaliste possible.

Pour ce faire, les concepteurs utilisent des constructeurs fournis par un modèle de données d'un haut niveau d'abstraction, indépendant donc de toute contrainte d'implémentation. Ils doivent également se baser sur la spécification des besoins collectés auprès des utilisateurs du système pendant la phase d'analyse précédente, à laquelle le domaine de l'ingénierie des besoins s'est associé. D'ailleurs, nombreux sont les concepteurs qui ont tendance à considérer la phase d'analyse (ou du moins une partie d'elle) comme première étape du processus de conception, vu leur chevauchement [4].

Parmi les modèles de haut niveau les plus répandus, on peut citer : E/A, le digramme declasses d'UML [1] et lemodèle Express[2].

[1]http://www.uml.org/
[2]http://www.omg.org/spec/EXPRESS/

Parallèlement, les programmeurs et analystes doivent analyser la partie applicative qui couvreles procédures aidant à réaliser les traitements de la BD et/ou transformer les données en informationsutiles et exploitables. Cela peut inclure la tâche d'identification des processus quiconsistent en un ensemble d'opérations complexes. Les entrées/sorties ainsi que le comportementfonctionnel de chacune des procédures/opérations doivent être spécifiés [5].

Des outils etnotations sont utilisés pour spécifier les processus comme BPwin, les outils demodélisation deworkflow, certains diagrammes UML (comme le diagramme d'activité et le diagramme de séquence),ou certains modèles de la méthode Merise qui permettent de représenter les traitementsde l'application au niveau conceptuel comme le modèle conceptuel de traitements. La conceptiondoit également tenir compte de modifications futures, et veiller à ce que l'investissementdes entreprises dans les ressources d'information perdurera. Tout ce processus peut se résumeren trois tâches :

(i) analyse de données et d'exigences,

(ii) modélisation et normalisation,

(iii) vérification/test du MCD.

Selon l'étendue du système, la structure centralisée ou distribuée de la BD et les préférencesdu concepteur, il existe deux approches classiques de conception :

(i) Une approche descendante (top-down) qui commence par identifier les entités (ensembles de donnée) puis définit les attributs(éléments) pour chaque entité,

(ii) Ou inversement une approche ascendante (bottom-up)où l'on commence par identifier les attributs puis on les regroupe dans des entités. Cela dit, lesdeux approches sont complémentaires puisque l'approche ascendante est souvent utilisée pourles BD distribuées ou de petites BD avec peu d'entités, attributs, relations et transactions, etl'approche descendante dans le cas contraire [6].

1.4.3- Modélisation logique :

Comme mentionné auparavant, l'objectif de cette étape est de concevoir une BD à l'échelle de l'entreprise basée sur un modèle de données particulier qui soit indépendant de tout détail physique d'implémentation. Concrètement, le concepteur établit un modèle logique de données (MLD) à partir du MCD. Pour ce faire, il existe des étapes à suivre de façon itérative.

(i) Tousles objets (entités et contraintes) du MCD sont traduits vers les constructeurs du modèle logiquechoisi. Cette traduction se fait de manière directe voire automatique, en suivant des règlesde traduction prédéfinies.

(ii) Valider le MLD par la normalisation et l'intégrité référentiellesurtout dans le cas des modèles relationnels, et d'une manière générale, vérifier sa cohérenceaprès la phase de traduction, qui peut être amenée à ajouter de nouveaux attributs voire denouvelles tables.

(iii) Valider les contraintes d'intégrité : définition des domaines des attributs,des énumérations, des attributs obligatoires et optionnels, des droits d'accès, etc. La dernièreétape est le test qui confronte le MLD obtenu aux besoins des utilisateurs en termes de données,transactions, et de sécurité.

1.4.4- Modélisation physique :

La conception physique est un processus technique qui gère non seulement l'accessibilitéaux données présentes sur le(s) support(s) de stockage mais surtout la performance du système,l'intégrité et la sécurité des données. Elle s'appuie sur le MLD et fournit le modèle physique desdonnées (MPD) qui dépend du SGBD, du système d'exploitation, et du matériel de stockage.Elle comprend les étapes suivantes :

(i) définir l'organisation du stockage des données qui passe par l'évaluation du volume, la fréquence des mises à jour, la localisation des tables, l'identification des index, de leurs types et des vues matérialisées [7],

(ii) définir les mesures de sécurité et d'intégrité comme, entre autres, les groupes des utilisateurs, et l'ensemble des privilèges affectés à chaque individu ou groupe

(iii) déterminer les mesures de performance en prenant en compte des caractéristiques du support de stockage, tels que le temps de recherche, la taille du secteur/bloc (page), la taille de la mémoire tampon, et le nombre de plateaux de disque et de têtes de lecture/écriture.

1.4.5- Déploiement et maintenance :

Le résultat des phases de conception de base de données estune série d'instructions détaillant la création de tables, d'attributs, d'index, de contraintes desécurité, de règles de stockage, de performance, 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 fonctionnecomme prévu. L'administrateur doit aussi effectuer des activités de maintenance dans la basede données. Certaines des activités de maintenance périodique comprennent la sauvegarde, larestauration, la modification des attributs et tables, génération de statistiques, audits de sécurité,etc.

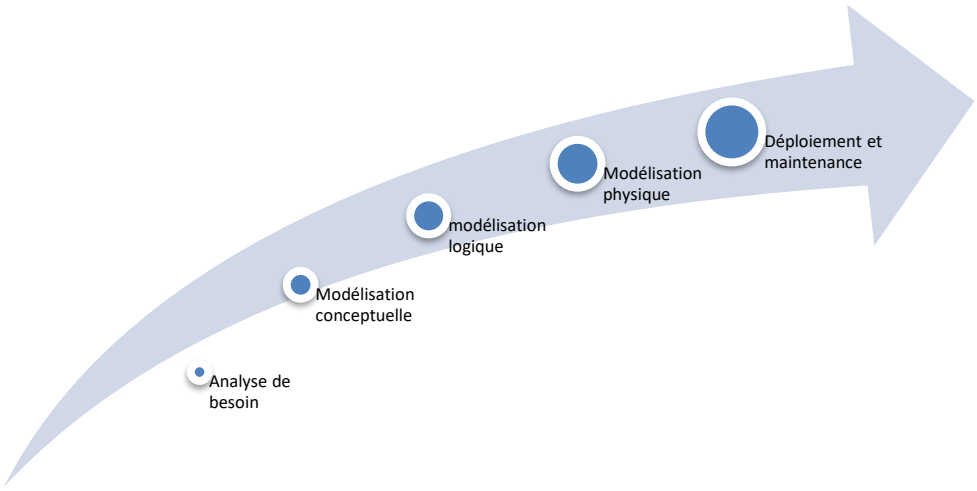

Figure 1. 11 cycle de vie d'une BDD

1.5- Traitement de requêtes dans un SGBD relationnel :

Après la conception et la création de la base de données, les utilisateurs peuvent exploiter les données.

Le traitement de requêtes est le groupe de composants d'un SGBD qui transforme les requêtes desutilisateurs et les commandes de modification des données en une séquence d'opérations de base dedonnées et exécute ces opérations [8]. Notre but est d'examiner le processus de traitement de requêtesdans un SGBD pour des opportunités de minimisation d'énergie.

Dans ce chapitre, nous discuterons des techniques utilisées en interne par un SGBD pour traiterdes requêtes de haut niveau. Une requête exprimée dans un langage de requête de haut niveau commeSQL doit d'abord être analysée syntaxiquement. L'analyseur identifie les mots clés de la requête (lesmots clés SQL, les nomsd'attributs et les noms de relations) qui apparaissent dans le texte de la requête,et vérifie la syntaxe de requête pour déterminer si elle est formulée selon les règles de syntaxe (règlesde grammaire) du langage de requête. La requête doit également être validée en vérifiant que tous lesnoms d'attributs et de relations sont des noms valides et sémantiquement significatifs dans le schéma dela base de données. En utilisant des règles de transformation, une représentation interne de la requêteest ensuite créée, habituellement sous la forme d'une structure de données arborescente appelée arbrealgébrique. Le SGBD doit alors concevoir une stratégie d'exécution ou un plan de requête pour extraire lesrésultats de la requête à partir des fichiers de base dedonnées. Une requête a de nombreuses stratégiesd'exécution possibles, et le processus de choisir un approprié pour traiter une requête est connu sousle nom d'optimisation de requête [9].

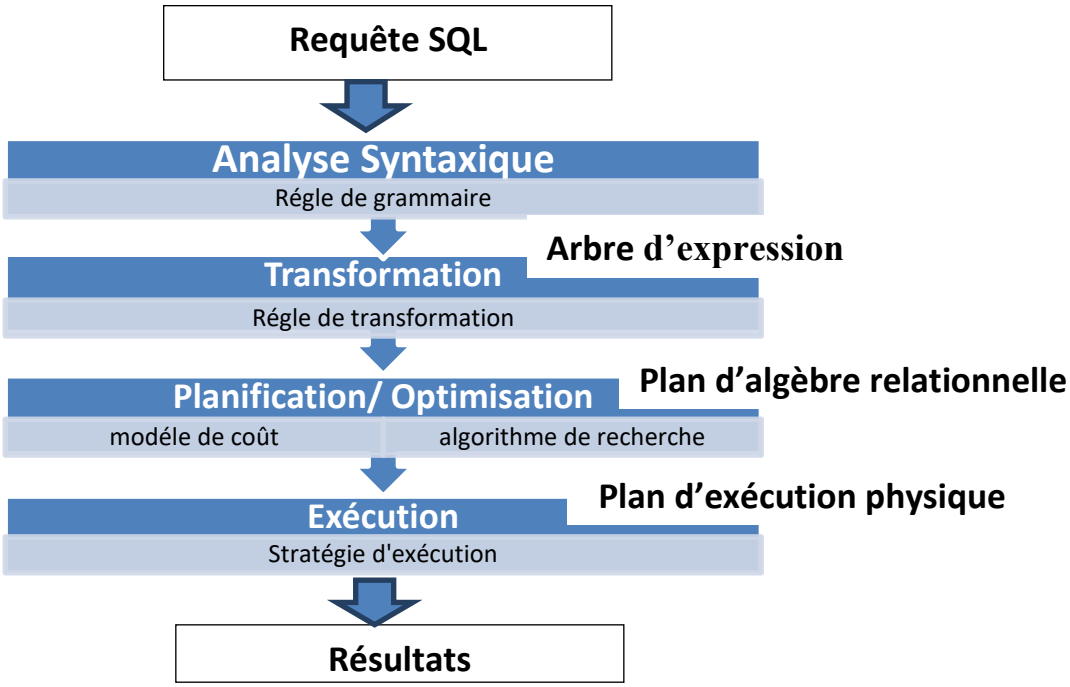

Figure 1. 12 Architecture globale pour le traitement de requêtes dans un SGBD.

1.5.1- Analyse

Le travail de l'analyseur est de prendre du texte écrit dans une langue comme SQL et de le convertiren un arbre d'analyse, qui est un arbre dont les nœuds correspondent soit à :

(1) atomes, qui sont des éléments lexicaux tels que des mots clés (par exemple, SELECT), des noms d'attributs ou de relations, des constantes, des parenthèses, des opérateurs tels que + ou <, et d'autres éléments de schéma,

(2) ou des catégories syntaxiques, qui sont des noms pour des sous-parties de requête qui ont un rôle similaire. Ils sont représentés par un nom descriptif entre les symboles <et >[11]. Par exemple, <Requête>est utilisé pour représenter une requête, et <Condition>représentera toute expression qui est une condition. Nous présenterons ces propos par un exemple dans la section suivante.

Une grammaire pour un sous-ensemble simple de SQL

Traduction algébrique

• Produire une expression algébrique équivalente à la requête SQL

– Clause SELECT opérateur de projection

– Clause FROM les relations qui apparaissent dans l'expression

– Clause WHERE

• Condition "Attr = constante" opérateur de sélection

• Condition "Attr1 = Attr2" jointure ou sélection

• Résultat: expression algébrique

– Représentée par un arbre de requête = plan d'exécution de l'expression algébrique relationnelle

– Point d'entrée dans la phase d'optimisation

Exemple de traduction algébrique

• Soit le schéma relationnel (notation simplifiée) :

Cinéma (ID-cinéma, nom, adresse)

Salle (ID-salle, ID-cinéma, capacité)

Séance (ID-salle, heure-début, film)

• Requête: quels films commencent au Multiplex à 20 heures?

SELECT Séance.film FROM Cinéma, Salle, Séance

WHERE Cinéma.nom = 'Multiplex' AND

Séance.heure-début = 20 AND

Cinéma.ID-cinéma = Salle.ID-cinéma AND

Salle.ID-salle = Séance.ID-salle

• **Expression algébrique**

Πfilm (σnom = 'Multiplex' $\wedge$ heure-début = 20((Cinéma $\bowtie$ Salle) $\bowtie$ Séance)

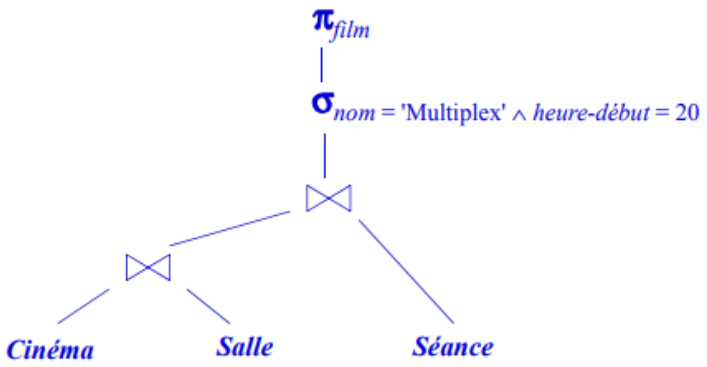

Figure 1. 13 Arbre de requête

L'analyseur doit :

- **Vérifiez les relations**: Chaque relation mentionnée dans une clause FROM doit être une relation ou une vue dans le schéma de la base de données courante.
- **Vérifiez et résoudre les attributs**: Chaque attribut mentionné dans la clause SELECT ou WHERE doit être un attribut d'une certaine relation dans la requête. Par exemple, l'attribut Nom est un attribut de Clients, donc l'analyseur valide cette utilisation. Il vérifie également l'ambiguïté, signalant une erreur si un attribut est dans la portée de deux ou plusieurs relations.
- **Vérifiez les types :** Tous les attributs doivent être d'un type approprié à leurs utilisations. Par exemple, heure-débutest utilisé dans une comparaison =, ce qui nécessite que cet attribut

soit une chaîne ou un type qui peut être contraint à une chaîne. De même, les opérateurs sont vérifiés pour voir qu'ils s'appliquent aux valeurs des types appropriés et compatibles.

1.5.2- Transformation

Dans cette étape, l'arbre d'analyse résultant de l'étape précédente est transformé en une expression de l'algèbre relationnelle étendue. L'objectif de la transformation des requêtes est : (1) la construction d'un arbre standardisé pour l'optimisation des requêtes (standardisation),
(2) l'élimination de la redondance (simplification),
(3) et la construction d'une expression améliorée (amélioration) [10].

Nous verrons comment appliquer des heuristiques qui améliorentl'expression algébrique de larequête, en utilisant certaines des nombreuses règles algébriques issues de l'algèbre relationnelle [9]. Àtitre préliminaire, cette section répertorie les règles algébriques qui transforment un arbre d'expressionen un arbre d'expression équivalent qui peut avoir un plan de requête physique plus efficace. Le résultatde l'application de ces transformations algébriques est le plan de requête logique qui est la sortie decette phase.

1.5.2.1- Règles algébriques pour améliorer les plans de requête

- Une règle est dite **commutative** sur un opérateur si le résultat reste invariable si l'on intervertit les arguments de l'opérateur. Par exemple, + et × sont des opérateurs commutatifs d'arithmétique. Plus précisément, x + y = y + x et x _y = y _x pour tout nombre x et y.D'autre part, - n'est pas un opérateurarithmétique commutatif : x ×y ≠y ×x.
- Une règle est dite **associative** sur un opérateur si nous pouvons regrouper deuxutilisations del'opérateur soit de la gauche ou de la droite. Par exemple, + et _ sont des opérateurs arithmétiquesassociatifs, ce qui signifie que (x + y) + z = x + (y + z)et(x × y)× z = x ×(y × z). D'autre part, - n'estpas associatif : (x × y)× z ± x ×(y × z).

Nous allons présenter quelques règles de transformation de certains opérateurs SQL [9]:

1. **Décomposition de σ** : Une condition de sélection conjonctive peut être décomposée en une cascade (c'est-à-dire une séquence) d'opérations σ individuelles :
$$\sigma_{c_1\ And\ c_2\ And \ldots And\ c_n}(R) \equiv \sigma_{c_1}\left(\sigma_{c_2} \ldots (\sigma_{c_n}(R)) \ldots\right)$$
2. **Commutativité de σ** : L'opération de sélection σ est commutative :
$$\sigma_{c1}\left(\sigma_{c2}(R)\right) \equiv \sigma_{c2}\left(\sigma_{c1}(R)\right)$$
3. **Décomposition de π:** Dans une décomposition (séquence) d'opérations de projectionπ, tous sauf le dernier peuvent être ignorés :
$$\pi_{liste1}\left(\pi_{liste2}(\ldots\left(\pi_{liste\ n}(R)\right)\ldots)\right) \equiv \pi_{liste1}(R)$$
4. **Commuter σ avec π** : Si la condition de sélection c ne concerne que les attributsA_1,…, $A_n$dans la liste de projection, les deux opérations peuvent être commutées :

$$\pi_{A_1,A_2,\ldots,A_n}\ \left(\sigma_c(R)\right) \equiv \sigma_c\left(\pi_{A_1,A_2,\ldots,A_n}(R)\right)$$

5. **Commutativité de ⋈ (et ×) :** L'opération de jointure est commutative, de même que l'opérationde produit cartésien × :

$$R \bowtie_c S \equiv S \bowtie_c R$$

$$R \times S \equiv S \times R$$

6. **Commuter σ avec ⋈ (ou×) :** Si tous les attributs de la condition de sélection c ne comportent que les attributs d'une des relations jointes, par exemple R, les deux opérations peuvent être commutées comme suit :

$$\sigma_c(R \bowtie S) \equiv \left(\sigma_c(R)\right) \bowtie S$$

7. **Commuter π avec ⋈ (ou ×) :**Supposons que la liste de projection est L = $A_1,\ldots,A_n,B_1,\ldots,B_m$, où $A_1,\ldots,A_n$ sont des attributs de R et $B_1,\ldots,B_m$ sont des attributs de S. Si la condition de jointure c implique seulement les attributs en L, les deux opérations peuvent être commutées comme suit :

$$\pi_L(R \bowtie_c S) \equiv \left(\pi_{A1,\ldots An}(R)\right) \bowtie_c \left(\pi_{B1,\ldots,Bm}(S)\right)$$

8. **Commutativité des opérations ensemblistes** : Les opérations ensemblistes ∩et ∪ sont commutatives, mais - ne l'est pas.
9. **Associativité de ⋈ , ×∪et ∩**Ces quatre opérations sont individuellement associatives ; C'est-à dire si les occurrences de θ représentent la même opération qui est l'une de ces quatre opérations, on a :

$$(R\ \theta\ S)\ \theta\ T \equiv R\ \theta\ (S\ \theta\ T)$$

10. **Commuter σ avec des opérations ensemblistes.**L'opération σ commute avec ∪, ∩ et -. Si θ représente l'une de ces trois opérations dans une expression, on a :

$$\sigma_c(R\ \theta\ S\ ) \equiv \left(\sigma_c(R)\right)\ \theta\left(\sigma_c(S)\right)$$

11. **L'opération π commute avec ∪ :**

$$\pi_L(R\ \cup\ S\ ) \equiv \left(\pi_L(R)\right)\ \cup\left(\pi_L(S)\right)$$

12. **Conversion d'une séquence (σ, ×) en ⋈** : Si la condition c d'une σ qui suit un × correspond à une condition de jointure, convertissez la séquence (σ, ×) en une ⋈ comme suit :

$$\left(\sigma_c(R\ \times\ S\ )\right) \equiv (R\ \bowtie_c\ S\ )$$

13. **Descendre σ en bas en conjonction avec la différence ensembliste.**

$$\sigma_c(R - S\ ) \equiv \left(\sigma_c(R)\right) - \left(\sigma_c(S)\right)$$

14. **Descendre σ à un seul argument dans ∩** : Si dans la condition σ_c tous les attributs sont de la relation R, alors :

$$\sigma_c(R \cap S\ ) \equiv \left(\sigma_c(R)\right) \cap \left(\sigma_c(S)\right)$$

15. **Élimination des doubles :**L'opérateur σ élimine les doublons d'un ensemble. Si θ représentel'une de ces trois opérations :⋈ , $\bowtie_c$ ou× dans une expression, alors :

$$\delta(R\ \theta\ S\ ) \equiv \big(\delta(R)\big)\ \theta\ \big(\delta(S)\big)$$

$$\delta\big(\sigma_c(R)\big) \equiv \sigma_c\big(\delta(R)\big)$$

16. **Règles de groupement et d'agrégation** : L'opérateur γa plusieurs règles, par exemple, il absorbe δ. Nous pouvons projeter des attributs inutiles avant d'appliquer l'opérationδ. Nous pouvons également supprimer les doublons dans le cas d'agrégation de type MIN et MAX :

    $\delta\big(\gamma_L(R)\big) \equiv \gamma_L(R)$

    $\gamma_L(R) \equiv \gamma_L\big(\pi_M(R)\big)$, Si M est une liste contenant tous les attributs mentionnés dans L.

    $\gamma_L(R) \equiv \gamma_L\big(\pi_M(R)\big)$; L est MIN et/ou MAX.

17. **Quelques transformations triviales.**

    Si S est vide, alors R ∪ S = R

    Si la condition c dans σ_c est vraie pour tout R, alors $\sigma_c(R)$= R

1.5.2.2- Amélioration du plan de requête logique

Il existe un certain nombre de règles algébriques qui améliorent les plans de requêtes logiques. Les règles suivantes sont les plus couramment utilisées par les optimiseurs [9] :

- Les sélections peuvent être poussées vers le bas dans l'arborescence d'expression dans la mesure du possible. Si une condition de sélection est composée de AND de plusieurs conditions, nous pouvons diviser la condition et pousser chaque partie vers le bas de l'arbre séparément. Cette stratégie est probablement la technique d'amélioration la plus efficace.
- De même, les projections peuvent être poussées vers le bas de l'arbre, ou de nouvelles projections peuvent être ajoutées.
- Les opérations d'éliminations des doubles peuvent parfois être supprimées ou déplacées vers une position plus pratique dans l'arborescence.
- Certaines sélections peuvent être combinées avec un produit ci-dessous pour transformer la paire d'opérations en une jointure, ce qui est généralement beaucoup plus efficace à évaluer que les deux opérations séparément.

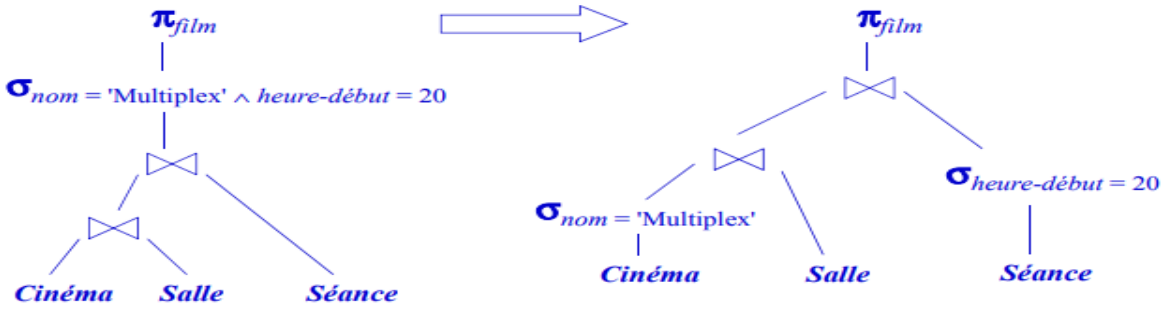

Figure 1. 14 Transformation du plan de requête logique

1.5.3- Génération des plans et optimisation

Après avoir analysé et transformé une requête en un plan de requête logique, la prochaine étape consiste à transformer le plan logique en un plan physique. Cela se fait normalement en considérant plusieurs plans physiques qui sont dérivés du plan logique, et en évaluant ou en estimant le coût de chacun. Après cette évaluation, souvent appelée génération de plans basée sur le modèle coût [12], le plan de requête physique avec le coût le moins estimé est choisi. Ce plan est passé au moteur d'exécution de requêtes [9].

Le premier problème est de savoir comment estimer les coûts des plans avec précision. Ces estimations sont basées sur :

(1) des statistiques sur données de la base, et

(2) des statistiques de système. Compte tenu des valeurs de ces paramètres, nous pouvons faire un certain nombre d'estimations raisonnables de la taille des relations qui peuvent être utilisés pour prédire le coût d'un plan physique complet.

1.5.3.1- Statistiques sur les données

Étant donné un arbre d'opérateur d'une requête, il est d'une importance fondamentaled'être enmesure d'évaluer avec précision et efficacité son coût. L'estimation des coûts doit être précise parce quel'optimisation est seulement aussi bonne que ses estimations de coûts. L'estimation des coûts doit êtreefficace car elle est dans la boucle interne de l'optimisation des requêtes et est invoquée à plusieursreprises [13]. Un SGBD moderne permet généralement à l'utilisateur ou à l'administrateur de demanderexplicitement la collecte de statistiques telles que T(R), le nombre de tuples dans une relation R, etV (R,a), le nombre de valeurs différentes dans la colonne de relation R pour l'attribut a. Ces statistiquessont ensuite utilisées dans l'optimisation des requêtes, inchangées jusqu'à la prochaine commande decollecte de statistiques. Généralement, les statistiques sont sauvegardées dans la base de donnéescomme relations appelées catalogue ou dictionnaire de données [9].

En balayant entièrement la relation R, il est facile de compter le nombre de tuples T (R) et ausside découvrir le nombre de valeurs V (R ,a) différentes pour chaque attribut a. De plus, un SGBD peutcalculer un histogramme de distribution des valeurs d'un attribut donné. Les histogrammes les plus courants sont [14] :

- ✓ Largeur égale.
- ✓ Hauteur égale.
- ✓ Valeurs les plus fréquentes.

1.5.3.2- Statistiques de système

Le coût de l'évaluation des requêtes peut être mesuré en fonction d'un certain nombre deressourcessystème différentes. Les ressources ou paramètres les plus importants sont les suivants :

- ✓ Coût d'accès au stockage secondaire
- ✓ Coût de stockage sur disque.
- ✓ Coût de calcul.
- ✓ Coût d'utilisation de la mémoire.
- ✓ Coût de la communication

1.5.3.3- Obtention des paramètres et leurs relations

Les paramètres du modèle de coût, tels que le coût d'E/S et de CPU, doivent être calculés.Les méthodes empirique et dynamique peut obtenir ces paramètres [13] :

1-Méthode empirique

Cette approche traite le SGBD comme une boîte noire. Elle consiste à identifier d'abord un ensemble de caractéristiques de requête et de données qui déterminentpotentiellement les coûts des opérateurs. Puis, exécuter ces requêtes d'apprentissages sur lesystème. Ensuite, appliquer des modèles statistiques ou d'apprentissage automatique aux donnéescollectées, et à partir de ceux-ci déterminer les paramètres finaux du modèle de coût et la relationentre eux [15].

2-Méthode dynamique

La méthode précédente suppose que l'environnement d'exécutionest toujours stable. Cependant, Dans la plupart des cas, les facteurs d'environnement d'exécutionchangent fréquemment, tels que la charge et la vitesse du CPU, le débit d'E/S, la disponibilité demémoire, le schéma et les données de la BD, etc. Cette approche se base sur la méthodeempirique, et de plus, elle surveille le comportementd'exécution du SGBD et collecte et ajustedynamiquement les informations sur les coûts pour faire face au changement d'environnement[16].

1.5.3.3- Approches pour l'énumération des plans physiques

Maintenant, considérons l'utilisation des estimations de coûts dans la conversion d'un plan derequête logique en un plan de requête physique. L'approche de référence, appelée exhaustive, consisteà

considérer toutes les combinaisons de choix possible. Chaque plan physique est affecté à un coûtestimé, et celui avec le plus petit coût est sélectionné [13]. Cependant, il existe d'autres approches poursélectionner un plan physique. En termes d'exploration de l'espace des plans physiques possibles, ilexiste deux approches principales : (1) descendante et (2) ascendante. Dans la première approche,l'estimation se fait à partir de la racine de l'arbre du plan de requête logique vers le bas en choisissant le meilleur choix à chaque étape. Cette approche est adoptée par Volcano, Cascades, Tandem etMicrosoft SQLServer .Tandis que dans la deuxième approche, l'estimation se fait à partir desfeuilles vers la racine. Cette approche est utilisée dans System R, DB2, Oracle et Informix [17]. Nous décrirons ensuite les différentes techniques proposées dans la littérature pour l'énumération des plans physiques. Les différentes techniques proposées pour l'énumération des plans physiques [13] sont :

- ✓ Sélection heuristique.
- ✓ Séparation et évaluation (Branch-and-Bound).
- ✓ Escalade (Hill climbing).
- ✓ Programmation dynamique.
- ✓ System-R.

1.5.4- Exécution

L'exécution de la requête est la dernière étape du processus. Dans c'elle ci, toutes les opérationsd'E/S et de CPU indiquées dans le plan physique sont exécutées.

1.6. Conclusion :

Comme nous avons vu au cours de ce chapitre, l'objectif principal lors du traitement de requête est la minimisation du temps de réponse des requêtes et exécuter un plan optimal.

Le problème consiste à développer des techniques qui offrent le meilleur compromis entre la minimisation du temps de réponse aux requêtes et la minimisation de la consommation d'énergie du système.

Dans ce chapitre nous avons décrit l'ensemble des phases traditionnelles de cycle de vie de conception d'une base de données : l'analyse de besoins, la modélisation conceptuelle, logique, physique et le déploiement.

Nous avons identifié l'opportunité d'intégrer l'énergie dans la phase de conception physique, plus précisément, dans le choix d'une structure d'optimisation.

Deuxièmes, nous avons étudié la phase de traitement de requêtes dans un SGBD relationnel, qui inclue : la phase d'analyse, la transformation, la génération de plans et optimisation et enfin l'exécution.

Dans le prochain chapitre, nous allons présenter le concept d'énergie, ses propriétés et les travaux de minimisation d'énergie dans les systèmes informatiques.

CHAPITRE 2

L'ÉNERGIE DANS LES SYSTÈMES INFORMATIQUES

2.1- Introduction :

Pour continuer à être de plus en plus puissants, les ordinateurs ont besoin de plus en plus d'énergie. Cette consommation pouvait être ignorée jusque-là, elle atteint désormais des sommets, ce qui pose denombreux problèmes. Il est donc de plus en plus important de faire attention à la consommationénergétique de nos programmes et bases de données.

Dans ce chapitre on présente les concepts de base liés à l'énergie et quelques techniques d'optimisation de l'énergie.

2.2- L'énergie dans la technologie de l'information

Deux métriques sont couramment utilisées pour qualifier la performance énergétique de nos bases de données. La première est l'énergie, la seconde la puissance. Ces deux métriques représentent deux choses distinctes, mais liées.

- **Energie :**

Définition 1: l'énergie est une mesure de la capacité d'un système à modifier un état, à produire un travail entraînant un mouvement, un rayonnement électromagnétique ou de la chaleur. Dans le Système international d'unités (SI), l'énergie s'exprime en joule.

Définition 2: L'énergie est aujourd'hui définie comme la « capacité d'un corps ou d'un système à produire du travail mécanique ou son équivalent.

- **Puissance :**

Définition :la puissance est l'énergie fournie à un système par un autre par unité de temps et mesuré en Watts.

La réduction de l'énergie soit par :

- La minimisation du laps du temps.
- La minimisation la puissance consommée.

En général, la consommation de la puissance électrique d'un tel système est divisée en deux parties :

1. **Puissance de base :** La consommation de la puissance lorsque le système est inactif.
2. **Puissance active :**La consommation de la puissance lors de l'exécution d'une charge de travail.

Deux concepts de puissance électrique doivent être pris en considération lors de l'évaluation de l'utilisation d'énergie dans un système : la puissance moyenne représentant la puissance moyenne

consommée au cours d'exécution de la charge de requête, et le pic représentant la puissance maximale (peak power).
Lorsqu'on évoque des économies de l'énergie, on utilise les termes « efficacité » ou « efficience ».

De manière générale, l'Efficacité Énergétique (EE)ou efficience énergétique désigne l'état de fonctionnement d'un système pour lequel la consommation d'énergie est minimisée pour un service rendu.

Nous pouvons améliorer l'efficacité énergétique en réduisant la puissance, le temps ou les deux à la fois.

2.3- Approches d'EE dans les systèmes informatiques

Aujourd'hui, les enjeux de l'informatique éco-énergétique sont de plus en plus étudiés à tous les niveaux dans les infrastructures informatiques. Compte tenu d'une brève description des méthodes d'évaluation d'énergie dans les systèmes, nous présentons dans cette section un état de l'art des travaux d'EE des systèmes informatiques sur trois niveaux [18] :

(1) matériel,
(2) système d'exploitation et
(3) application.

2.3.1- Approches d'efficacité d'énergie au niveau matériel

Un ordinateur de calcul contient plusieurs composants. Pour économiser l'énergie, chaque composant peut être optimisé. Nous allons présenter les techniques d'optimisation proposées pour chaque composant :

1- **Pour réduire la consommation d'énergie du CPU**, L'idée est le contrôle dynamique de la vitesse d'horloge pour réduire la consommation d'énergie pour un travail particulier comme ils l'ont proposé les auteurs [19]. Celle-ci permet la réduction dela tension électrique qui suit le ralentissement de l'horloge. Mais cette technique engendre l'augmentation de temps d'exécution. Cela à encourager les chercheurs [20]de construit un modèle théorique qui va trouver le moyen le plus économe en énergie pour ordonnancer les tâches tout en respectant les délais d'exécution.
Une autre méthode essai de ralentir le CPU à chaque fois qu'il y a une seule tâche élu pour l'exécution, a été développée [21]. Deux algorithmes hors et en ligne sont considérés en respectant les délais, tout en réduisant la vitesse du cycle CPU autant que possible [16].Les informations de la date limite sont adoptées dans le système d'exploitation en temps réel avec la technique DVFS [22]. La norme ACPI (Advanced Configuration and Power Interface), disponible sur la plupart des systèmes d'exploitation. Le but de cette norme est de réduire la consommation d'énergie d'un ordinateur en mettant hors tension certains éléments : lecteurs CD-ROM, disques durs, écran, etc.
2- **La gestion de la mémoire** est aussi importante pour atteindre l'efficacité énergétique.Mais ni la gestion d'alimentation de mémoire ni encore la technique DVS sur le processeur, peuvent

économiser l'énergie de façon spectaculaire comme montré par les auteures de [23]. C'est avec la combinaison des deux techniques, 89% d'économie d'énergie, par rapport au cas de base, peut être achevé. Les techniques d'optimisation et les algorithmes proposés se sont concentrés sur les opportunités de basculer la mémoire entière ou une partie d'elle en mode faible consommation, soit pendant ou au moment de l'exécution des processus. D'autres technologies émergentes telles que la mémoire à changement de phase (PCM), transfert de spin et memristor ont également été proposées comme des alternatives de la mémoire principale économe en énergie [24].

3- **Les unités de traitement graphique (GPU)** sont également devenues une partie essentielle des centres de données hébergeant des applications scientifiques à cause de leur efficacité dans les fonctions à virgule flottante. De plus, les GPUs sont très efficaces sur le plan énergétique par rapport aux processeurs.

Dans les équipements de stockage pour atteindre l'EE Il existe deux techniques de base :

- ✓ rendre le matériel de stockage économe en énergie
- ✓ réduire la redondance des données.

Le principal objectif du matériel de stockage éco-énergétique est constitué par les disques SSDs (Solid-State Drive). Les disques SSD utilisent une mémoire Flash, c'est-à-dire une mémoire non volatile ayant des caractéristiques similaires à la mémoire morte effaçable électriquement et programmable (EEPROM). Les SSDs ont des propriétés proportionnelles à l'énergie, du fait que l'énergie utilisée est proportionnelle aux opérations d'E/S par seconde [25]. D'autre part, les disques durs (HDDs) consomment 85% d'énergie lorsqu'ils sont inactifs [26]. Les SSDs deviennent l'avenir du stockage primaire économe en énergie dans les systèmes.

4- **Les périphériques réseau**

Constituent une autre ressource énergétique dans les centres de données. La consommation d'énergie dans les périphériques réseau peut être gérée par des techniques de taux de liaison adaptatif (ALR) [27]. Les techniques d'ALR atteignent l'EE par :

- ✓ les débits de données réduits.
- ✓ les transitions d'état de puissance inférieure/inactive.

2.3.2- Approches d'EE au niveau système d'exploitation

Les composants du système consommant la majeure partie de l'énergie en trois catégories [27] :

- unités de calcul.
- unités de communication.
- unités de stockage.

Les techniques éco-énergétique auprès de trois phases d'une conception de système :

Premièrement c'est la conceptualisation et la modélisation ensuite la conception et la mise en œuvre dernièrement la gestion d'exécution.

Les systèmes d'exploitation (Operating system OS) ont une consommation d'énergie diverse et peuvent-être optimisés pour consommer moins d'énergie selon les chercheurs. Cette consommation des ordinateurs d'une version d'un système d'exploitation à l'autre (des versions de Windows et linux) est non négligeable selon les méthodes de mesure similaires.

Pour le système d'exploitation Linux les chercheurs ont développé un gestionnaire d'alimentation en temps réel appelé le gouverneur à la demande (LGD). Le gestionnaire surveille en permanence l'utilisation du processeur et définit une fréquence d'horloge qui correspond aux exigences de performances actuelles, ce qui minimise la consommation d'énergie [28].

2.3.3- Approches d'EE au niveau application

L'efficacité énergétique au niveau application est devenue très importantes dans un domaine de recherche parce que [29] :

1- les techniques d'optimisation de bas niveau (matériel et système d'exploitation) dépendent de l'estimation précise de la puissance des applications. Cependant, beaucoup d'informations et de statistiques nécessaires pour ces estimations ne sont disponibles que dans le niveau application.

2- Divers applications peuvent être exécutées de différentes manières pour accomplir la même tâche de calcul. Les applications éco-énergétique, qui adaptent leurs comportements en fonction des états liés à l'alimentation des systèmes sous-jacents, peuvent offrir des possibilités supplémentaires d'économie d'énergie.

SEEDS est un Framework pour aider les ingénieurs des logiciels à développer des applications éco-énergétiques de haut niveau. Une application récente des techniqueséco-énergétique inclut MapReduce [30].

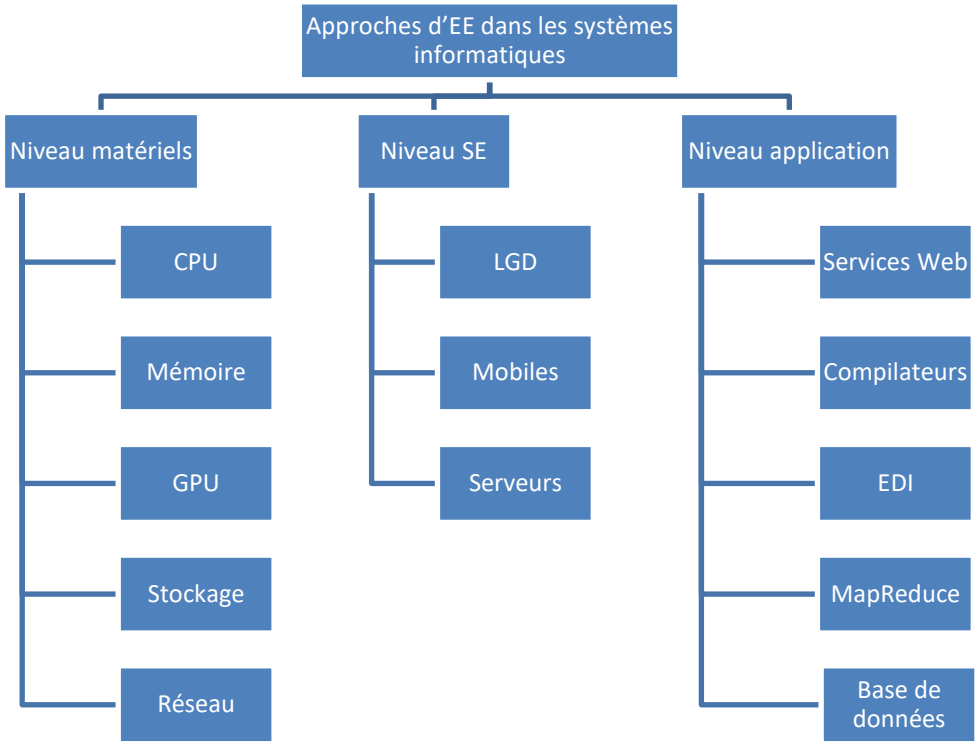

Figure 2.1 Classification des approches d'EE dans les systèmes informatiques.

La consommation d'énergie dans les bases de données commence à attirer l'attention de la communauté des chercheurs. L'objectif principal est de concevoir des SGBDs avec l'optimisation de la consommation d'énergie.

2.4- Approches d'EE dans les BDD

La gestion de l'énergie des systèmes de bases de données qu'est un sujet très important dans nos jours. On distingue principalement deux approches principales :

- Approches Orientées Matériels (AOM).

-Approches Orientées Logicielles (AOL).

Nous allons détaillés les travaux de chaque approche comme suit :

2.4.1- Approches Orientées Matériels

Les approches matérielles utilisent la désactivation de composants électroniques pendant les périodes d'inactivité et la modification dynamique de la performance des composants matériels pour une meilleure efficacité énergétique qui peut être illustrer par la technique DVFS (Dynamic Voltage and FrequencyScaling) offerte par les CPU [31]. Cependant, les techniques appliquées au niveau du matériel peuvent être divisées en deux catégories :

- Dispositif de traitement
- La gestion du stockage.

➢ Dispositif de traitement

Cette catégorie utilise des nouvelles approches pour minimiser la consommation d'énergie des appareils informatiques. *Lang et al* ont proposé le mécanisme PVFC (Processor Voltage/Frequency Control) pour équilibrer la consommation d'énergie et la performance, par l'exécution d'une tache à une basse tension et fréquence de processeur.

La technique proposée ajuste dynamiquement le niveau DVFS du processeur en fonction des performances du SGBD ou le plan de requête choisi pour l'exécution.

Afin d'économiser l'énergie, ils gèrent le niveau DVFS en fonction du débit de travail et des caractéristiques de la charge de requêtes (en basant sur le nombre d'E/S ou de CPU) [32], cette technique est utiliséepar Xu et al avec un mécanisme de contrôle de rétroaction pour les charges de requêtes d'une base de données.

Oracle et Intel tente de réduire les coûts d'exploitation et de répondre efficacement aux objectifs d'informatique durable (green computing) a donné lieu à une nouvelle version de Oracle ExadataDatabase Machine [33]. La solution proposée met dynamiquement le processeur Intel Xeon et la mémoire dans l'état de puissance disponible le plus bas convenable pour répondre aux exigences de la charge de requêtes.

L'utilisation de disques Smart Solid State (SSSDs) pour traiter les données peuvent-être plus avantageux à la fois en termes de performance et de consommation d'énergie. Les SSSDs se sont des dispositifs de stockage flash qui intègrent une mémoire et un processeur à l'intérieur du dispositif de SSD, et pourrait être utilisé pour exécuter des programmes généraux définis par l'utilisateur. Les premiers résultats de l'exécution d'un sous-ensemble de requêtes SQL (simples requêtes de sélection et d'agrégation) sur un SSSD fournissent jusqu'à 3x de réduction de la consommation d'énergie.

➢ Gestion du stockage

Les approches de gestion de stockage tentent de trouver un bon niveau de consolidation de la charge, l'amélioration des techniques de mise en cache et de pré-chargement et l'optimisation des modes d'alimentation dans les matrices de disques afin de minimiser la dissipation d'énergie [34].

Un modèle de gestion de puissance dynamique qui prend des décisions en temps réel sur le passage des disques à des modes de faible puissance. Le modèle est intégré dans le SGBD et utilisé pour obtenir le compromis optimal entre la consommation d'énergie et le temps de réponse des requêtes. Le rapport montre 60%des économies d'énergie.

L'utilisation de disques SSD comme un périphérique de stockage des bases de données pour améliorer l'EE est une autre direction qui a été étudiée par des travaux récents. Dans [35], les auteurs ont étudié le comportement de la performance et la consommation d'énergie des SSD pour des schémas d'accès typiques pour les applications de base de données à forte intensité d'E/S. Ils montrent, en outre, que la technologie SSD offre un nouveau niveau de performance et d'EE, et suggèrent l'exploitation de

leurs nouvelles caractéristiques par les serveurs de base de données. De même, le travail de [36] analyse et évalue les performances de traitement des données et l'EE des SSD.

La mémoire est un autre consommateur d'énergie important dans les systèmes informatiques, en particulier avec l'utilisation croissante des bases de données orientée colonnes où le stockage des données se fait directement dans la mémoire.

Comme pour les composants précédents, la mémoire principale a également des différents états de puissance et de la fréquence à modifier en fonction de son utilisation. Sur cette base, les auteurs ont utilisé la fréquence mise à l'échelle et des modes de mise hors tension des DRAM pour améliorer la consommation d'énergie sans sacrifier les performances. Le travail de [37] propose une architecture de mémoire hybride comprenant à la fois DynamicRandom Access Memory (DRAM) et de la mémoire non volatile (NVM) pour réduire la consommation d'énergie de base de données, grâce à une politique de gestion des données au niveau de l'application qui décide de placer des données sur la DRAM ou la NVM.

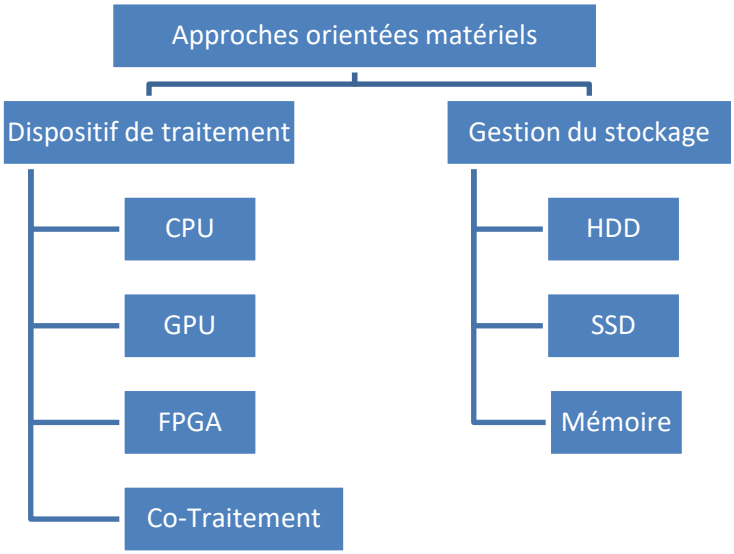

Figure 2.2 Classification des approches d'EE orientées matériels.

2.4.2- Approches Orientées Logicielles

Les approches logicielles sont l'autre élément dans les mécanismes d'optimisation de l'EE. A ce niveau les chercheurs recommandent de revoir les techniques d'optimisation des requêtes, les algorithmes d'ordonnancement, la conception physique et les techniques de mise à jour de base de données en tenant compte de l'énergie. La plupart des études concernent deux aspects principaux :

- La définition des modèles de coûts pour prédire l'énergie.
- La proposition des techniques basées sur les modèles de coûts pour réduire l'énergie.

Dans ces approches, un optimiseur basé sur les coûts estime le coût de chaque plan d'exécution possible en appliquant des formules heuristiques utilisant un ensemblede statistiques relatives à la base de données (taille des tables, des indexes, longueur de tuples, les facteurs de sélectivité de jointure et

prédicats de sélection, la taille des résultats intermédiaires, etc.) et au matériel (taille du tampon mémoire, la taille de page, etc.) [38].

2.4.2.1- Définition des modèles de coûts

Une extension du modèle de coût de PostgreSQL pour prédire la consommation d'énergie d'une seule requête exécutée d'une façon isolée (pas de parallélisme inter-requête).
Un profil de puissance statique pour chaque opération de base de données basique est défini dans la phase du traitement des requêtes. Le coût de l'énergie d'un plan peut être calculé à partir des opérations de base de données SQL, comme le coût de la puissance du processeur pour accéder à un tuple, coût du disque pour la lecture/écriture d'une page et par l'intermédiaire de différentes méthodes d'accès et d'opérations de jointure, en utilisant une technique de régression linéaire simple.

Dans le même sens, le travail de [39] propose un framework pour un traitement éco-énergétique des requêtes dans les bases de données. Il étend les plans de requêtes produites par l'optimiseur de requête traditionnelle avec une prédiction de la consommation d'énergie pour certains opérateurs de bases de données spécifiques comme la sélection, projection et la jointure en utilisant la technique de régression linéaire.

2.4.2.2- Techniques basées sur des modèles de coûts

Les modèles de coûts proposés pour prédire l'énergie sont ensuite utilisés dans diversestechniques pour optimiser la consommation d'énergie de base de données. Les auteurs de [40] ont proposé un mécanisme pour améliorer l'EE des requêtes en introduisant des retards explicites, en utilisant l'agrégation des requêtes pour tirer parti des composants de requêtes communs dans la charge. Le retard explicite permet de grouper les requêtes dans une file d'attente en mémoire tampon, ce qui conduit alors à une évaluation plus efficace de l'ensemble du lot par des techniques d'optimisation multi-requêtes.

Le travail de [39] a montré que le traitement d'une requête aussi vite que possible ne s'avère pas toujours être la façon la plus économe en énergie dans un SGBD. Sur la base de leurs modèles de coût proposé pour prédire l'énergie, ils choisissent des plans de requêtes qui réduisent la consommation d'énergie et répondent aux objectifs de performance traditionnels. Ce choix est fait manuellement. Les résultats montrent que les économies d'énergie peuvent atteindre jusqu'à 23% pour une augmentation de 5% du temps de réponse en ce qui concerne les requêtes équijointure.

Le travail de [41] mène une étude en profondeur dans l'analyse de l'EE d'un serveur de base de données sous différents paramètres de configuration, en faisant valoir que les avantages obtenus dans les systèmes de base de données sont de petite taille. Cependant, dans leur environnement, ils donnent de l'importance à la puissance du processeur plutôt que la puissance active globale du système. Néanmoins, dans la pratique, il existe des requêtes qui sont intensive en E/S. Considérant que le CPU à la place de l'ensemble des composants peut réduire considérablement la marge d'amélioration d'EE [42].

2.5. Approches et algorithmes dans les systèmes récupérateurs d'énergie :

On identifie dans la littérature principalement deux approches pour résoudre les problèmes liés aux systèmes temps réel récupérateur d'énergie. La première utilise la technique de gestion dynamique de la consommation appelée DPM (mémoire non volatile, en anglais). Cette approche consiste à gérer l'énergie disponible sans avoir à baisser la consommation [43].Cela passe par une sélection judicieuse des moments de consommation d'énergie et ceux de rechargement de batterie ou des périodes d'activité et celles d'inactivité du processeur afin de ne jamais tomber à court d'énergie et faire au mieux pour respecter toutes les échéances. La deuxième approche vise à minimiser la consommation d'énergie avec l'utilisation de la technique d'adaptation dynamique de la tension d'alimentation et de la fréquence du processeur(notée DVFS) au besoin courant de l'application en termes deperformances [44].

Nous présentons, dans ce qui suit, les principaux algorithmes, suivant l'approche adoptée :

2.5.1. Les algorithmes basés sur la technique DPM

Dans cette sous-section nous présentons les algorithmes de la technique DPM.

2.5.1.1. Frame-BasedAlgorithm (FBA) :

FBA est le premier algorithme proposé pour les systèmes récupérateurs d'énergieambiante par Allavena et Mossé dans [43]. Cet algorithme est spécifiqueau modèle de tâches périodiques (dites basiques) (Frame Based System, en anglais)de périodes identiques appelées fenêtre et d'échéance commune (délai critique commun).

L'idée est de diviser les tâches de la configuration en deux groupes : les tâches ditesconsommatrices qui ont un taux de consommation supérieur au taux de rechargement dela batterie, et les tâches dites génératrices qui ont un taux de consommation inférieurou égal à ce taux de rechargement. L'algorithme ordonnance alors en continue les tâchesconsommatrice pour vider la batterie puis les génératrices pour la recharger. Afin d'assurerl'équilibre entre la production d'énergie et sa consommation, l'ordonnanceur doitinsérer un intervalle de temps d'inactivité (temps creux) du processeur n'engendrant niconsommation ni production d'énergie liée à l'exécution d'une tâche. La même séquenced'ordonnancement est répétée à chaque période.

2.5.1.2. LazySchedulingAlgorithm (LSA) :

LSA est un algorithme d'ordonnancement en ligne basé sur EDF(Earliest Deadline First). Il a été proposé parMoser et al. de l'Institut Polytechnique de Zurich dans [45] où la preuve deson optimalité ainsi que la condition d'ordonnancement ont été présentés dans un premiertemps. Par la suite, des détails de l'algorithme et des études de cas de simulation ontété apportés dans [46] montrant les bienfaits de cettenouvelle approche d'ordonnancement. Cet algorithme permet d'ordonnancer une configurationde tâches périodiques ou apériodiques critiques (c'est-à-dire munies d'échéancesstrictes). LSA est dirigé par l'énergie (energy-drivenalgorithm, en anglais) contrairementaux algorithmes classiques lesquels sont

dirigés par le temps (time-drivenalgorithm, enanglais). Cela veut dire que la décision d'ordonnancement sous LSA est prise uniquementsur la base de l'énergie disponible et sur la connaissance du profil énergétique de la source.

Ceci vient du fait que les tâches consomment l'énergie avec le même taux et que le tempsd'exécution des tâches dépend du taux de consommation appliqué.

2.5.1. 3. EDF with energy guaranty (EDeg) :

L'algorithmeEDeg (Earliest Deadline with energy guarantee) estproposédans[47]. Intuitivement, cet algorithme vise à ordonnancerles tâches selon EDF. Cependant, avant d'autoriser une instance à s'exécuter, EDegutilise la notion du laxité énergétique (slackenergy , en anglais) pour prédire d'éventuelsfuturs dépassements d'échéances dus à une insuffisance d'énergie. Cette notion dela laxité énergétique est l'équivalent de la laxité temporelle (slack time, en anglais) utiliséepar EDL [48] (où la laxité temporelle représente lalongueur de l'intervalle de temps pendant lequel le processeur peut rester inactif tout engarantissant le respect des contraintes temporelles). La laxité énergétique représente laquantité maximale d'énergie qui peut être consommée sans tomber dans une situationd'épuisement énergétique ce que remis en question l'exécution des tâches à venir. Si undépassement d'échéance futur, dû au manque d'énergie, est détecté, les travaux actuelssont retardés aussi longtemps que possible en consommant le temps creux du processeurdisponible pourreconstituer un maximum d'énergie. En outre, lorsque Emax est atteintau cours d'une périoded'inactivité, l'algorithme EDeg reprend les exécutions afin d'éviterle gaspillage d'énergie.

2.5.2. Les algorithmes basés sur la technique DVFS

Dans cette sous-section nous présentons les algorithmes de la technique DVFS.

2.5.2.1. Energy-aware DVFS Algorithm (EA-DVFS)

L'algorithme EA-DVFS.(Energy-aware DVFS Algorithm) a été proposé par Liu etal. Dans [49] pour améliorer les performances de LSA. Cet algorithme modifiéle comportement du processeur en fonction de la quantité de l'énergie stockée et de cellequi sera récupérée dans le futur (prédiction). Les auteurs considèrent un système tempsréel équipé d'un processeur fonctionnant avec la technique DVFS. Ce processeur possèdeN fréquences discrètes :$f_{min} = f_1 < f_2 < ... < f_N = f_{max}$. Chaque fréquence fn correspondà une puissance de consommation P_n. Ils définissent le facteur de vitesse Sn, associé àchaque fréquence fn, où Sn égale à la fréquence normalisée f_n par rapport à la fréquencemaximale f_{max},donc $Sn = f_n/f_{max}$.

2.5.2.2. Harvesting-aware DVFS algorithm (HA-DVFS)

L'algorithme HA-DVFS (Harvesting-awareDynamic Voltage and FrequencySelectionAlgorithm) proposé dans [44] calcule approximativement la date dedébut d'exécution (start time, en anglais) notée s_t et la date de fin d'exécution (finishingtime, en anglais) notée f_t de chaque tâche à partir des contraintes

temporelles comme lesWCET (Estimation of Wind Chill Equivalent Temperatures) et énergétiques dans le but d'ordonnancer toutes les tâches avec une vitesse minimalepossible du processeur. Cependant, étant donné la variabilité des durées d'exécutiondes tâches rend impossible la connaissance à priori la date de fin d'exécution exacte.Enplus, HA-DVFS comme EA-DVFS se base sur l'utilisation de WCETs.

2.6. Prise en compte de l'énergie dans la phase d'exploitation des bases de données volumineuses [50] :

Il a proposé un modèle de coût basé sur les pipelines pour estimer la consommationd'énergie lors de l'exécution d'une requête SQL et un ensemble de requêtes exécutées simultanément.

Il a aussi indiqué comment la modélisation de pipeline pourrait être un indicateur robuste pourla prédiction d'énergie. Le modèle de coût est construit sur la base des résultats empiriques obtenus àpartir d'une charge de requêtes d'apprentissage qui a été soigneusement créée. Les paramètres matérielssontobtenus par un algorithme de régression polynomiale multiple. En outre, il a effectué destestssur son modèle avec une BDD réelle en exécutant un ensemble de requêtes, et a comparé leurscoûts énergétiques avec ceux prédits par le modèle. Leurs résultats montrent que le modèle, qui utiliseuniquement des coûts de l'optimiseur en entrées, peut prédire l'énergie avec une petite erreur.

2 .7 . Conclusion :

Les bases de données représentent aujourd'hui l'un des principaux défis en matière d'EE. La réduction d'énergie dans les BD est d'une grande importance économique, car ils sont utilisés par la plupart des environnements informatiques des entreprises. Dans un centre de données, la majorité des ressources informatiques sont dédiées aux serveurs de bases de données, ce qui en fait le plus grand consommateur d'énergie parmi toutes les applications logicielles déployées.

Face à cette situation, les industriels et les chercheurs de la communauté scientifique ont pris des initiatives accompagnées de solutions réelles couvrant plusieurs actions, à savoir :

• la conception de matériel avec une consommation énergétique réduite. • la restructuration des logiciels pour minimiser la consommation.

• l'exploitation de divers états d'alimentation disponibles dans les nouveaux matériaux.

• une bonne conception, utilisation et contrôle de l'infrastructure des centres de données. Et pour cela la technique DVFS est l'un des méthodes appliquées pour réduire l'énergie consommé par des ordinateurs lors d'exécution des requêtes SQL et des accès aux bases de données.

CHAPITRE 3

ANALYSE ET CONCEPTION

3.1. Introduction :

Après avoir introduit, dans les chapitres précédents les déférentes concepts nécessaires à l'aboutissement de notre travail, nous passons maintenant à la partie principale qui est mise en place d'un système qui compte l'énergie et le te temps de réponse d'un CPU en appliquant la méthode DVFS.

Notre travail consiste à concevoir et mettre une application qui permettre d'économiser l'énergie du CPU. Nous présentant de langage UML.

3.2. L'approche de conception :

Aujourd'hui, en programmation, il existe deux principaux modèles de représentation du monde :

- **Le modèle fonctionnel**
- **Le modèle objet**

Dans **une approche fonctionnelle**, les programmes sont composés d'une série de fonctions, qui assurent ensemble certains services. Il s'agit d'une approche logique, cohérente et intuitive de la programmation.

Cette approche a un avantage certain appelé la factorisation des comportements (c'est à dire que pour créer une fonction d'une application, rien ne vous empêche d'utiliser un autre ensemble de fonctions (qui sont donc déjà écrites)).

Mais, l'approche fonctionnelle a aussi ses défauts, comme par exemple une maintenance complexe en cas d'évolution d'une application (une simple mise à jour de l'application à un point donné peut impacter en cascade sur d'autres fonctions de l'application). L'application sera alors retouchée dans sa globalité.

L'approche fonctionnelle n'est pas adaptée au développement d'applications qui évoluent sans cesse et dont la complexité croit continuellement (plusieurs dizaines de milliers de lignes de code).

L'approche objet possède donc plusieurs avantages :

- la modélisation des objets de l'application (consiste à modéliser informatiquement un ensemble d'éléments d'une partie du monde réel en un ensemble d'entités informatiques. Ces entités informatiques sont appelées objet)

- la modularité (la programmation modulaire permet la réutilisation du code, via l'écriture de librairies)
- la réutilisabilité
- l'extensibilité (pour une meilleure productivité des développeurs et une plus grande qualité des applications).

L'approche objet a été donc inventée pour faciliter l'évolution d'applications complexes. Elle apporte l'indépendance entre les programmes, les données et les procédures parce que les programmes peuvent partager les mêmes objets sans avoir à se connaître comme avec le mécanisme d'import/export.

L'approche objet a introduit indépendamment de tout langage de programmation à l'objet trois concepts de base : objet, classe et héritage entre classes. Les concepts objet et classe sont interdépendants, c'est-à-dire qu'un objet est une instance d'une classe et la classe décrit la structure et le comportement communs d'objets (ses instances).

De nombreux langages orientés objet ont été mis au point, tel que: C++, Object Pascal, Java, etc.

Nous avons choisi pour la modélisation de notre système la langage UML, que nous allons introduire ci-dessous.

3.3. **Pourquoi utiliser l'UML dans le cadre d'un projet informatique ?**

De nos jours, les outils de modélisation de processus métier (ex BOUML) s'étoffent chaque année et les suites logicielles sont de plus en plus nombreuses. L'usage et les fonctionnalités d'UML diffèrent d'un périmètre à un autre, selon les besoins des clients et des fournisseurs d'applications.

Dans le cadre d'un projet informatique pour le SI, le recours à la modélisation UML procure de nombreux avantages qui agissent sur :

- La modularité
- L'abstraction
- La dissimulation
- La structuration cohérente des fonctionnalités et des données

Il permet aussi dans un premier temps de bien définir les besoins clients, et ainsi d'éviter des surcoûts liés à la livraison d'un logicielle qui ne satisfait pas le client (Selon une étude du K Molokken et

M Jorgensen en 2003, pour 53% des logiciels crées, le taux de délai de livraison non respecté est de 120%, on avait 90 % de budgets non tenus, et 60% de non disponibilité de certaines fonctionnalités) [51].

De plus, la modélisation UML permet de vulgariser les aspects liés à la conception et à l'architecture, propres au logiciel, au client. Aussi, elle apporte une compréhension rapide du programme à d'autres développeurs externes en cas de reprise du logiciel et facilite sa maintenance.

Toutefois, il existe aussi des inconvénients quant à l'utilisation d'UML. Pour la plupart, essentiellement liés au dépassement du délai de livraison du logiciel.

Pour pallier à ce problème, le recours à un cycle de projet en spirale est recommandé car il apporte plus d'agilité et une meilleure gestion des risques.

L'utilisation d'UML nécessite par ailleurs, une formation préalable pour connaître ses normes standards, cela peut être un inconvénient pour le client qui n'a pas de compétences dans ce domaine.

Notons aussi, qu'il peut y avoir une mauvaise correspondance entre l'UML et le projet finalisé (Schéma d'élaboration différent au niveau de la conception : l'analyste et le développeur étant deux personnes distinctes).

UML définit 9 types de diagrammes dans deux catégories de vues, les vues statiques et les vues dynamiques.

Vues statiques :

- **Les diagrammes de cas d'utilisation** décrivent le comportement et les fonctions d'un système du point de vue de l'utilisateur
- **Les diagrammes de classes** décrivent la structure statique, les types et les relations des ensembles d'objets
- **Les diagrammes d'objets** décrivent les objets d'un système et leurs relations
- **Les diagrammes de composants** décrivent les composants physiques et l'architecture interne d'un logiciel
- **Les diagrammes de déploiement** décrivent la répartition des programmes exécutables sur les différents matériels

Vues dynamiques :

- **Les diagrammes de collaboration** décrivent les messages entre objets (liens et interactions)
- **Les diagrammes d'états-transitions** décrivent les différents états d'un objet
- **Les diagrammes d'activités** décrivent les comportements d'une opération (en termes d'actions)

- **Les diagrammes de séquence** décrivent de manière temporelle les interactions entre objets et acteur

a- Modèle fonctionnel :

Le modelé fonctionnel décrit les calculs effectués par u système (un calcul est une transformation de donnée) sans tenir compte du séquencement des décisions ni des structures d'objets. Il précise la signification d'opération dans le modèle objet et la signification d'action dans le modèle dynamique.

Le modèle fonctionnel correspond aux **diagrammes de cas d'utilisation** en UML qui permettent de décrire l'interaction entre les acteurs et le système.

a-1. La spécification des besoins :

a.1 .1. Identification des acteurs et des scénarios :

Les acteurs Ce sont les utilisateurs directs du système. Ils doivent avoir une bonne connaissance des fonctionnalités du système. Ce sont des entités externes au système qui interagissent ou dialoguent avec lui. Ils sont identifiés par des rôles joués par des personnes ou d'autres systèmes logiciels. On distingue :

− **Des acteurs primaires (ou principaux**) : utilisateurs du système pour l'accomplissement de leurs buts.

− **Des acteurs secondaires :** autres participants qui peuvent fournir ou recevoir de l'information, ou s'occuper de la supervision ou de l'entretien du système. Pour trouver les acteurs d'un système, il faut identifier quels sont les différents rôles qui vont devoir jouer ses utilisateurs. Il faut vérifier que les acteurs communiquent bien directement avec le système par émission et réception de messages.

- **Identification des acteurs de notre système** :

Le principal acteur et le seul que nous avons pu identifier c'est : ***l'utilisateur***.

Les principales tâches effectuées par l'utilisateur peuvent être résumées dans le tableau 3.1 suivant :

Tableau 3.1 : récapitulatif des différentes taches des acteurs

Acteurs	tâches
L'utilisateur	b. L'accès à l'interface connexion à la base de données. c. Choix de la taille BDD. d. Etablir la connexion à la BDD e. L'accès à l'interface de l'exécution des requêtes. f. Le choix de la requête. g. Le choix de la fréquence du CPU. h. L'exécution de la requête. i. Le Traçage du graphe.

a.2. **Diagramme des cas d'utilisation :**

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système.

- **Rôle de diagramme d'utilisation :**

- Donne une vue du système dans son environnement extérieur.
- Définit la relation entre l'utilisateur et les éléments que le système met en œuvre.
- Est la base du modèle UML.
- Un acteur est l'archétype de l'utilisateur (personne, processus externe, ...) qui interagit avec le système.

L'acteur principal :

- Directement concerné par le cas d'utilisation décrit.
- Sollicite le système pour obtenir un résultat perceptible.

Un acteur secondaire :

- Est sollicité pour des informations complémentaires.
- nécessaires au déroulement du cas d'utilisation décrit.

Un cas d'utilisation modélise le service rendu par le système sans en imposer le mode de réalisation.

Une relation d'association est un lien de communication entre un acteur et un cas d'utilisation.

La relation d'inclusion spécifie qu'un cas d'utilisation est nécessairement une partie d'un autre cas d'utilisation.

La relation d'extension spécifie qu'un cas d'utilisation est éventuellement une partie d'un autre cas d'utilisation.

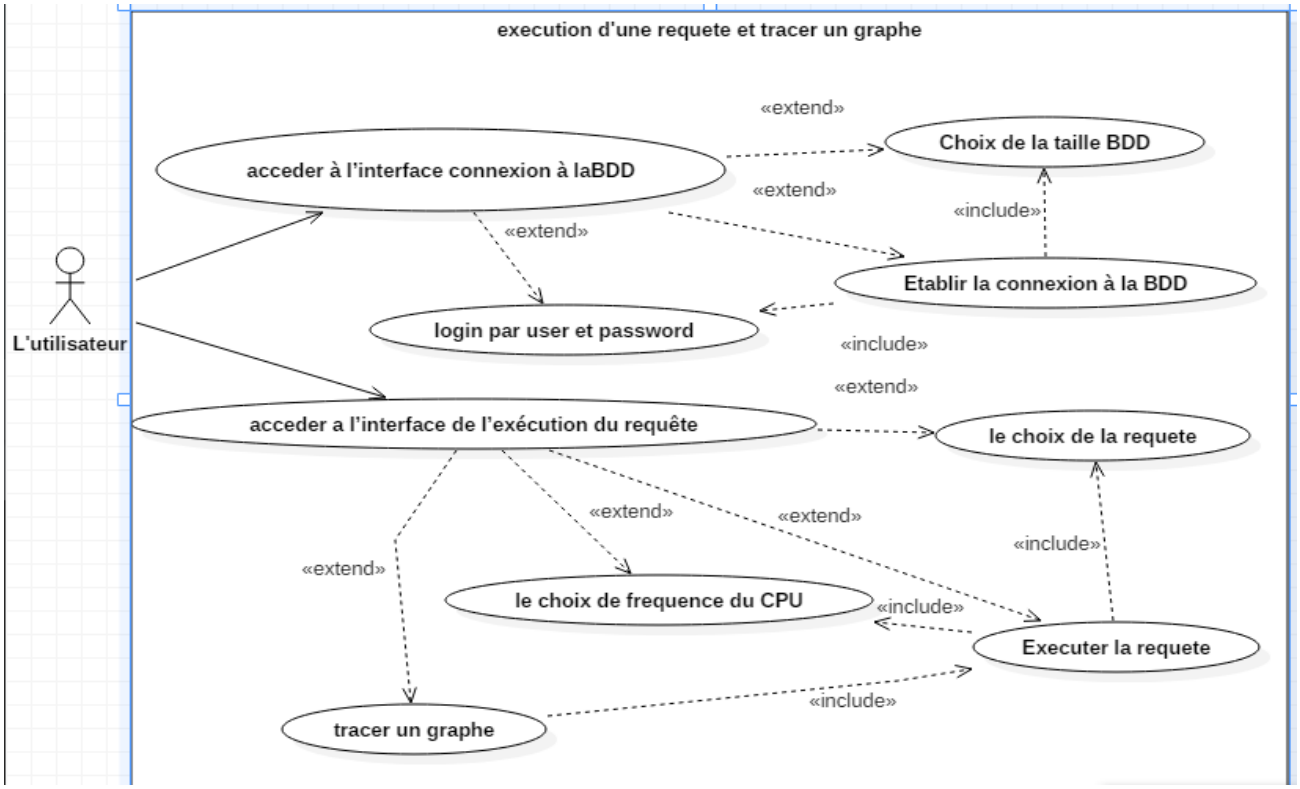

Figure 3.1 Diagramme de cas d'utilisations.

b- Le modèle Objet :

Le modèle des classes peut être décrit par les diagrammes de classes et diagrammes d'objet.

b.1. Le diagramme de classes :

Point central de la modalisation du système pour exprimer sa structure statique. Représentation d'un ensemble de classes, d'interfaces et de paquetages ainsi que leurs relations. Le diagramme de classes que nous avons élaboré est donné dans la figure 3.2

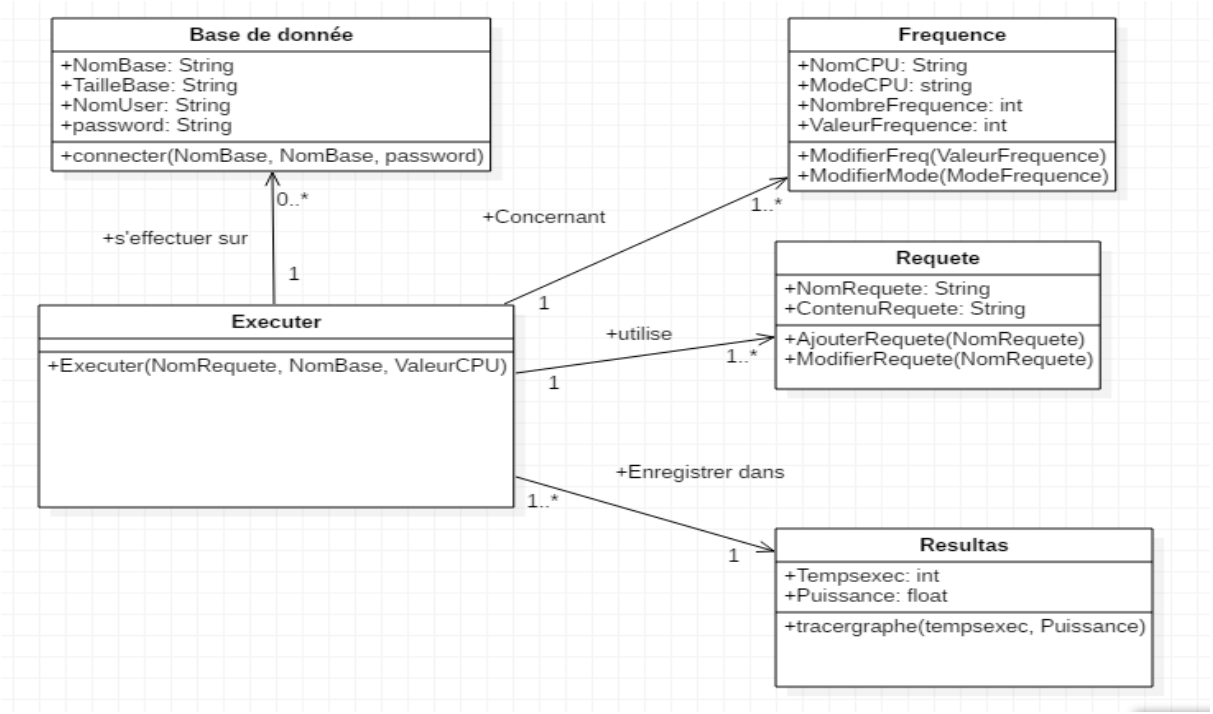

Figure 3.2 Le diagramme de classes

c- Le modèle dynamique :

La modélisation dynamique implique des étapes séquentielles, voire concurrentes dans un processus. Elle permet de modéliser le flot d'un objet lorsqu'il passe d'un état à un autre.

Ce modèle est décrit par les diagrammes de séquence et le diagramme d'activité d'UML.

L'activité analyse commence par l'examen des cas d'utilisation et de leurs scenarios ainsi que les besoins fonctionnels du système. Cette activité aboutit à un modèle constitué d'un ensemble de diagramme de séquence. Les classes d'objets peuvent être reparties en trois catégories :

- **Les objets interfaces** : ils représentent l'interface entre les différentes acteurs et le système tels que les écrans de saisies, les formulaires….etc.
- **Les objets entités** : ce sont les objets décrits dans le cas d'utilisation. Ils s'obtiennent généralement en isolant les noms des uses case.
- **Les objets contrôles** : ils représentent les processus, les activités du système. Ils s'obtiennent généralement en isolent les verbes de uses case.

L'utilisation de ces différentes classes d'objets doit obéir aux règles suivantes :

- Les acteurs n'interagissant qu'avec les objets interface.
- Les objets entités n'interagissant qu'avec les objets contrôle.
- Les objets contrôles interagissant avec tous les autres objets.

c.1. L'analyse des cas d'utilisation : Nous allons seulement présenter l'analyse de quelques cas d'utilisation :

❖ **Le cas d'utilisation « choix de la fréquence du CPU » :**

- **Les objets interface :**
    - Interface exécution de la requête.
- **L'objet contrôle :**
    - Modifier la fréquence du CPU.
- **L'objet entité :**
    - Système(CPU).

❖ **Le cas d'utilisation « choix de la requête » :**

- **Les objets interface :**
    - Interface exécution de la requête.
- **L'objet contrôle :**
    - Choix la requête.
- **L'objet entité :**
    - requête.

❖ **Le cas d'utilisation « exécutions de la requête » :**

- **Les objets interface :**
    - Interface exécution de la requête.
- **L'objet contrôle :**
    - Exécution de la requête.

- **L'objet entité :**
    - La Base de données.

C.2. Les diagrammes de séquences :

Les diagrammes de séquences permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs :

- Les objets au cœur d'un système interagissent en s'échangent des messages.
- Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

o Messages

Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie :

- Ils sont représentés par des flèches
- Ils sont présentés du haut vers le bas le long des lignes de vie, dans un ordre chronologique
- Un message définit une communication particulière entre des lignes de vie (objets ou acteurs).

- Plusieurs types de messages existent, dont les plus courants :
- l’envoi d’un signal ;
- l’invocation d’une opération (appel de méthode) ;
- la création ou la destruction d’un objet.

La réception des messages provoque une période d’activité (rectangle vertical sur la ligne de vie) marquant le traitement du message (spécification d’exécution dans le cas d’un appel de méthode).

Messages synchrones et asynchrones

- Un message synchrone bloque l’expéditeur jusqu’à la réponse du destinataire. Le flot de contrôle passe de l’émetteur au récepteur.
- Si un objet A envoie un message synchrone à un objet B, A reste bloqué tant que B n’a pas terminé.
- On peut associer aux messages d’appel de méthode un message de retour (en pointillés) marquant la reprise du contrôle par l’objet émetteur du message synchrone.
- Un message asynchrone n’est pas bloquant pour l’expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.

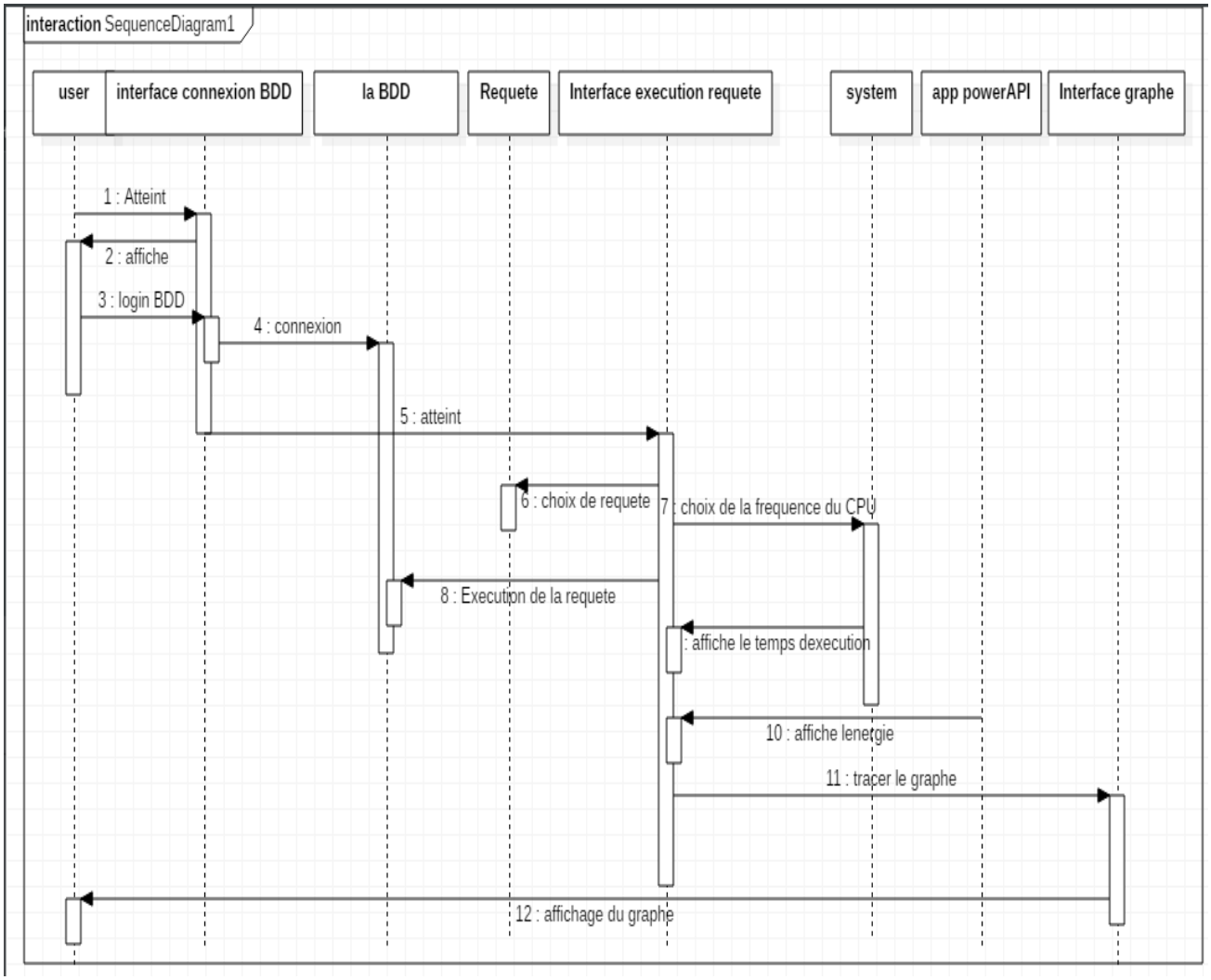

Figure 3.3 Diagramme de séquence

C.3. Diagrammes d'activités :

Les diagrammes d'activités permettent de modéliser un comportement tout en mettant l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle (séquence, choix, itération, parallélisme) et de flots de données entre activités alors que les diagrammes d'interaction modélisent le flot de contrôle entre objets.

Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation. Les diagrammes d'activités sont relativement proches des diagrammes d'états-transitions dans leur présentation, mais leur interprétation est différente.

Ils permettent de spécifier des traitements a priori séquentiels et offrent une vision très proche de celle des langages de programmation impératifs comme C++ ou Java. Les diagrammes d'activités sont très utilisés dans la phase de conception. Ils sont particulièrement adaptés à la description des cas d'utilisation. Plus précisément, ils viennent illustrer et consolider la description textuelle des cas d'utilisation.

De plus, leur représentation sous forme d'organigrammes les rend facilement intelligibles et beaucoup plus accessibles que les diagrammes d'états-transitions.

Les diagrammes d'activités sont également utiles dans la phase de réalisation car ils permettent une description si précise des traitements qu'elle autorise la génération automatique du code.

Un diagramme d'activités comprend :

- Les activités effectuées par le système et par les acteurs,
- L'ordre dans lequel ces activités sont effectuées et
- Les dépendances éventuelles entre les activités.

Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles Le passage d'une activité vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).

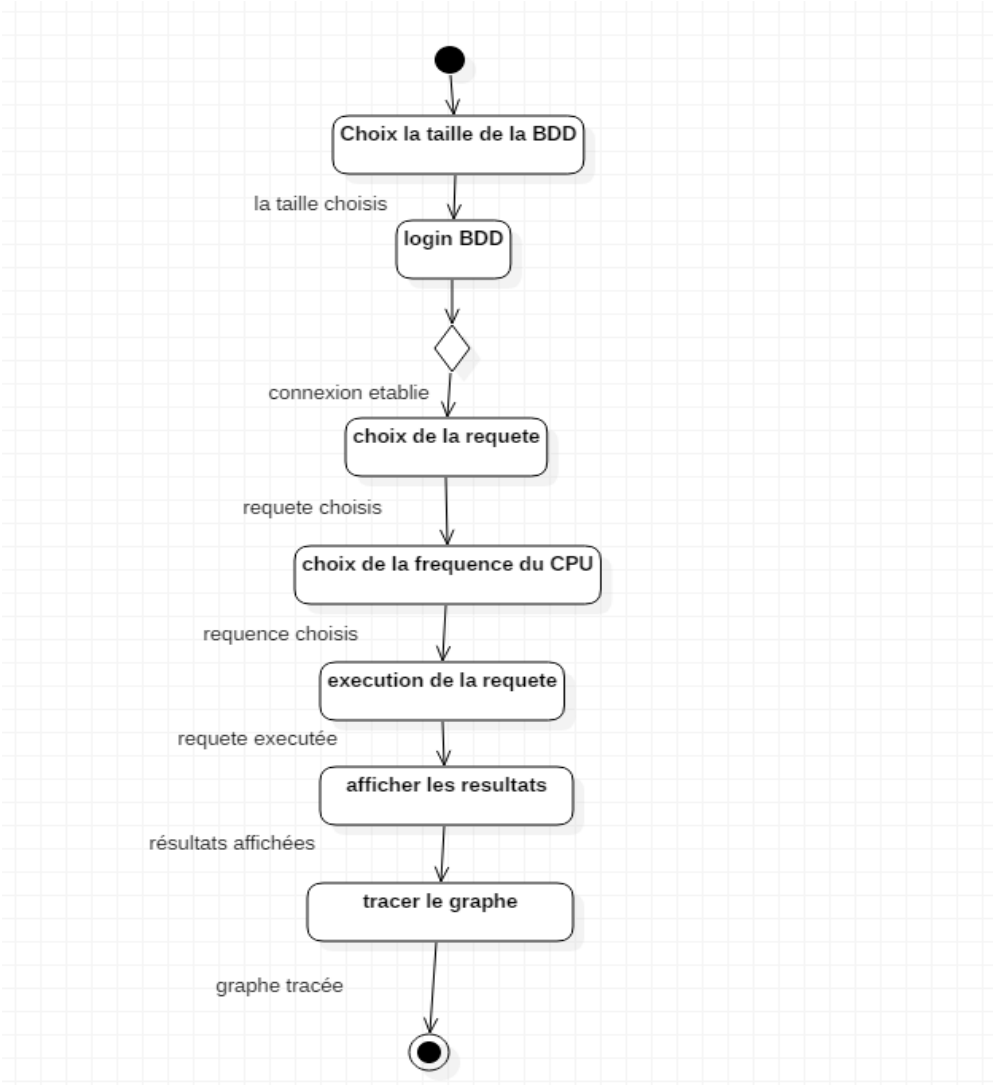

Figure 3.4 Diagramme d'activités

3.4. Conclusion :

Dans ce chapitre, nous avons présenté les caractéristiques de notre système, notamment le changement des fréquences du CPU ainsi que la sélections des requêtes est l'accès a les bases de données benchmarks de taille 1G et 10G.

Dans le chapitre suivant, nous allons compléter ces caractéristiques fonctionnelles avec les caractéristiques techniques relatives à l'outil de développement choisi.

CHAPITRE 4

RÉALISATION

4.1. Introduction

Dans ce chapitre nous allons présenter la structure générale, la modélisation et l'implémentation de notre système pour réaliser une application qui minimise la consommation d'énergie d'un CPU lors d'exécution d'une requête au choix de l'utilisateur.

Pour réaliser notre application nous avons utilisé le langage de programmation Eclipse et SGBD PostgreSql et PowerApi pour évaluer l'énergie de CPU.

4.2. Présentation d'Eclipse

Eclipse est un projet de la Fondation Eclipse visant à développer tout un environnement de développement libre, extensible, universel et polyvalent. Son objectif est de produire et fournir divers outils gravitant autour de la réalisation de logiciel, englobant les activités de codage logiciel proprement dites (avec notamment un environnement de développement intégré) mais aussi de modélisation, de conception, de test, de reporting, etc. Son environnement de développement notamment vise à la généricité pour lui permettre de supporter n'importe quel langage de programmation.

Le projet Eclipse est pour cela organisé en un ensemble cohérent de projets logiciels distincts, sa spécificité tenant à son architecture totalement développée autour de la notion de plugin (en conformité avec la norme OSGi) : toutes les fonctionnalités de l'atelier logiciel doivent être développées en tant que *plug-in* bâti autour de l'**IDE Eclipse Platform**.

Eclipse recouvre donc notamment également à cet effet tout un framework de développement logiciel fournissant des briques logicielles à partir desquelles développer tous ces outils. C'est la raison pour laquelle Eclipse est présenté dans la littérature tout autant comme un EDI ou comme un Framework [52].

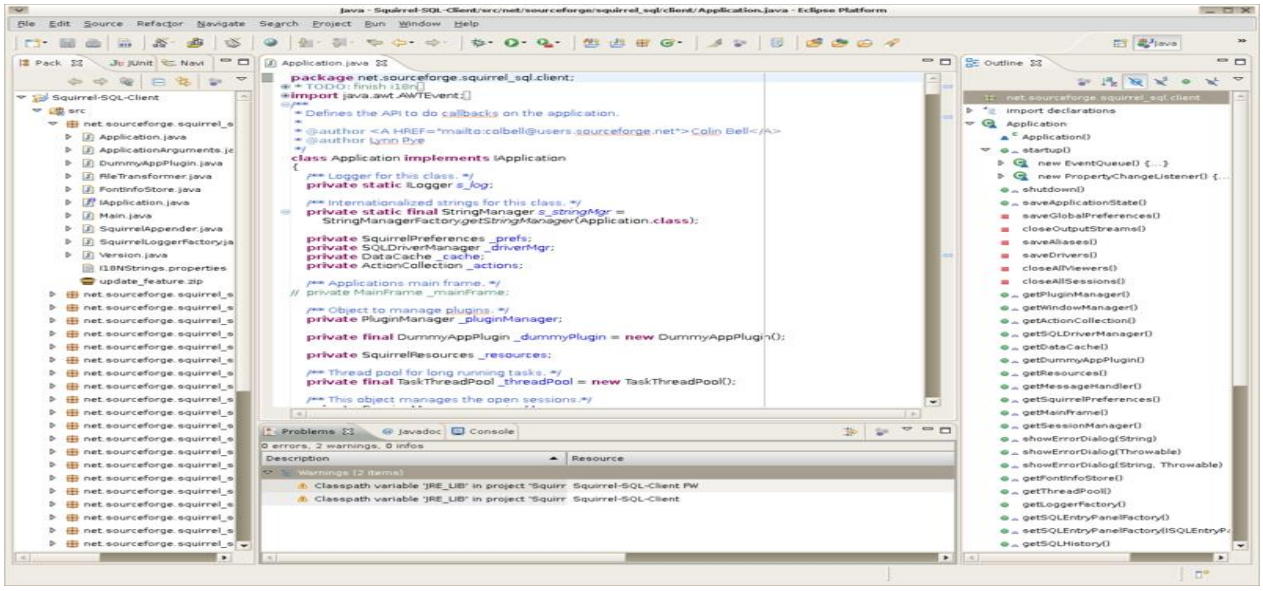

Figure 4.1 interface Eclipse

4.3. Présentation de PostgreSql :

Postgresql est un système de gestion de base données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD.

Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB et Firebird),ou propriétaires
(comme Oracle, MySQL, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises [53].

4.4. Présentation PowerApi

PowerAPI est une boîte à outils de middleware pour la création de compteurs de puissance définis par logiciel. Les compteurs de puissance définis par logiciel sont des bibliothèques de logiciels configurables pouvant estimer la consommation électrique des logiciels en temps réel. PowerAPI prend en charge l'acquisition de mesures brutes à partir d'une grande diversité de capteurs (compteurs physiques, interfaces de processeur, compteurs de matériel, compteurs de système d'exploitation) et la fourniture de consommations via différents canaux (système de fichiers, réseau, Web, graphique). En tant que kit d'outils de middleware, PowerAPI offre la possibilité d'assembler des compteurs de puissance *à la carte* pour répondre aux besoins des utilisateurs [54].

4.4. Description de l'application

4.4.1. Interface de Connexion à la base de données :

4.4.1.1. Choisir la taille de la base de données: l'utilisateur va choisir la taille de base de données 1G ou 10G.

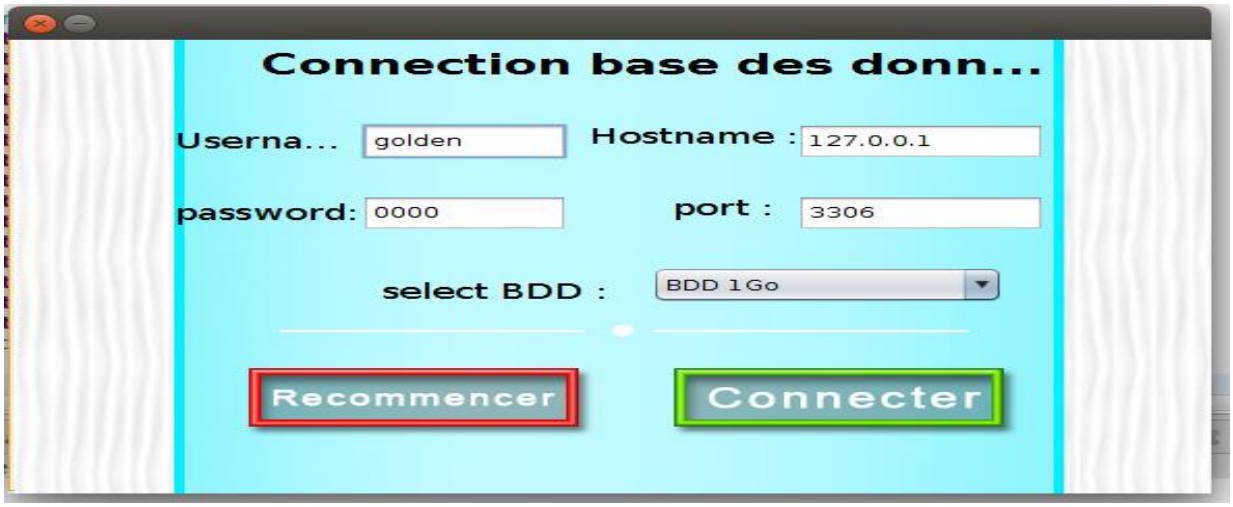

Figure 4.2 Interface de Connexion à la base de données

4.4.1.2. Présentation de la base de données benchmarks :

Dans notre travail, nous utilisons les données et les requêtes issues d'un benchmark TPC-H [55], interrogeant des données de 1 Go et de 10 Go de taille. Le TPC-H est un benchmark décisionnel conçu pour le requêtage ad-hoc où les requêtes ne sont pas connues à l'avance. Le schéma modélise un modèle produits-commandes-fournisseurs contenant 2 tables de faits et 6 tables de dimensions (cf. Figure 4.3), avec une charge de 22 requêtes décisionnelles.

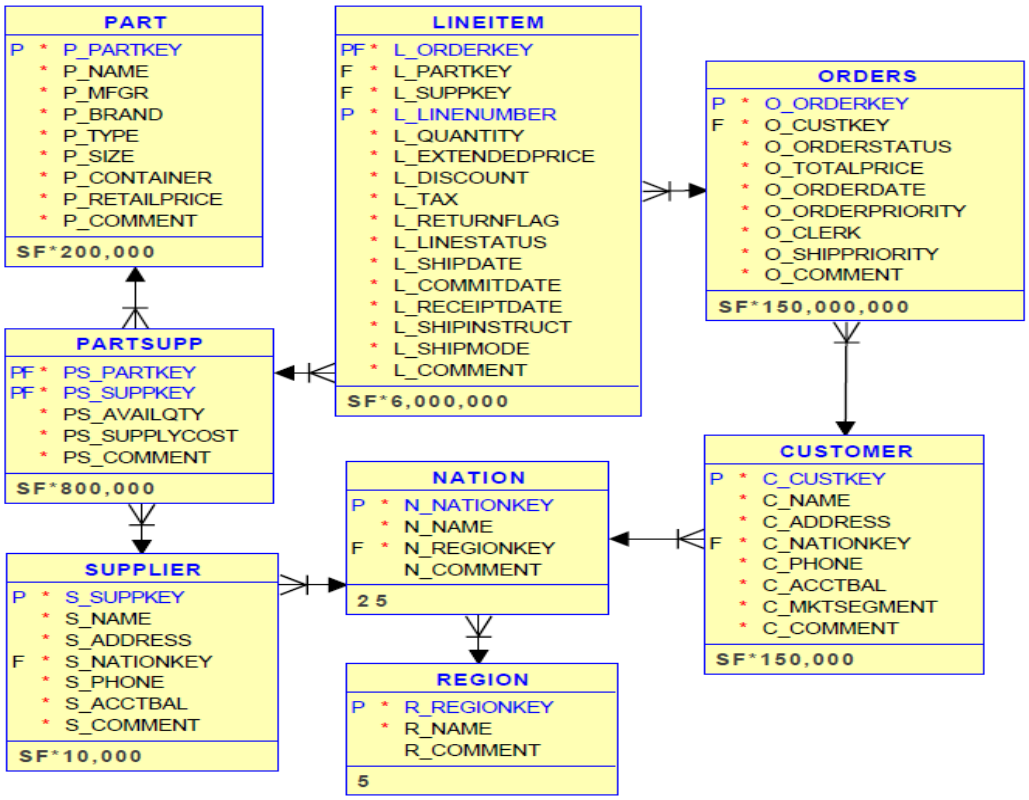

Figure 4.3 Schéma du benchmark TPC-H.

4.4.3. Interface d'exécution de la requête :

Dans cette interface l'utilisateur va sélectionner une requête à leur choix et une fréquence du CPU puis il s'exécute sur la base de données choisi précédemment. Nous obtiendrons les résultats dans un tableau concernant le temps d'exécution de chaque requête dans une fréquence de CPU choisi

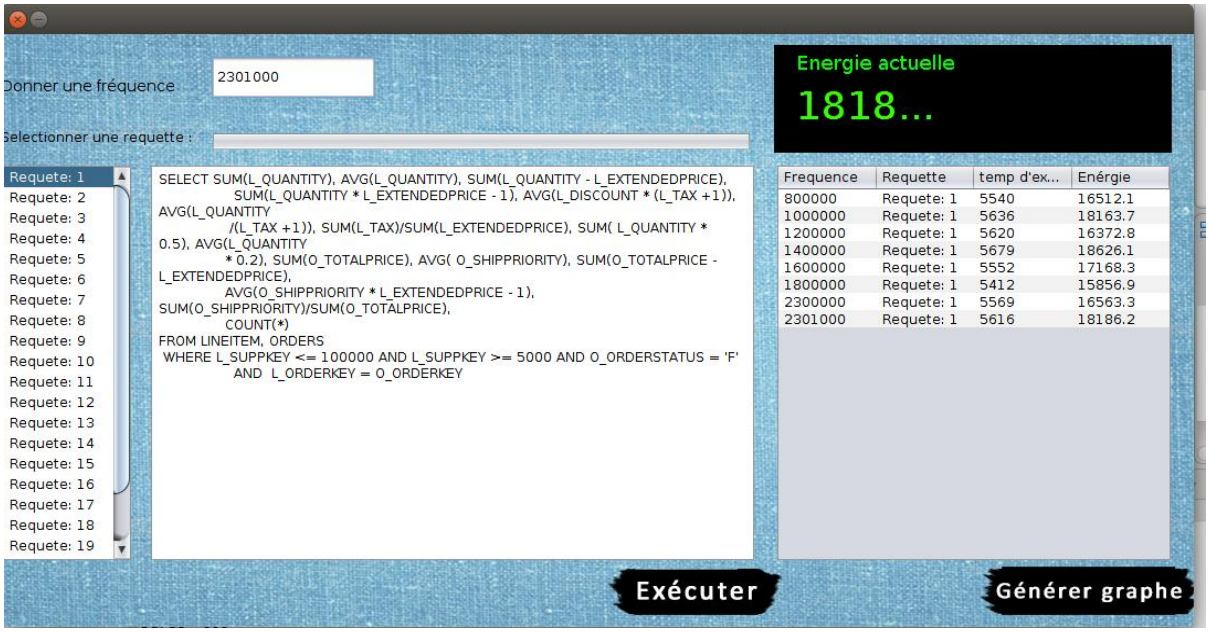

Figure 4.4 Interface d'exécution de la requête

4.4.3.1. Modifier facilement la fréquence processeur

Linux offre une possibilité intéressante : celle de configurer assez finement la vitesse du processeur depuis la ligne de commande. Il permet d'agir indépendamment (pour autant que l'architecture matérielle le permette) sur les différents cœurs des processeurs. On m'a demandé récemment une petite illustration pratique de ces possibilités : en voici un résumé.

- **Configurer la vitesse CPU avec /sys**

La première opération nécessaire est de connaître la liste des fréquences supportées par un CPU. Cette liste est accessible grâce au pseudo système de fichier /sys.

ls /sys/devices/system/cpu/

cpu0 cpu2 cpufreqkernel_maxmodalias online power uevent

cpu1 cpu3 cpuidle microcode offline possible present

cat /sys/devices/system/cpu/online

0-3

cat /sys/devices/system/cpu/possible

0-7

cat /sys/devices/system/cpu/present

0-3

Ici, le noyau peut gérer sept CPU (paramètre possible), mais il n'y en a que quatre connectés (present). Ils sont tous actifs (onlines). Choisissons, arbitrairement, le CPU numéro 1 et vérifions ses fréquences de

fonctionnement.

```
ls /sys/devices/system/cpu/cpu1/
cachecrash_notesfirmware_node online thermal_throttle
cpufreqcrash_notes_size microcode power topology
cpuidle driver node0 subsystem uevent
ls /sys/devices/system/cpu/cpu1/cpufreq/
affected_cpusfreqdomain_cpusscaling_governor
bios_limitrelated_cpusscaling_max_freq
cpuinfo_cur_freqscaling_available_frequenciesscaling_min_freq
cpuinfo_max_freqscaling_available_governorsscaling_setspeed
cpuinfo_min_freqscaling_cur_freq stats
cpuinfo_transition_latencyscaling_driver
cat /sys/devices/system/cpu/cpu1/cpufreq/scaling_available_frequencies
2301000 2300000 1800000 1600000 1400000 1200000 1000000 800000
cat /sys/devices/system/cpu/cpu1/cpufreq/scaling_cur_freq
1800000
```

Le CPU accepte huit fréquences, il est actuellement configuré avec la fréquence 1.8 GHz. On peut la modifier ainsi.

```
echouserspace> /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
echo 2301000 > /sys/devices/system/cpu/cpu1/cpufreq/scaling_setspeed
cat /sys/devices/system/cpu/cpu1/cpufreq/scaling_cur_freq
2301000
```

➢ Configurer le comportement du CPU

Plutôt qu'agir directement sur la fréquence, ce qui nécessite de lire la valeur et d'inscrire celle choisie, le noyau nous propose des comportements, des heuristiques, qu'il nomme governors, qui vont assurer la modification de la fréquence CPU afin de l'ajuster aux besoins de l'utilisateur.
On trouvera des explications plus détaillées sur les governors dans cet article. Il existe (suivant la configuration du noyau à la compilation) jusqu'à cinq governors, dont les noms sont visibles ainsi.

```
cat /sys/devices/system/cpu/cpu1/cpufreq/scaling_available_governors
conservativeondemanduserspacepowersaveperformance
```

➢ Leurs comportements :

powersave : économiser la batterie (d'un ordinateur portable) en utilisant la fréquence la plus faible possible.

performance : optimiser la vitesse de traitement en adoptant la fréquence disponible la plus élevée.
ondemand : faire varier la fréquence en fonction de la charge système afin d'optimiser la vitesse de traitement lorsqu'il y a une forte demande, tout en économisant la batterie lorsqu'il y a peu de tâches en cours.
conservative : adopter le même comportement que ondemand, en évitant les modifications trop fréquentes de la vitesse du CPU.
userspace : laisser l'utilisateur fixer lui-même la vitesse qui lui convient,

echo performance > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
cat /sys/devices/system/cpu/cpu1/cpufreq/scaling_cur_freq
2301000
Afficher les max_freq et min_freq supportées
sudo cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_max_freq
2301000
sudo cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_min_freq
800000

Il est intéressant de pouvoir configurer facilement la vitesse du processeur indépendamment des fréquences exactes, en employant par exemple une heuristique powersave sur un portable, performance sur un serveur, ou ondemand sur un poste de travail généraliste.

4.4.4. Interface de génération d'un graphe (l'énergie par a port temps de réponse) :

Quand le système compte le temps de réponse et l'énergie consommer par le CPU pendant l'exécution la requête choisi par l'utilisateur .Si l'utilisateur veut savoir l'énergie et temps de réponse moyenne en cliquant sur le bouton « générer graphe » et celui-ci va apparaitre.

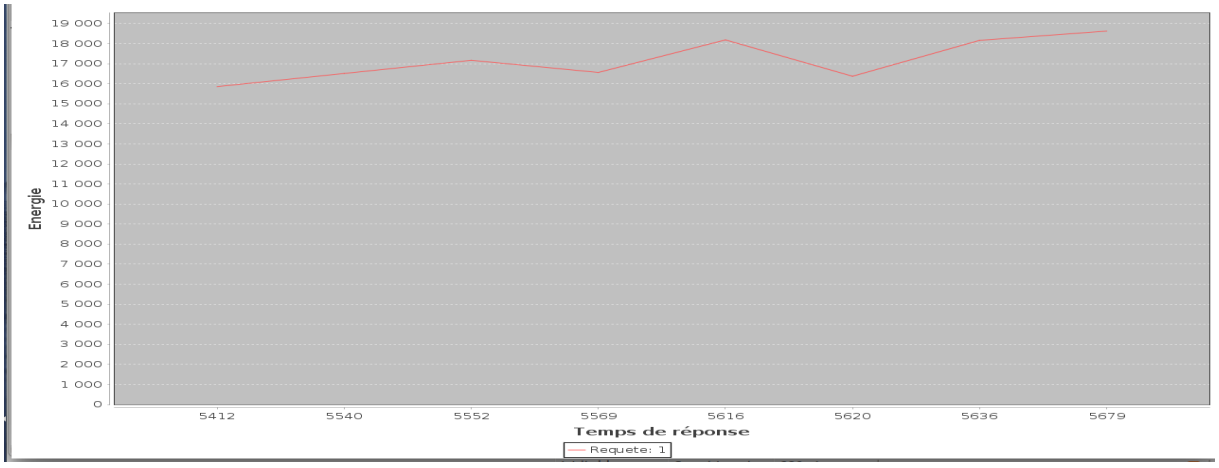

Figure 4.5 Interface de génération d'un graphe

4.5. Graphe de la puissance Moyenne et temps Moyen pour neufsrequêtes (Taille de la BDD 1G0) avec tous les fréquences du CPU:

A partir de graphe générer par l'application de puissance par a port au temps de réponse on génère le graphe de puissance moyenne et de temps moyen. On remarque que l'énergie consommé augmente quand la requête est complexe et même pour le temps de réponse.

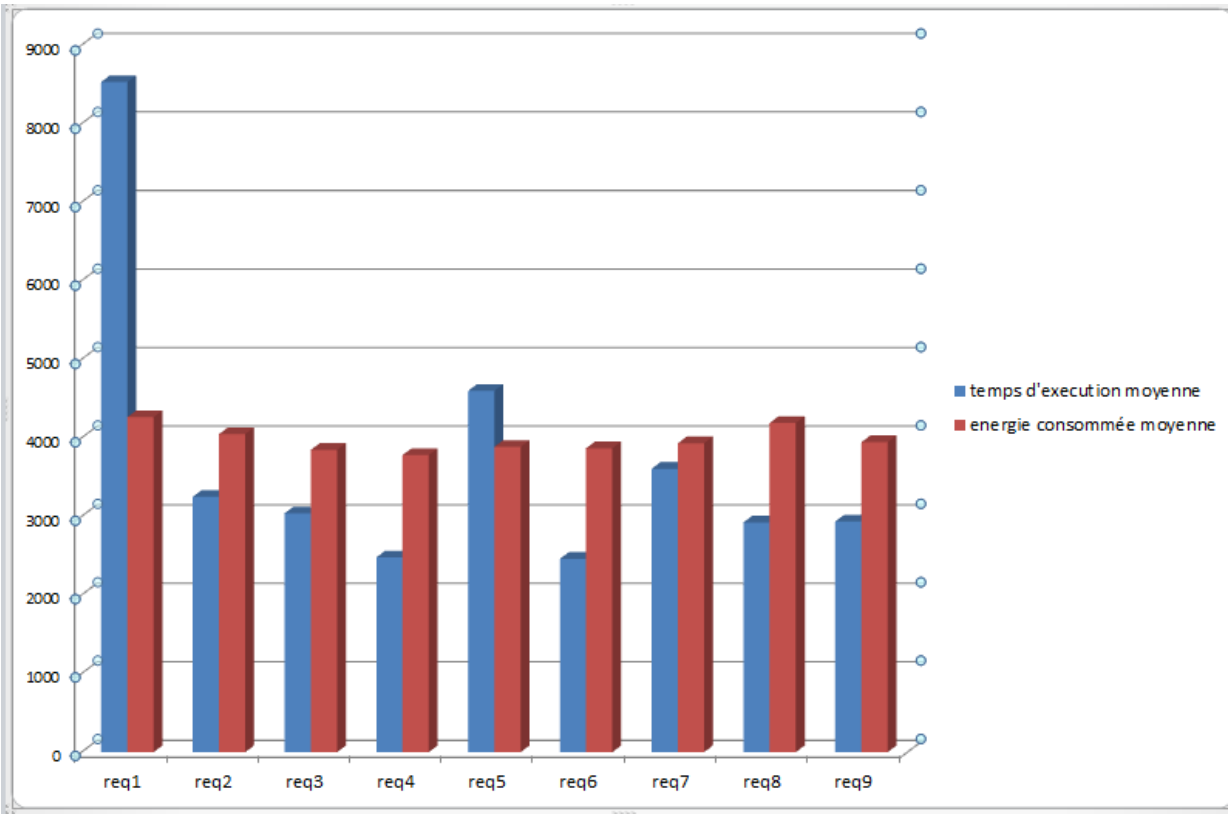

Figure 4.6 Graphe de puissance Moyenne et temps Moyen pour neufs requêtes

4.6. Graphe de puissance moyenne et temps moyen pour neufs requêtes avec la taille de base des données 1G et 10G : dans ce graphe onvoit que quand la taille de base de données est volumineuse le CPU consomme beaucoup d'énergie et le temps de réponse augmente.

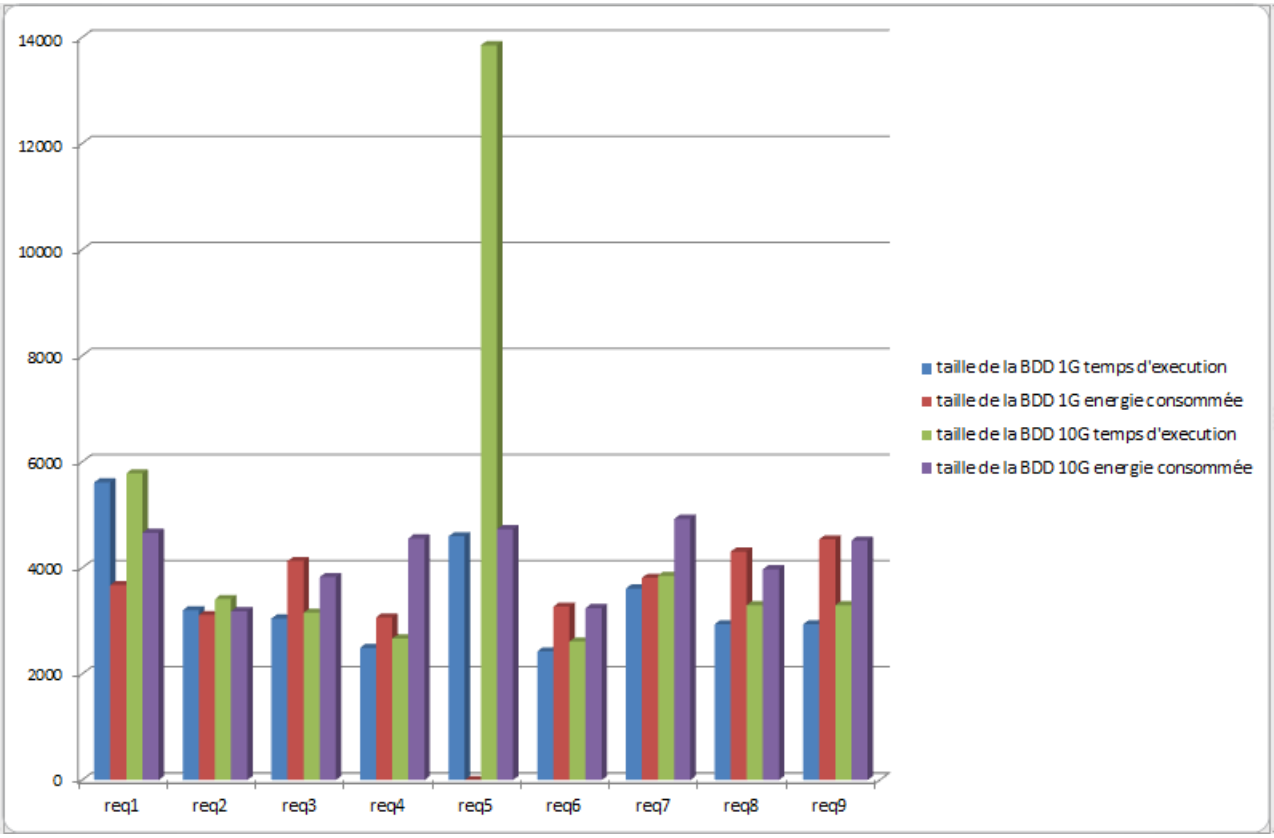

Figure 4.7 Graphe de puissance Moyenne et temps Moyen pour neufs requêtes avec taille de base des données 1G et 10Gavec une fréquence maximal de CPU= 2301000Hz

4.5. Conclusion

Dans ce chapitre nous avons présenté l'environnement technique de développement que nous avons utilisé pour mettre en œuvre notre application puis nous avons décrit les principales interfaces de notre logiciel.

Conclusion Générale :

Les systèmes informatiques d'aujourd'hui considère la quantité de puissance électrique qu'ils consomment ne s'ajuste pas en fonction de la quantité de travail que le système réalise. Jusqu'à présent, le principal objectif de la plupart des systèmes informatiques à usage général était de maximiser les performances sans tenir compte de la consommation d'énergie. Nous nous sommes intéressés dans ce mémoire au problème de consommation d'énergie de CPU lors d'exécutions des requêtes dans les bases de données avancées afin de minimiser leur consommation en utilisant la technique DVFS.

L'objectif principal de ce mémoire se situe dans le cadre de la réalisation d'un outil pour minimiser la consommation d'énergie d'un CPU lors d'exécution d'une requête dans la base de données benchmark.

Notre application met à l'utilisateur la possibilité de choisir une fréquence CPU et d'exécuter une requête à leurs choix.

Ce travail nous a permis d'acquérir des connaissances dans un domaine en vague et aussi large qu'est le domaine de consommation d'énergie dans le système informatique, d'appliquer le langage UML et de familiarise avec le langage Eclipse.

ANNEXE : REQUÊTES D'APPRENTISSAGE DU MODÈLE DE COÛT

Requêtes d'apprentissage

Les requêtes suivantes sont les requêtes utilisées pour construire notre modèle de coûténergétiquelors de la phase d'apprentissage. Ils sont créés à partir du schéma de benchmarkTPC-H [55] avec unetaille de 10 Go.

```
1 -- Q1
2 SELECT SUM(L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE),
SUM(L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)), AVG(L_QUANTITY
/(L_TAX +1)), SUM(L_TAX)/SUM(L_EXTENDEDPRICE), SUM( L_QUANTITY * 0.5), AVG(L_QUANTITY
* 0.2),SUM(O_TOTALPRICE), AVG( O_SHIPPRIORITY), SUM(O_TOTALPRICE - L_EXTENDEDPRICE),
AVG(O_SHIPPRIORITY * L_EXTENDEDPRICE - 1), SUM(O_SHIPPRIORITY)/SUM(O_TOTALPRICE),
COUNT(*)
3FROM LINEITEM, ORDERS
4WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'F'
AND  L_ORDERKEY = O_ORDERKEY;
5
6  -- Q2
7 SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS
8FROM LINEITEM, ORDERS
9WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'F' AND
            L_ORDERKEY = O_ORDERKEY
10GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS;
11
12 -- Q3
13SELECTCOUNT(*)
14FROM LINEITEM, ORDERS
15WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND L_ORDERKEY = O_ORDERKEY;
16
17 -- Q4
18SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS
19FROM LINEITEM, ORDERS
20WHERE L_SUPPKEY <= 1000 AND L_ORDERKEY = O_ORDERKEY
21GROUPBY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS;
22
23 -- Q5
24SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, COUNT(*)
25FROM LINEITEM, ORDERS
26WHERE L_SUPPKEY <= 10000 AND L_SUPPKEY >= 500 AND O_ORDERSTATUS = 'F' AND
            L_ORDERKEY = O_ORDERKEY
27GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS
28ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS;
29
30 -- Q6
31SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, SUM(L_QUANTITY), AVG(
            L_QUANTITY), SUM(O_TOTALPRICE), COUNT(*)
32FROM LINEITEM, ORDERS
33WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY
34GROUPBY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS
35ORDERBY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS;
36
37 -- Q7
38SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, SUM(L_QUANTITY), AVG(
            L_QUANTITY), SUM(O_TOTALPRICE), COUNT(*)
39FROM LINEITEM, ORDERS
40WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'F' AND
           L_ORDERKEY = O_ORDERKEY
41GROUPBY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS
42ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS;
```

```
43
44 -- Q8
45 SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY, SUM(
             L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE), SUM(
           L_QUANTITY * L_EXTENDEDPRICE - 1), SUM(O_TOTALPRICE),
AVG(O_SHIPPRIORITY), COUNT(*)
46FROM LINEITEM, ORDERS
47WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY
48GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY
49ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY;
50
51 -- Q9
52SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY, SUM(
           L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE), SUM(
           L_QUANTITY * L_EXTENDEDPRICE - 1), SUM(O_TOTALPRICE), AVG(
          O_SHIPPRIORITY), COUNT(*)
53FROM LINEITEM, ORDERS
54WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'P' AND
           L_ORDERKEY = O_ORDERKEY
55GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY
56ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY;
57
58 -- Q10
59SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY, SUM(
           L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE), SUM(
           L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)), AVG(
          L_QUANTITY / (L_TAX +1)), SUM(O_TOTALPRICE), AVG(O_SHIPPRIORITY), SUM
          (O_TOTALPRICE - L_EXTENDEDPRICE), AVG(O_SHIPPRIORITY *
          L_EXTENDEDPRICE - 1), COUNT(*)
60FROM LINEITEM, ORDERS
61WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY
62GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY
63ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY;
64
65 -- Q11
66SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY, SUM(
           L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE), SUM(
           L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)), AVG(
           L_QUANTITY / (L_TAX +1)), SUM(O_TOTALPRICE), AVG(O_SHIPPRIORITY), SUM
          (O_TOTALPRICE - L_EXTENDEDPRICE), AVG(O_SHIPPRIORITY *
          L_EXTENDEDPRICE - 1), COUNT(*)
67FROM LINEITEM, ORDERS
68WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'P' AND
           L_ORDERKEY = O_ORDERKEY
69GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY
70ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY;
71
72 -- Q12
73SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY, SUM(
           L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE), SUM(
           L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)), AVG(
          L_QUANTITY / (L_TAX +1)), SUM(L_TAX)/SUM(L_EXTENDEDPRICE), SUM(
          L_QUANTITY * 0.5), AVG(L_QUANTITY * 0.2), SUM(O_TOTALPRICE), AVG(
         O_SHIPPRIORITY), SUM(O_TOTALPRICE - L_EXTENDEDPRICE), AVG(
        O_SHIPPRIORITY * L_EXTENDEDPRICE - 1), SUM(O_SHIPPRIORITY)/SUM(
        O_TOTALPRICE), COUNT(*)
74FROM LINEITEM, ORDERS
75WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY
76GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY
77ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY;
78
79 -- Q13
80SELECTCOUNT(*)
```

```
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 1000 AND L_ORDERKEY = O_ORDERKEY;

-- Q14
SELECT L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY, SUM(
        L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE), SUM(
        L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)), AVG(
        L_QUANTITY / (L_TAX +1)), SUM(L_TAX)/SUM(L_EXTENDEDPRICE), SUM(
        L_QUANTITY * 0.5), AVG(L_QUANTITY * 0.2), SUM(O_TOTALPRICE), AVG(
        O_SHIPPRIORITY), SUM(O_TOTALPRICE - L_EXTENDEDPRICE), AVG(
        O_SHIPPRIORITY * L_EXTENDEDPRICE - 1), SUM(O_SHIPPRIORITY)/SUM(
        O_TOTALPRICE), COUNT(*)
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'P' AND
        L_ORDERKEY = O_ORDERKEY
GROUP BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY
ORDER BY L_SHIPMODE, L_SHIPINSTRUCT, O_ORDERSTATUS, O_ORDERPRIORITY;

-- Q15
SELECTSUM(O_TOTALPRICE)
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 10000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY;

-- Q16
SELECTSUM(L_QUANTITY), AVG(L_QUANTITY), SUM(O_TOTALPRICE), COUNT(*)
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY;

-- Q17
SELECTSUM(L_QUANTITY), AVG(L_QUANTITY), SUM(O_TOTALPRICE), COUNT(*)
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'F' AND
          L_ORDERKEY = O_ORDERKEY;

-- Q18
SELECTSUM(L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE),
        SUM(L_QUANTITY * L_EXTENDEDPRICE - 1), SUM(O_TOTALPRICE), AVG(
          O_SHIPPRIORITY), COUNT(*)
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY;

-- Q19
SELECTSUM(L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE),
SUM(L_QUANTITY * L_EXTENDEDPRICE - 1), SUM(O_TOTALPRICE), AVG(
O_SHIPPRIORITY), COUNT(*)
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'F' AND
L_ORDERKEY = O_ORDERKEY;

-- Q20
SELECTSUM(L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE),
SUM(L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)),
AVG(L_QUANTITY / (L_TAX +1)), SUM(O_TOTALPRICE), AVG(O_SHIPPRIORITY),
SUM(O_TOTALPRICE - L_EXTENDEDPRICE), AVG(O_SHIPPRIORITY *
L_EXTENDEDPRICE - 1), COUNT(*)
FROM LINEITEM, ORDERS
WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY;

-- Q21
SELECTSUM(L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE),
SUM(L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)),
AVG(L_QUANTITY / (L_TAX +1)), SUM(O_TOTALPRICE), AVG(O_SHIPPRIORITY),
```

```
        SUM(O_TOTALPRICE - L_EXTENDEDPRICE), AVG(O_SHIPPRIORITY *
        L_EXTENDEDPRICE - 1), COUNT(*)
123FROM LINEITEM, ORDERS
124WHERE L_SUPPKEY <= 100000 AND L_SUPPKEY >= 5000 AND O_ORDERSTATUS = 'F' AND
        L_ORDERKEY = O_ORDERKEY;
125
126 -- Q22
127SELECTSUM(L_QUANTITY), AVG(L_QUANTITY), SUM(L_QUANTITY - L_EXTENDEDPRICE),
        SUM(L_QUANTITY * L_EXTENDEDPRICE - 1), AVG(L_DISCOUNT * (L_TAX +1)),
        AVG(L_QUANTITY / (L_TAX +1)), SUM(L_TAX)/SUM(L_EXTENDEDPRICE), SUM(
L_QUANTITY * 0.5), AVG(L_QUANTITY * 0.2), SUM(O_TOTALPRICE), AVG(
O_SHIPPRIORITY), SUM(O_TOTALPRICE - L_EXTENDEDPRICE), AVG(
        O_SHIPPRIORITY * L_EXTENDEDPRICE - 1), SUM(O_SHIPPRIORITY)/SUM(
O_TOTALPRICE), COUNT(*)
128FROM LINEITEM, ORDERS
129WHERE L_SUPPKEY <= 1000 AND L_SUPPKEY >= 500 AND L_ORDERKEY = O_ORDERKEY;
```

Bibliographie :

[1]: David Mourjanhttp://infiniment-it-fujitsu.com/linformatique-resoudre-puzzle-de-lenergie-mondiale ; 2017

[2]: D Riva GeSISMARTer 2020 the Role of ICT in Driving a Sustainable Future ;2012

[3]: C Coronel, S Morris Databassystems: design, implementation, et management ; 2016

[4]: L Toumi Optimisation des performances par administration et tuning d'entrepôt de données ; 2018

[5] Fabien DOUCHET Optimisation énergétique de data centers par utilisation de liquides pour le refroidissement des baies informatiques ; 2015

[6] S Agrawal, S Chaudhuri, VR NarasayyaAutomated selection of materialized views and indexes in SQL databases ; 2000

[7] K Park, YS Kee, JM Patel, J Do, C Park Query Processing on Smart SSDs.

[8] N Bame, H Naacke, I Sarr, S Ndiaye Architecture répartie à large échelle pour le traitement parallèle de requête de biodiversité2012

[9] R Elmasri- Pearson Education India Fundamentals of database systems

[10] I Manolescu, S ManegoldPerformance evaluation in database research: principles and experience

[11] L Audibert Bases de données de la modélisation au SQL ; 2009

[12] IF Sbalzarini, S Müller Multiobjective optimization using evolutionary algorithms; 2000

[13] MT Özsu , P ValduriezPrincipes des systèmes de bases de données distribuées ; 2011.

[14] S ChaudhuriUn aperçu de l'optimisation des requêtes dans les systèmes relationnels,1998.

[15] R Singhal, M NambiarPrédiction du temps d'exécution des requêtes SQL pour un volume de données important ; 2016.

[16] MBSUF Minhas, OZ Khan, A Aboulnaga, PPDJ TaylorUne approche bayésienne de la modélisation des performances en ligne pour les appliances de bases de données utilisant des modèles gaussiens ; 2011

[17] V Nicolicin-Georgescu, V BenatierUn système autonomique basé sur des bases de connaissances pour améliorer les performances d'un entrepôt de données ;2009

[18] M Elnozahy, M Kistler, R Rajamony Politiques d'économie d’énergie pour les serveurs web ; 2003.

[19] W Yuan, K NahrstedtEnergy-efficient soft real-time CPU scheduling for mobile multimedia systems ; 2003

[20] F Yao, A Demers, S ShenkerA scheduling model for reduced CPU energy; 1995

[21] L Benini , G MicheliOptimisation de la puissance au niveau système: techniques et outils ; 2000.

[22] R Jejurikar, R GuptaMise à l'échelle de tension dynamique pour la minimisation de l'énergie dans l'ensemble du système réel - temps systèmes embarqués ; 2004.

[23] Q Deng , D Meisner , A Bhattacharjee Coordination des processeurs CPU et des dvfs de systèmes de mémoire dans les systèmes de serveurs ;2012.

[24] M Korkmaz, A Karyakin , M Karsten, K SalemVers un dimensionnement dynamique des serveurs de bases de données ; 2015.

[25] M Dayarathna , Y Wen , R Fan Modélisation de la consommation énergétique des centres de données: une enquête; 2016.

[26] Y Kim , A Gupta , B Urgaonkar , P Bermanun système de stockage performant et économique combinant SSD et disques durs ; 2011.

[27] J Shuja , K Bilal , SA Madani , M OthmanEtude des techniques et des architectures pour la conception de datacenters économes en énergie ;2016.

[28] F GruianHard real-time scheduling for low-energy using stochastic data and DVS processors ; 2001.

[29] C Lefurgy , X Wang , M WareContrôle de l'alimentation au niveau du serveur ; 2007.

[30] Y Chen , S Alspaugh , R KatzTraitement analytique interactif dans les systèmes de données volumineuses: étude intersectorielle des charges de travail mapréduites;2012.

[31] M Lin , A Wierman , LLH AndrewDimensionnement dynamique pour les datacenters à puissance proportionnelle ;2013.

[32] W Lang , S Harizopoulos , JM Patel , MA Shah Vers un cluster de base de données économe en énergie;2012.

[33] R Weiss A technical overview of the oracle exadata database machine and exadata storage server; 2012.

[34] M Korkmaz, A Karyakin , M Karsten, K SalemVers un dimensionnement dynamique des serveurs de bases de données. ; 2015.

[35] J Shuja, K Bilal, SA Madani, M OthmanSurvey of techniques and architectures for designing energy-efficient data centers; 2016.

[36] SK Cheong, CS Lim Performance de traitement de base de données et évaluation de l'efficacité énergétique du système de stockage DDR - SSD et HDD basé sur le TPC - C ; 2012.

[37] J Arulraj , A Pavlo Comment construire un système de gestion de base de données en mémoire non volatile ; 2017.

[38] AY HalevyRépondre aux requêtes à l'aide de vues, 2000.

[39] Z Xu , YC Tu , X WangEstimation dynamique de l'énergie des plans de requête dans les systèmes de base de données ; 2013.

[40] W Lang , R Kandhan, JM PatelRepenser le traitement des requêtes pour l'efficacité énergétique: ralentir pour gagner la course ; 2011.

[41] Z Ou , B Pang, Y Deng, JK NurminenAnalyse de l'efficacité énergétique et des coûts des grappes basées sur lesarmements; 2012.

[42] SY Ihm, A Nasridinov, JH Lee , YH Park Appariement ultérieur basé sur la dualité efficace sur des données de séries chronologiques en informatique verte; 2014.

[43] AAllavena, D MosséScheduling of frame-based embedded systems with rechargeable batteries; 2001

[44] Liu S, Lu J, Wu Q et Qiu Q Harvestingaware power management for real-time systems with renewable energy; 2012.

[45] Moser C ,Brunelli D, Thiele L et Benini L Realtime scheduling with regenerative energy; 2006.

[46] Moser C, Brunelli D, Thiele L. et Benini L Realtime scheduling for energy harvesting sensor nodes; 2007.

[47] Chetto M, El Ghor H et ChehadeRH Realtime scheduling for energy harvesting sensors ;2011.

[48] Lehoczky JP et Ramos-Thuel S An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems; 1992.

[49] Liu S, Qiu Q et Wu Q Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting, 2008.

[50] A Roukh Prise en compte de l’énergie dans la phase d’exploitation des bases de données volumineuses ; 2017.

[51] K Molokken, M JorgensenA review of software surveys on software effort estimation; 2003.

[52] www.informatique.systems/sites/default/files/eclipse-pdt.pdf.

[53] www.aditu.fr/portail-des-metiers/systemes-bases-de-donnees/postgresql.

[54] https://github.com/Spirals-Team/powerapi.

[55] http://www.tpc.org/tpch/ (cf. p. 94, 113, 129, 207).

Résumé :

La gestion de la consommation d'énergie par les serveurs et les centres de données est devenue un défi majeur pour les entreprises, les institutions et les pays. Parmi les applications déployées sur les centrées données, les systèmes de gestion de base de données (SGBD) sont l'un des principaux consommateurs d'énergie lors de l'exécution des requêtes complexes impliquant de très grandes taille de données. Par ailleurs, le traitement de ce type de base de données requiert des infrastructures informatiques et matérielles coûteuses et consommatrices d'énergie. Le facteur le plus important pour l'utilisateur est la minimisation du temps de réponse de requêtes.
Dans ce mémoire nous proposons une formalisation multi-objective des problèmes d'exploitation des bases de données, en tenant compte de deux besoins non-fonctionnels : la performance et la consommation d'énergie lors de l'exécution d'une charge de requêtes. Pour cela nous avons utilisé la technique DVFS pour réduire l'énergie consommer par CPU et minimiser le temps de réponse lors d'exécution d'une requête dans des bases de données benchmarks TPC-H de taille 1G et 10G.

Mots clés : Efficacité énergétique, modèles de coût, traitement de requêtes, optimisation multi-objectifs.

Abstract :

Managing power consumption by servers and data centers has become a major challenge for businesses, institutions and countries. Among the applications deployed on data centers, database management systems (DBMS) are one of the main energy consumers when running complex queries involving very large data sizes. In addition, the processing of this type of database requires costly and energy-intensive IT and hardware infrastructures. The most important factor for the user is the minimization of the query response time.
In this thesis we propose a multi-objective formalization of database exploitation problems, taking into account two non-functional needs: performance and energy consumption during the execution of a request load. For this we used the DVFS technique to reduce the energy consumed per CPU and minimize the response time when executing a query in 1G and 10G TPC-H benchmark data bases.

Keywords : Energy efficiency, cost models, query processing, multi-objective optimization.