Efficacité énergétique des phases de conception et d'exploitation des entrepôts de données

Issam Ghabri

THESE

Pour l'obtention du Grade de
DOCTEUR DE L'ECOLE NATIONALE SUPERIEURE DE MECANIQUE ET D'AEROTECHNIQUE
(Diplôme National – Arrêté du 25 mai 2016 Modifié par l'Arrêté du 26 Août 2022)

Ecole Doctorale :
Sciences et Ingénierie pour l'Information et Mathématiques

Secteur de Recherche : Informatique et Applications

Présentée par :

Issam GHABRI

****

Efficacité énergétique des phases de conception et d'exploitation des entrepôts de données.

****

Directeur(s)de thèse : Ladjel BELLATRECHE et Sadok BEN YAHIA

****

Soutenue le 29 Décembre 2022

devant la Commission d'Examen

****

JURY

Président :
Borgi Amel, Professeure, ISI, Tunis, Tunisie
Rapporteurs :
Amous Ikram, Professeure, Enet'Com, Sfax, Tunisie
Layaida Nabil, Directeur de Recherches, Inria, France
Membres du jury :
Faiz Rim, Professeure, IHEC, Tunis, Tunisie
Benslimane Djamal, Professeur, Univ-Lyon, Lyon1, France
Boukhalfa Kamel, Professeur, USTHB, Alger, Algérie
Ben Yahia Sadok, Professeur, FST, Tunis, Tunisie
Bellatreche Ladjel, Professeur, ISAE-ENSMA, Poitiers, France

Remerciements

Avant tout développement sur cette expérience, il apparaît opportun de commencer ce rapport par des remerciements, à ceux qui m'ont beaucoup appris au cours de ces années de thèse, et même à ceux qui ont eu la gentillesse d'en faire un moment très profitable.

. . .

Je tiens à adresser mes vifs remerciements et à témoigner toute ma reconnaissance à mes directeurs de thèse, Professeur *Sadok BEN YAHIA* et Professeur *Ladjel BELLATRECHE*, pour leurs conseils et disponibilités, pour avoir dirigé mon travail, pour leurs encouragements et leurs soutiens continus. L'intérêt qu'ils ont manifesté pour mon travail, leurs suggestions, et leurs remarques ont été d'une importance capitale. Sous leur direction, j'ai appris davantage de rigueur, de sens critique, de la discipline et surtout la patience.

. . .

Je tiens aussi à remercier tous ceux qui ont contribué de près ou de loin à l'aboutissement de cette thèse, ceux qui m'ont toujours soutenue et ceux qui ont pris part de leur temps pour m'aider à élaborer ce travail.

. . .

Une pensée particulière pour mes collègues, les jeunes docteurs et les doctorants du *LIPAH* et du *LIAS*, sans exception, pour leur présence, leur gentillesse et les moments agréables et parfois difficiles que nous avons vécu ensemble

. . .

Mes derniers remerciements et non les moindres, iront à mes proches, et en particulier à ma mère, mon père, ma sœur, mes cousins , tous les membres de ma famille et mes amis qui m'ont toujours apporté leur soutien sans faille. Je les remercie de toute l'affection et tout l'amour qu'ils m'ont témoigné.

Table des matières

Table des figures

Liste des tableaux

Liste des Abréviations

AFO	**A**verage **F**an-**O**ut
BD	**B**ase de **D**onnées
BnF	**B**esoin **N**on **F**onctionnel
CE	**C**ommission **E**uropéenne
CFI	**C**onfiguration **F**inale d'**I**ndexes
EE	**E**fficacité **É**nergétique
E/S	**E**ntrée **S**ortie
FH	**F**ragmentation **H**orizontale
FHD	**F**ragmentation **H**orizontale **D**érivée
FH	**F**ragmentation **H**orizontale **P**rimaire
GES	**G**az à **E**ffet de **S**erre
HJ	**H**ash **J**oin
IJB	**I**ndex de **J**ointure **B**inaire
LDD	**L**angage de **D**éfinition des **D**onnées
LMD	**L**angage de **M**anipulation des **D**onnées
OR	**O**rdonnancement des **R**equêtes
PSIJB	**P**roblème de **S**élection des **I**ndexes de **J**ointure **B**inaires
PSEIJB	**P**roblème de **S**élection **E**co-énergétique des **I**ndexes de **J**ointure **B**inaires
SIA	**S**ystème de **I**mmunitaires **A**rtificiel
SO	**S**tructure d'**O**ptimisation
SGBD	**S**ystème de **G**estion de **B**ase de **D**onnée
SMJ	**S**ort **M**erge **J**oin
SSB	**S**tar **T**raitement **B**enchmark
STD	**S**ystèmes **S**chema **D**onnées
TE	**T**ravail **E**ffectué
TIC	**T**echnologies de l'**I**nformation et de la **C**ommunication
TM	**T**raverse **M**inimale
TR	**T**raitement des **R**equêtes
VM	**V**ue **M**atérialisée

Première partie

Introduction Générale

Chapitre 1

Introduction générale

1.1 Contexte et problématiques

Depuis quelques années, toutes les entreprises informatiques sont à la recherche des données, qui sont devenues une ressource importante de notre société, pour accroître leur valeur ajoutée, comme mentionné dans un article du *The Economist "la ressource la plus précieuse du monde n'est plus le pétrole, mais les données."* [1]. Hormis, comme le pétrole, les données polluent, comme l'explique le blog de *Martin Tisné* publié en 2019 (*"Data isn't the new oil, it's the new CO2"* [2]). Cette pollution est causée par le stockage et le traitement de ces données. Les technologies de l'information et de la communication (TIC) ont connu une croissance très rapide, dont l'objectif est de proposer de nouvelles solutions pour faciliter la vie quotidienne des utilisateurs. En fait, le nombre d'utilisateurs d'internet, les visiteurs des sites web, l'e-commerce, le nombre de mails envoyés, les recherches sur Google, les articles de blog, les tweets, les vidéos sur Youtube, les photos téléchargées sur Instagram, Facebook, Google+, Twitter, Linkedin et Pinterest, les appels Skype, les ordinateurs, les smartphones et les tablettes vendus surtout après la pandémie, et le trafic internet, augmentent chaque jour dans le monde. Par conséquent, la consommation d'énergie, l'électricité utilisée et les émissions de gaz à effet de serre (GES) dues aux TIC augmentent également de manière spectaculaire. Et en prenant en considération les larges parties de l'économie n'étant pas encore numérisées, les TIC devraient encore progresser au cours des prochaines années, et l'arrivée des économies émergentes sur le marché, les TIC sont appelées à se développer encore davantage au cours de la prochaine décennie.

Selon l'organisation météorologique mondiale (OMM) [3], la température moyenne annuelle mondiale en 2021 était d'environ 1.11 ± 0.13 °C au-dessus de la moyenne pré-industrielle de 1850-1900. Les sept années les plus récentes, de 2015 à 2021, sont les sept années les plus chaudes jamais enregistrées et les vagues de chaleur exceptionnelles ont battu des records dans l'ouest de l'Amérique du Nord et en Méditerranée. En Californie, la température a atteint 54,4 °C le 9 juillet, égalant ainsi la valeur la plus élevée enregistrée dans le monde depuis au moins les années 30. Les répercussions de cette augmentation de température, sur notre planète, sont graves, en raison des catastrophes naturelles arrivées ces dernières années. La variation de la température lors de ces quatre dernières décennies est donnée dans la figure 1.1.

Le niveau moyen mondial de la mer a atteint un nouveau record en 2021, après avoir augmenté en moyenne de 4.5 mm par an sur la période 2013-2021. Ce taux est plus que le double de celui enregistré entre 1993 et 2002 et il est principalement dû à la perte accélérée de la masse de glace des calottes glaciaires. La sécheresse a touché de nombreuses régions du monde, notamment l'Afrique, le Canada, l'ouest des États-Unis, et la Turquie. En Amérique du Sud subtropicale, la sécheresse a causé de grosses pertes agricoles et a perturbé la production d'énergie et le transport fluvial. La sécheresse dans la Corne de l'Afrique s'est intensifiée jusqu'à présent en 2022. L'Afrique de l'Est est confrontée à la perspective très réelle d'un manque de pluies pour une quatrième saison consécutive, plaçant l'Éthiopie, le Kenya et la Somalie dans une sécheresse d'une longueur jamais connue au cours des 40 dernières années. Les agences humanitaires mettent en garde contre les effets dévastateurs sur les populations et les moyens de subsistance dans la région. Pour résumer, les effets des conflits régionaux, les événements climatiques extrêmes et les chocs économiques, encore aggravés par la pandémie de COVID-19, ont sapé des décennies de progrès vers l'amélioration de la sécurité alimentaire dans le monde. L'aggravation des crises humanitaires en 2021 a également conduit à un nombre croissant de pays à risque de famine.

1. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data
2. https://luminategroup.com/posts/blog/data-isnt-the-new-oil-its-the-new-co2
3. https://public.wmo.int/en/media/press-release/four-key-climate-change-indicators-break-records-2021

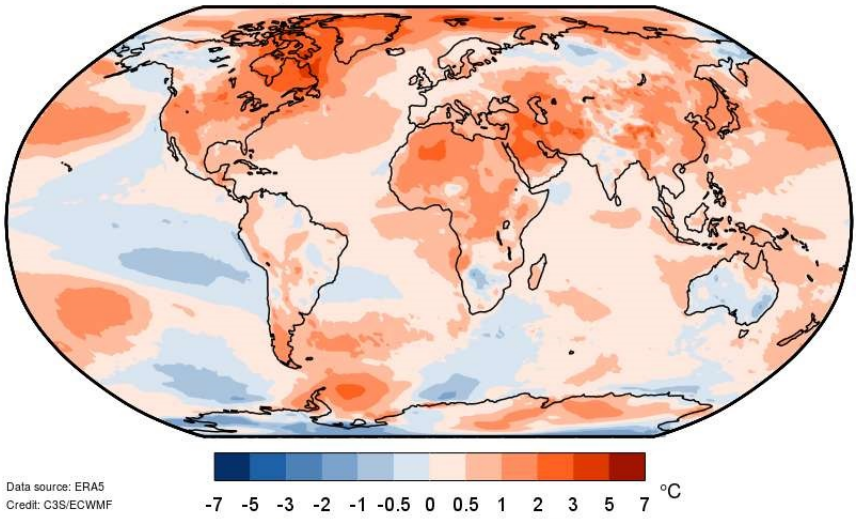


FIGURE 1.1 – L'augmentation de la température entre 1981-2010 et 2021.

2021 a été la deuxième pire saison de feux de forêt dans l'Union européenne depuis l'année 2000, et les dommages causés n'ont été dépassés que par ceux de 2017, année où plus d'un million d'hectares (*ha*) sont brûlés dans l'UE. Les incendies importants et extrêmes ont touché de nombreux pays, notamment dans le bassin méditerranéen, ce qui a mis en garde les dirigeants contre les conditions dangereuses actuelles qui peuvent alimenter les feux de forêt. En 2021, des incendies ont été observés dans 39 pays, brûlant 1113464 *ha*. La Turquie a été la plus touchée par les feux de forêt, avec 206013 *ha* brûlés par un grand nombre d'incendies massifs. Elle était suivie par l'Italie, avec 159537 *ha* et un record de 1422 incendies, près de quatre fois la moyenne des 13 dernières années [4].

Aujourd'hui, une grande majorité de personnes, notamment les jeunes, sont conscients de la gravité du changement climatique (*CC*). Les jeunes qui sont considérés comme la population clé pour mener la lutte contre le *CC*, sont les chercheurs, les décideurs, et les futurs leaders de demain [5]. La situation s'est aggravée, principalement pendant la pandémie de COVID-19, où plusieurs scientifiques ont établi un lien entre cette pandémie et le *CC*. Par exemple, *Bill Gates* a lancé un avertissement sévère au monde : "*Aussi terrible que soit cette pandémie, le changement climatique pourrait être pire*" [6]. Pour réduire les problèmes environnementaux de l'informatique et créer un environnement durable, *Murugesan* a appelé le secteur des TIC ainsi que tous les utilisateurs d'ordinateurs d'écologiser leurs systèmes informatiques, ainsi que la manière dont ils les utilisent en disant : *"Nous sommes légalement, éthiquement et socialement tenus de rendre écologiques nos produits, applications, services et pratiques informatiques."* Cela inclut toute l'informatique, à l'intérieur et à l'extérieur des centres de données, c'est ce que nous appelons l'informatique verte [120].

Ainsi, le *CC* a été pris d'assaut et plusieurs initiatives vertes ont été lancées avec succès par des particuliers, des scientifiques et des institutions nationales et internationales. La *Cop26* qui a eu lieu en novembre 2021 à Glasgow dans laquelle 25000 délégués de 200 pays étaient présents et environ 120 chefs d'état, parmi lesquels le président américain *Joe Biden*, la chancelière allemande *Angela Merkel*, et le président français *Emmanuel Macron*, ce qui reflète l'importance d'une

4. https://joint-research-centre.ec.europa.eu/jrc-news/eu-2021-wildfire-season-was-second-worst-record-find
en
5. https://www.un.org/youthenvoy/environment-climate-change/
6. https://www.businessinsider.fr/us/

telle conférence. Le *Pacte de Glasgow pour le climat* qui est le résultat des délibérations des partenaires internationaux, proclame l'importance de l'efficacité énergétique et encourage d'autres initiatives telles que des normes de régulation de la production et de la consommation d'énergie, en investissant plus de 20 milliards de dollars en faveur d'une transition de la production de l'électricité du charbon à l'énergie propre [7].

L'objectif de toutes ces initiatives se base sur deux points : **(*i*)** maximiser l'efficacité énergétique (EE) de tous les secteurs de notre vie, et **(*ii*)** réduire la consommation et développer des sources alternatives d'énergie propre. L'EE désigne l'obtention du même niveau de services tout en consommant moins d'énergie. *Steven Chu*, lauréat du prix Nobel et ancien secrétaire américain de l'énergie de l'administration d'*Obama*, a promu les initiatives EE, car elles sont plus simples que le deuxième objectif et à la portée des jeunes. Par exemple, lors des dernières élections françaises, où plusieurs grandes villes (par exemple Lyon et Poitiers) ont été conquises par le parti vert, montre l'engagement individuel contre le CC. Très récemment, *Xi Jinping*, le dirigeant chinois, s'est engagé à accélérer les réductions d'émissions dans la nation la plus polluante du monde et à atteindre la neutralité carbone d'ici 2060 [8]. Ainsi, l'indispensabilité de l'énergie, et son apport à la qualité de vie des êtres vivants en général et de l'humanité en particulier grâce à son énorme contribution au développement socio-économique, rendent son économie une nécessite vitale. Néanmoins, la production et la consommation de l'énergie engendrent des émissions importantes du gaz à effet de serre, dans laquelle, le secteur de l'énergie est responsable d'un quart des émissions mondiales. Le charbon est le principal contributeur au CC consécutif à l'activité humaine. Ces gaz sont responsables du dérèglement climatique à cause de leur augmentation dans l'atmosphère. Cette dernière est due aux activités humaines comme la surconsommation énergétique pour satisfaire les besoins industriels sans cesse, ce qui a conduit au changement climatique et à l'avènement de nombreuses maladies.

En général, l'efficacité énergétique a retenu l'attention de plusieurs secteurs stratégiques tels que l'agriculture, le bâtiment, qui a connu d'énormes progrès, où l'intégration de l'énergie est prise en compte dès la conception et plusieurs réglementations sont mises en place pour encourager l'utilisation des matériaux et des équipements plus performants. Le secteur du transport a également consacré des efforts afin d'améliorer l'EE avec l'émergence des voitures électriques. Pour les services informatiques, l'explosion étonnante des technologies modernes, comme l'internet des objets (IoT), a poussé le monde universitaire et l'industrie à développer des approches basées sur les données pour améliorer l'EE des secteurs mentionnés et les qualifier d'écologiques surtout dans les centres de données. Ces derniers sont coupables d'une consommation énergétique excessive qui ne cesse pas de croître et ils tendent à devenir l'un des plus gros consommateurs d'énergie au monde avec un ratio qui s'élèvera de 3% en 2017 à 4,5% en 2025 selon les études dans [105].

Cette consommation peut être répartie de la manière suivante : (i) l'utilisation d'énergie par les équipements informatiques tels que les serveurs, les réseaux, le stockage, etc (ii) l'énergie consommée par les installations d'infrastructure telles que les systèmes de refroidissement (pour un kilowatt dépensé par un serveur, un autre kilowatt serait nécessaire pour le refroidir [175]), et (iii) le fonctionnement des équipements sans arrêt [42]. La réparation de la consommation énergétique des équipements informatiques dans les centres de données est illustrée dans la figure 1.2.

Pour faire face à cette augmentation considérable de la consommation de l'énergie, la communauté scientifique et les industriels ont pris des initiatives couvrant plusieurs actions, à savoir :

7. https://ukcop26.org/wp-content/uploads/2022/03/FR-COP26-Presidency-Outcomes-The-Climate-Pact.pdf
8. https://www.nytimes.com/2020/09/23/world/asia/china-climate-change.html

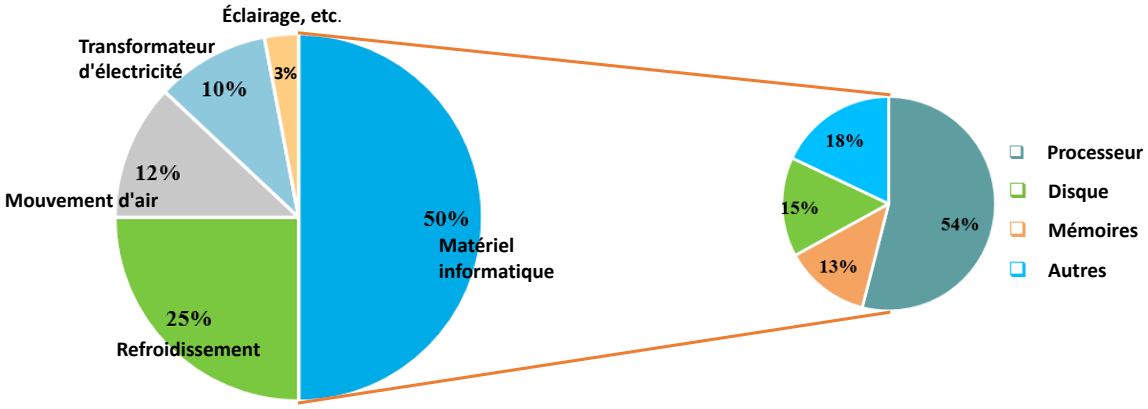


FIGURE 1.2 – La réparation de la consommation énergétique des équipements informatiques dans les centres de données [154]

(*a*) concevoir du matériel en réduisant sa consommation énergétique (*b*) re-développer des logiciels en minimisant la consommation, (*c*) exploiter les divers états d'alimentation offerts dans les nouveaux matériaux, et (*d*) améliorer la conception, l'utilisation et le contrôle de l'infrastructure des centres de données [89].

En particulier, les améliorations apportées par les fabricants, concernant les composants et le matériel des TIC, sont considérables et les limites d'amélioration sont presque atteintes. En fait, les composants physiques d'un appareil consomment de l'énergie, cependant, l'utilisation de logiciels implique des opérations sur le matériel qui nécessitent une consommation d'énergie. Par conséquent, le logiciel est aussi indirectement une source de consommation d'énergie. Ainsi, de plus en plus de recherches sont réalisées pour économiser l'énergie d'un point de vue logiciel.

La communauté des bases de données ne peut pas rester inactive et elle est appelée à produire des solutions vertes et obtenir son label écologique qui contribue à la réalisation d'une économie circulaire verte dans d'autres secteurs. Cette intégration est une priorité stratégique. Pour intégrer l'EE dans le contexte des bases de données, des études et des analyses doivent être effectuées.

La base de données (BD) est un domaine qui a atteint son stade de maturité, très varié et il n'a pas cessé d'évoluer au fil des années comme le montre la figure 1.3. Cette variété concerne tout le cycle de vie d'une BD, dès sa conception et jusqu'à son déploiement. Pour concevoir une BD, il existe plusieurs formalismes (Merise, UML, etc,.). Depuis l'apparition des bases de données vers la fin des années 1960, de différents modèles ont été introduits, en commençant par le modèle hiérarchique et en réseau, en passant par le modèle relationnel qui s'est largement imposé à partir de son apparition dans les années 70 et sa domination sur le marché dans les années 90. Les modèles introduits après le modèle relationnel n'ont pas connu le même succès tel que le modèle orienté objet et le modèle multidimensionnel pour les entrepôts des données, ce qui les a obligé à s'adapter et de proposer des modèles à mi-chemin entre leur conception initiale et le modèle relationnel ce qui a donné naissance à des modèles hybrides comme le ROLAP et l'objet-relationnel. Même les bases de données NoSQL qui ont attiré l'attention des utilisateurs qui cherchent une capacité de stockage énorme tout en ayant une flexibilité au niveau du type et des structures de données et qui souhaitent assouplir la rigidité du modèle relationnel, causée par ses propriétés ACID et les dépendances fonctionnelles, ont proposé des langages (SQL-Like) qui ressemblent beaucoup syntaxiquement au langage SQL comme le CQL (Cassandra Query Language de Cassandra) afin de profiter de la popularité de ce dernier et dans le but de vulgariser ces nouvelles technologies.

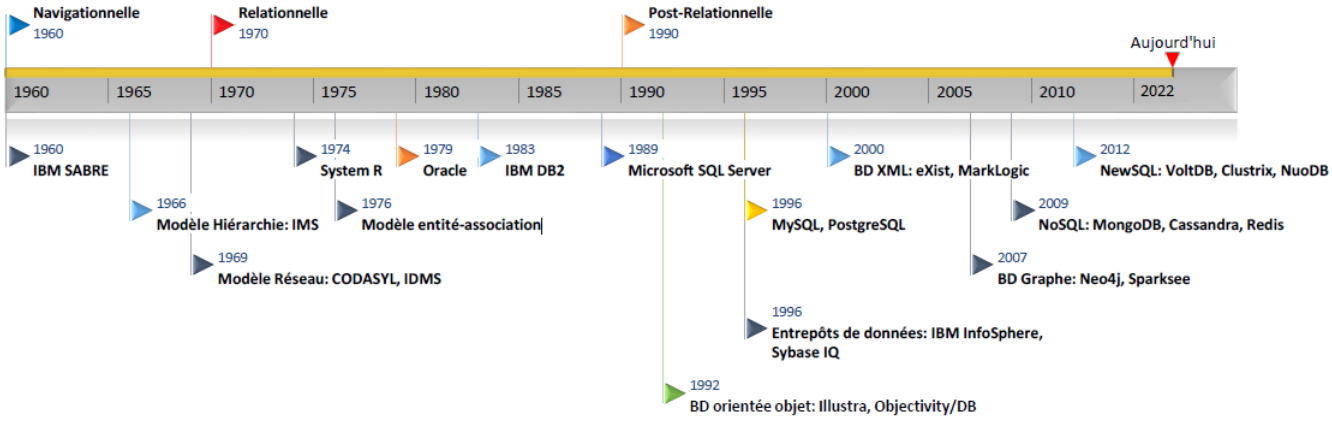


FIGURE 1.3 – L'évolution des bases de données

Au fil des années et grâce au progrès technologique, les exigences des utilisateurs des *BD* se sont multipliées et variées, par conséquent, la performance n'est plus le seul besoin non-fonctionnel exigé par les utilisateurs, et avec le réchauffement climatique plusieurs voix se sont levées afin de prendre en considération l'aspect énergétique et d'intégrer la consommation énergétique dans les bases de données en tant que besoin non-fonctionnel, voire une contrainte à satisfaire.

Le marché des systèmes de gestion de bases de données (*SGBD*) est ultra-concurrentiel, contenant presque 400 produits qui sont disponibles actuellement, où chaque système essai d'augmenter sa part du marché à travers la proposition de nouvelles fonctionnalités. Cette grande variété des SGBD a permis de les classer en familles en se basant sur le modèle de la base de données qu'ils hébergent, où nous trouvons les SGBD relationnels, orientés objet, spatial, les SGBD NoSQL qui eux aussi sont divisés en sous-familles (orienté colonne, clé-valeur, document et graphe), et puis nous trouvons les SGBD multi-modèle qui permettent de traiter plusieurs bases de données ayant des modèles différents.

Dans toutes les générations des bases de données, les techniques d'optimisation ont été présentes afin d'améliorer les performances. Il existe un large éventail de techniques utilisées, nous citons par exemple les indexes, les vues matérialisées, la fragmentation et le traitement parallèle. Chaque technique est adaptée à des conditions particulières qui dépendent du type de requêtes, les tailles des tables, le matériel utilisé et la plate-forme de déploiement.

Les bases de données dépendent également de la configuration matérielle qui est responsable du stockage des données et qui est indispensable pour l'exécution des requêtes. La configuration matérielle a une influence sur la qualité de la base de données, elle permet de la mise à l'échelle et son évolution, et vérifier la possibilité d'utiliser une technique d'optimisation ou pas (le cas du traitement parallèle). Au niveau du stockage, deux principaux types sont utilisés, les disques durs mécaniques et les disques du type SSD, et pour le traitement des requêtes la majorité des serveurs des données utilisent des CPU mono ou bien multi-cœur, ou bien des GPU qui ont prouvé leur efficacité.

La base de données est une communauté bien établie, très active avec des théories solides (l'algèbre relationnelle par exemple), et vulgarisée dans le monde académique, grâce aux cours de *BD* enseignés dans les différentes disciplines dans les universités, et même au lycée. Cependant, intégrer l'aspect énergétique dans les *BD* est loin d'être une tâche facile, à cause de (*i*) la complexité des systèmes de gestion des bases de données (matériel, logiciel, réseau, etc), (*ii*) la présence de plusieurs couches dans la base de données pour intégrer l'aspect énergétique comme l'ETL, la conception physique, le traitement des requêtes, la maintenance des données, et (*iii*) la dominance de la culture de la performance dans ce domaine, où les concepteurs et les

utilisateurs se soucient uniquement d'améliorer le temps d'exécution des requêtes et la capacité du stockage, ce qui rend le changement d'une telle mentalité un vrai challenge. La demande constante du traitement et de l'analyse des données qui n'ont guère cesser de croître et la présence des systèmes de données open source présentent des avantages et une source de motivation pour étudier l'intégration de la consommation énergétique dans les bases de données.

En partant de la discussion ci-dessus, dans laquelle nous avons abordé l'importance de la dimension énergétique dans les systèmes informatiques et nous avons examiné la variété des bases de données, nous nous focalisons, en ce qui suit, sur l'association de ces deux thématiques, afin de traiter le problème de la réduction de la consommation de l'énergie dans les bases de données, qui présente le contexte de notre thèse.

Les systèmes de traitement de données (STD) en général et les SGBD en particulier sont l'un des principaux consommateurs d'énergie du fait qu'ils sont en charge de traiter de nombreuses opérations de jointure, filtrage, agrégation, et tri. Traditionnellement, les STD sont conçus pour optimiser les accès aux données, et ils mettent à l'écart la consommation énergétique. Cette consommation excessive d'énergie a attiré l'attention des chercheurs du monde académique et du monde industriel. Goetz Graefe [63] de HP et Stavros Harizopoulos [71] de MIT, ont été les pionniers à s'intéresser à la réduction de l'énergie dans les BD et ils ont publié deux articles de vision. Après ces derniers, plusieurs travaux de recherche ont été menés, en abordant à la fois le coté logiciel et le coté matériel, qui est plus exploré que le premier.

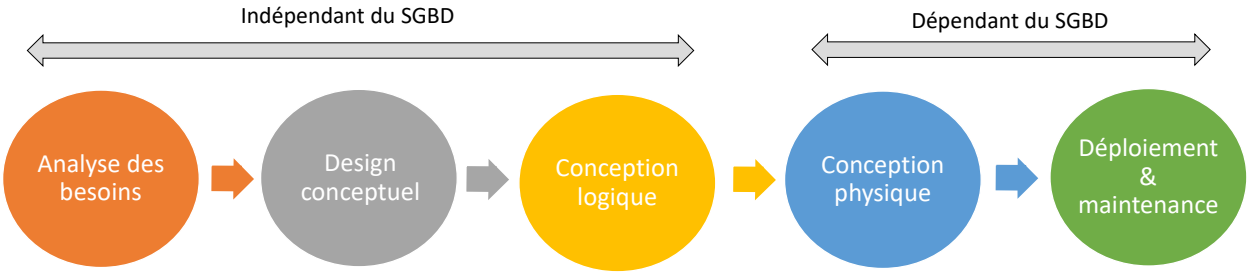


FIGURE 1.4 – La dépendance des étapes du cycle de vie de la base de données du SGBD

Après une revue de la littérature, nous avons constaté que les travaux existants se déclinent sous deux catégories : (1) les solutions orientées logicielles proposant des modèles analytiques pour estimer la consommation énergétique lors de l'exécution de requêtes. Ces modèles sont par la suite intégrés dans l'optimiseur de requêtes. (2) Les solutions orientées matérielles qui favorisent l'utilisation de matériels moins énergivores. En analysant ces travaux, nous avons identifié deux principales limites : (a) La quasi-totalité de ces travaux s'intéresse à la caractérisation de la consommation énergétique des requêtes en supposant que le STD et ses optimisations sont une boîte noire. Les différentes estimations des coûts de requêtes (entrées-sorties, CPU) établies par le STD sont extraites et utilisées par les modèles de coût analytiques dédiés à l'estimation de la consommation énergétique. Ces travaux dépendent généralement du SGBD hébergeant la BD comme le montre la figure 1.4, et ils ne prennent pas en considération la présence des structures d'optimisation et ils estiment que le schéma de la BD reste figé et n'évolue pas. Cette vision de boîte noire doit être substituée par une vision boîte blanche permettant aux concepteurs de sélectionner leurs propres optimisations et structures d'optimisation tels que les indexes pour optimiser la consommation énergétique lors de l'exécution de requêtes. (b) Les travaux existants ne sont pas focalisés sur d'autres phases de cycle de vie de conception d'une base de données telles que la phase logique. Cette dernière, qui est indépendante du SGBD cible et dans laquelle

le schéma de la base de données est défini, impacte fortement la phase physique et peut être elle aussi sensible à l'énergie.

Le traitement de l'énergie dans les *BD* est étudié au sein de notre laboratoire LIAS (Laboratoire d'Informatique et d'Automatique pour les Systèmes), et deux thèses ont été soutenues portant sur cette thématique. Ces dernières, ont traité le problème du traitement énergétique des requêtes en proposant des modèles de coût énergétiques, en utilisant des différents algorithmes de machine learning. Le traitement des requêtes dans les *SGBD*s multi-cœur et la sélection éco-énergétique des vues matérialisées sont également étudiés dans ces thèses.

La vision de notre thèse est de consolider les travaux existants dans la littérature et ceux effectués au sein du laboratoire en explorant d'autres pistes afin de surmonter leurs limites. Parmi les pistes explorées par cette thèse, (1) la sélection verte des indexes, et (2) l'étude de l'impact de la phase logique sur la consommation d'énergie. Malgré la grande importance des indexes et leur efficacité dans toutes les générations des *BD*, qui leur doivent leur succès comme ils représentent la composante principale lors de l'optimisation, aucun travail de recherche n'est consacré à l'étude de son comportement énergétique. D'autre part, la conception logique était complètement ignorée par les travaux sensibles à l'énergie.

Face à la complexité de notre sujet de recherche et la grande variété qui se présente à tous les niveaux d'une *BD*, le fait de se positionner afin d'intégrer la dimension énergétique est très important. Ainsi, plusieurs questions se posent et nécessitent une réponse :

- Le cycle de vie d'une base de données est composé de plusieurs étapes, et chacune présente une opportunité pour l'exploiter et essayer d'intégrer l'énergie. Quelles sont les étapes du cycle de vie qui seront les plus efficaces pour intégrer l'énergie ?
- Est-ce que la conception logique a un impact direct sur la consommation d'énergie d'une base de données, ou bien seule la couche physique qui nécessite d'être optimisée ?
- Est-ce que les structures d'optimisation ont la même amélioration de l'efficacité énergétique que celle qu'elles apportent sur la performance ?
- Plusieurs pratiques sont effectuées par la communauté des bases de données pour l'optimisation de la performance, est ce que ces mêmes pratiques restent valides pour réduire l'énergie ?
- La dé-normalisation est souvent considérée comme une technique d'optimisation, est-ce qu'elle a le même effet sur la consommation énergétique ?
- Les indexes en particuliers, ont montré leur efficacité dans l'amélioration des performances dans toutes les générations des BD. Serait-il intéressant de les étudier d'un point de vue énergétique ? Quel type d'indexes nous devons étudier ? Est-ce qu'ils vont garder leur même efficacité ? Quels sont les paramètres des indexes sensibles à l'énergie ?
- Il existe une multitude d'algorithmes de sélection d'indexes, est ce que nous avons besoin d'un nouvel algorithme ?

1.2 Objectifs et contributions

L'objectif principal de cette thèse est de sensibiliser les utilisateurs et les concepteurs de bases de données de l'importance de la dimension énergétique, en proposant des solutions dès la conception de la base de données jusqu'à sa mise en production. Nous visons à optimiser la consommation énergétique des bases de données d'un point de vue logiciel. Nous proposons également une vision de boîte blanche qui offre aux concepteurs la possibilité de sélectionner leurs propres structures d'optimisation réduisant la consommation énergétique. L'intégration de l'aspect énergétique dans ce contexte est un problème important, qui nécessite la compréhension du comportement des systèmes de traitement de données et l'identification des différents paramètres de la *BD* susceptibles d'être sensibles à l'énergie. Dans cette thèse, nos objectifs

consistent à (1) étudier le comportement énergétique des bases de données afin de découvrir les points d'intégration possibles de la dimension énergétique (2) identifier les paramètres potentiels qui peuvent influencer la consommation d'énergie de ces systèmes (3) intégrer l'énergie dans les différentes étapes du cycle de vie dans les bases de données.

Pour répondre aux questions posées dans la section précédente et pour réaliser les objectifs fixés, nos principales actions dans cette thèse, portent sur les points suivants :

1. **Identifier les étapes du cycle de vie pour intégrer l'énergie**
   Le cycle de vie d'une base de données est composé de plusieurs étapes successives et interdépendantes, ainsi, nous avons essayé d'identifier les étapes dans lesquelles il est possible d'intégrer l'énergie d'un point de vue logiciel, qui peuvent influencer les étapes suivantes et qui permettront d'améliorer l'efficacité énergétique une fois la base de données est déployée.
2. **L'étude de la consommation énergétique dans les entrepôt de données relationnels**
   Dans cette thèse, nous visons à réaliser une revue de la littérature des approches ayant traité la dimension énergétique dans les systèmes d'informations en général et dans les bases de données relationnelles en particulier. Une compréhension approfondie des approches proposée nous permettra de mieux aborder ce problème et nous donnera des pistes à explorer. Plusieurs travaux de recherche sont explorés ce qui nous a permis de mieux se positionner par rapport au problème.
3. **Identifier les paramètres sensibles à l'énergie**
   Avant d'intégrer l'énergie dans le problème de sélection des structures d'optimisation lors de la conception physique, il est primordial de choisir la meilleure structure d'optimisation et qu'elle soit adéquate aux besoins de la base de données cible. Il est également nécessaire de comprendre le comportement énergétique de cette structure. Pour remédier à ça, nous avons réalisé un audit des indexes de jointure binaires pour identifier les paramètres sensibles à l'énergie et de déterminer le cas idéal de leur utilisation afin de maximiser le profit.

Dans le but de réaliser nos objectifs, nos contributions portent sur les points suivants : (*i*) la réduction de l'énergie lors de la conception physique en utilisant la sélection éco-énergétique d'une structure d'optimisation, (*ii*) la proposition d'un nouvel algorithme de sélection des indexes, et (*iii*) l'intégration de la dimension énergétique dans la conception logique.

— **La conception physique éco-énergétique :**
nous avons proposé une approche multi-objectif de sélection d'une structure d'optimisation en considérant le cas des indexes de jointure binaires en intégrant l'aspect énergétique et en considérant deux besoins non-fonctionnels : la performance et la consommation énergétique. Nous avons proposé une nouvelle formalisation du problème de sélection des indexes de jointure binaires, à travers l'ajout de la consommation de l'énergie ce qui lui rend un problème multi-objectif.

— **Un nouvel algorithme de sélection d'indexes :**
les indexes de jointure binaires nécessite une couche supplémentaire en les comparant aux autres types d'indexes. Ils ont besoins d'un pré-traitement afin de sélectionner les attributs indexables. Ainsi, nous avons proposé un nouvel algorithme pour la réduction de l'énergie consommée qui prend en considération cette couche et qui utilise des paramètres sensibles à l'énergie. Cet algorithme se base sur une modélisation d'hypergraphe afin de représenter l'espace de recherche qui peut être très grand et implique des interactions entre les requêtes.

— **La conception logique éco-énergétique :**
l'intégration de l'énergie dans les bases de données peut se faire à plusieurs niveaux en suivant son cycle de vie. Dans cette thèse, nous avons essayé d'intégrer l'énergie à

un niveau supérieur du cycle de vie d'une base de données, plus précisément dans la phase logique. Le schéma logique est caractérisé par sa variété produite par les attributs des tables et les relations existantes entre elles. Nous avons proposé une approche qui permet de générer les schémas logiques possibles à partir d'un schéma initial et d'élaguer l'espace de recherche afin de réduire la complexité du problème. Cette approche permet à la fin de retourner le schéma logique le plus écologique.

1.3 Organisation du manuscrit

Ce rapport de thèse est composé de quatre chapitres, outre l'introduction générale et la conclusion générale. Les quatre chapitres sont divisés en deux parties composées de deux chapitres chacune. La première partie est dédiée à l'étude de l'état de l'art, quant à la deuxième, elle présente les deux contributions majeures autour desquelles s'articule cette thèse comme illustré dans la figure 1.5.

— Dans le deuxième chapitre, nous présentons les concepts de bases liés à notre domaine de recherche et qui sont nécessaires pour l'assimilation du reste de la thèse. En particulier, ce chapitre est consacré à la technologie des bases de données, en détaillant son cycle de vie, les différentes techniques d'optimisation employées dans la littérature avec une focalisation particulière sur les structures d'optimisation.
— Le troisième chapitre, présente une revue de l'état de l'art des différentes solutions et techniques proposées pour la réduction de la consommation énergétique dans les systèmes informatiques en général, et dans les bases de données en particulier. Les différents concepts de base et les outils utilisés pour mesurer l'énergie sont également présentés.
— Le quatrième chapitre décrit l'approche G-BJI qui permet la sélection éco-énergétique d'une structure optimisation intitulée les indexes de jointure binaires, en détaillant ses étapes et les résultats de l'étude expérimentale réalisée.
— Le cinquième chapitre fournit une description détaillée de l'approche proposée, nommée LS-Energy permettant de sélectionner le schéma logique le plus écologique. L'algorithme développé, l'étude expérimentale et l'analyse des résultats obtenus sont fournies.
— Enfin, le sixième chapitre, résume les résultats obtenus des contributions de la thèse, présente les conclusions générales de ce travail et esquisse diverses perspectives.

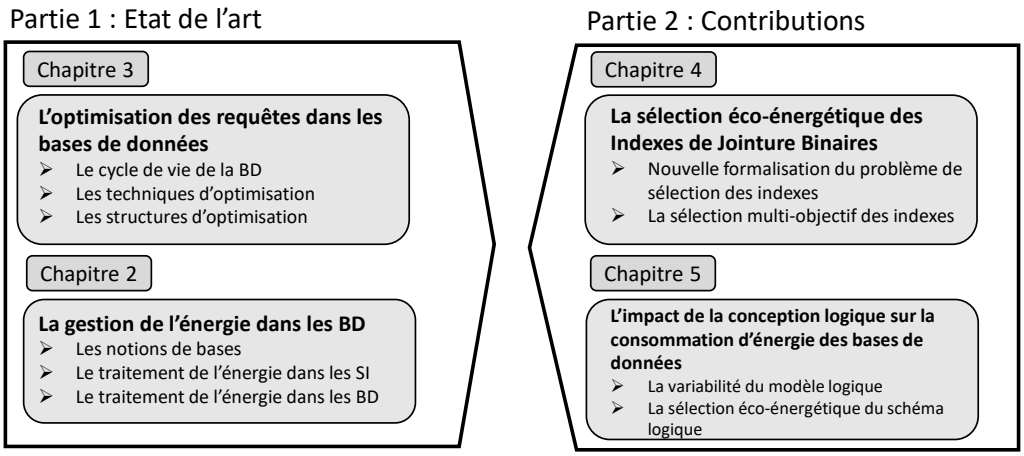


FIGURE 1.5 – La répartition des chapitres de la thèse

Cette thèse comprend également deux annexes. L'Annexe A liste les requêtes de la charge de travail utilisées pour le benchmark SSB durant les expérimentations réalisées dans le chapitre 4, quant à l'Annexe B, il décrit les requêtes de la charge de travail utilisée pour le benchmark TPC-H lors de l'étude expérimentale menée dans le chapitre 5.

Les principales contributions de notre thèse sont réparties dans le chapitre 4 et le chapitre 5. Les contributions ont donné lieu à des publications scientifiques dans des conférences internationales.

1. [60] Issam Ghabri, Ladjel Bellatreche, Sadok Ben Yahia : Selection of a Green Logical Data Warehouse Schema by Anti-monotonicity Constraint : Theory and Practice of Computer Science - 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus.
2. [61] Issam Ghabri, Ladjel Bellatreche, Sadok Ben Yahia : Energy Efficiency vs. Performance of Analytical Queries : The case of Bitmap Join Indexes : IEEE International Conference on Big Data (IEEE Big Data), 2021, Orlando, FL, USA.
3. [62] Issam Ghabry, Sadok Ben Yahia, Mohamed Nidhal Jelassi : Selection of Bitmap Join Index : Approach Based on Minimal Transversals : Big Data Analytics and Knowledge Discovery - 20th International Conference, (DaWaK 2018), 2018 Regensburg, Germany.

Deuxième partie

État de l'Art

Chapitre 2

L'optimisation des requêtes dans le bases de données

2.1 Introduction

Depuis leur apparition dans les années 60, les bases de données ont constitué une discipline fondamentale dans la science de l'informatique, elles sont omniprésentes dans de nombreuses applications et enseignées dans pratiquement toutes les filières universitaires. Les bases de données permettent de stocker et d'interroger toute information utile afin qu'elle soit efficacement exploitée par différents utilisateurs cibles. Elles sont le socle des systèmes d'information des entreprises, plus particulièrement dans les tâches de gestion.

L'explosion des technologies numériques a incité les organisations et les individus à augmenter leur intérêt à la collection, au stockage et à l'analyse des données, d'où le nombre et les types d'applications autour des bases de données se sont ainsi multipliés, et par conséquent le nombre de systèmes de gestion de bases de données à augmenter proportionnellement. Un SGBD est un logiciel qui permet d'accéder aux données contenues dans une base de données son objectif est de fournir une méthode pratique et efficace pour définir, stocker et récupérer les données contenues dans la base de données. Selon le rapport de *Statista* [1], les revenus des SGBD ont atteint 65 milliards de dollars en 2020, ce qui reflète l'importance des bases de données et son industrie. Une multitude de SGBD commerciaux et open source sont proposés par les industriels et les scientifiques. selon *DB-Engine* [2], plus que 390 SGBD sont disponibles, et 7 du top 10 sont des SGBD relationnels ce qui motive notre intérêt aux entrepôt de données relationnels.

Le succès des bases de données ne les a pas empêchées de s'étendre, elles n'ont guère cessé d'évoluer pour s'adapter aux nouveaux besoins des utilisateurs. Cette évolution est traduite par la proposition de nouveaux modèles de données (modèle hiérarchique, relationnel, etc.), de supports de stockage utilisés (HDD, flash, etc.), d'architecture déployées (centralisée, parallèle, Cloud, etc.), des techniques d'optimisations exploitées (index, vues matérialisées, parallélisation), et des langages de requêtes (SQL, SPARQL, XQuery).

Dans ce chapitre, nous présentons une revue de l'état de l'art de la technologie des bases de données relationnelles en général, en détaillant les différentes étapes de son cycle de vie de conception et d'exploitation. Nous nous focalisons particulièrement sur les techniques d'optimisation utilisées et le traitement des requêtes notamment dans les entrepôts de données relationnels qui représente notre cas d'étude dans cette thèse. Nous allons détailler les différentes structures d'optimisation et les travaux proposés pour chacune d'entre elles.

2.2 Le cycle de vie d'une base de données

Une base de données est un ensemble structuré de données enregistrées sur des supports accessibles par un ordinateur, représentant des informations du monde réel et pouvant être interrogées et mises à jour par un ensemble d'utilisateurs. Plus précisément, une base de données est une collection organisée d'informations connexes stockées avec un minimum de redondance, de manière à les rendre accessibles pour de multiples applications. Son objectif général est de rendre l'accès à l'information facile, rapide, peu coûteux et flexible pour l'utilisateur. La création d'une base de données suit un schéma itératif appelé cycle de vie, qui est un processus continu de création, de maintenance et d'amélioration de la BD comme le montre la figure 2.1.

La tâche essentielle d'un concepteur de bases de données est de concevoir leurs schémas. La conception d'une *BD* est la définition d'un modèle de données détaillé qui contient tous les choix de conception logique et physique nécessaires et les paramètres de stockage physique nécessaires pour générer le langage de définition de données, relatif au SGBD cible, qui sera utilisé ensuite pour créer une base de données. La bonne conception d'une BD facilite la gestion des données et génère des informations précises, par contre, une mauvaise conception peut

1. https://www.statista.com/statistics/724611/worldwide-database-market/
2. https://db-engines.com/en/ranking

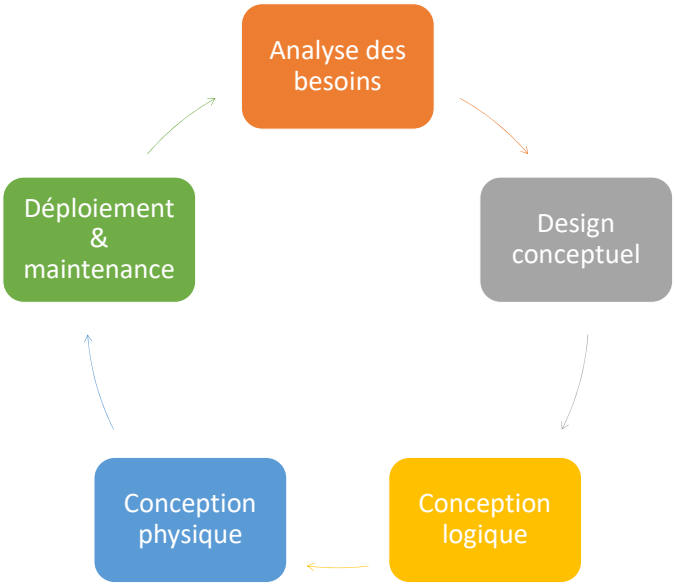


FIGURE 2.1 – Le cycle de vie d'une base données

engendrer des erreurs difficiles à tracer et une maintenance très coûteuse. La conception d'une base de données est tout simplement l'étape la plus importante durant le cycle de vie d'une BD qui a comme objectif principal de créer des modèles conceptuels, logiques et physiques complets, normalisés, non-redondants et entièrement intégrés [151].

En ce qui suit, nous détaillons les cinq étapes du cycle de vie de la BD. Notons que chacune de ces étapes est suivie par un processus itératif de test, d'évaluation et de modification jusqu'à ce que l'objectif soit atteint.

2.2.1 L'analyse des besoins

L'analyse des besoins, comme son nom l'indique, permet d'analyser la situation de l'entreprise afin d'identifier les problèmes et les contraintes, fixer les objectifs ce qui permet de définir la portée de la BD et ses limites par rapport à l'entreprise. Cette étape permet de collecter les besoins fournis par les futurs utilisateurs sous la forme d'un ensemble de besoins informels, inconsistants. Ensuite, il faut formaliser ces besoins et identifier leurs caractéristiques attendues. Plus précisément, les besoins et les exigences des utilisateurs sont traduits en un ensemble de besoins qui définissent ce que le système doit faire et comment il doit le faire. Le concepteur de la BD et les utilisateurs finaux doivent travailler ensemble pour s'assurer que les besoins sont compréhensibles, sans ambiguïté, complets et concis. La phase d'analyse du cycle de vie est, en fait, une vérification approfondie des besoins des utilisateurs. Le résultat de cette étape est une spécification textuelle en langage naturel de ces besoins.

Une bonne analyse des besoins est fondamentale pour réussir la conception de la BD. Généralement, les besoins définis se répartissent en deux types : (i) besoins fonctionnels et (ii) besoins non-fonctionnels tels qu'illustré dans la figure 2.2.

Les besoins fonctionnels (BF) sont tout besoin qui indique ce que le système doit réaliser. Autrement dit, un BF décrit le comportement du système lorsque certaines conditions sont valides. Parmi les BFs dans une base de données, nous pouvons citer : l'authentification, la gestion de transactions, le niveau d'autorisation, historier les données, etc.

Les besoins non-fonctionnels (BnF) sont définis comme tout besoin qui spécifie comment le système doit se comporter et quelles sont ses limites. Les BnFs spécifient généralement les caractéristiques de qualité du système. Les BnFs ne décrivent pas les fonctionnalités effectuées

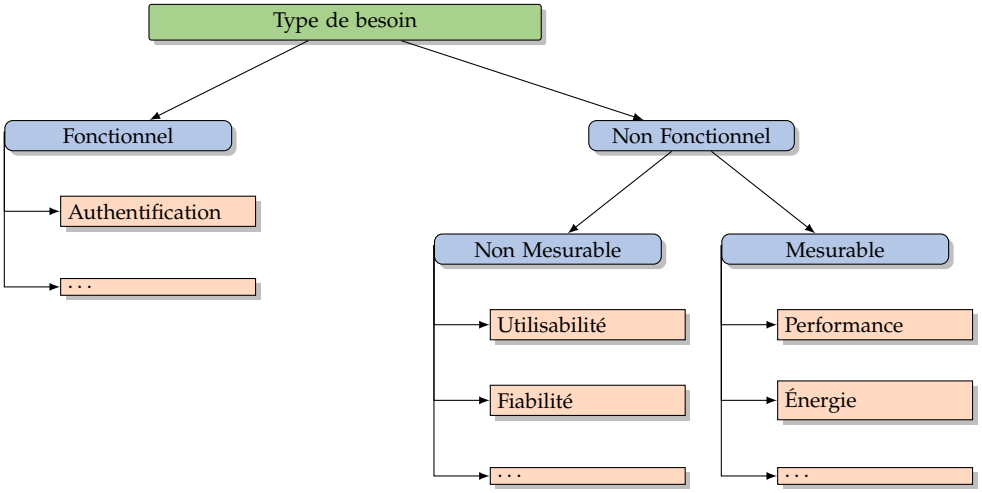


FIGURE 2.2 – Classification des besoins fonctionnels et non fonctionnels

par le système, mais plutôt comment il les remplit. Parmi les $BnFs$ dans une base de données, nous pouvons citer : la performance généralement mesurée en temps de réponse, la consommation d'énergie, l'utilisabilité, la fiabilité, la sécurité, etc. En outre, les besoins non-fonctionnels peuvent être classés en deux catégories : (1) BnF mesurable, qui sont des besoins qui peuvent être mesurés sur une échelle métrique définie par l'utilisateur comme la performance et la consommation d'énergie, et (2) BnF non-mesurable qui sont les besoins qui ne peuvent pas être décrits que qualitativement à l'aide d'une échelle ordinaire ou nominale comme la fiabilité et l'utilisabilité..

2.2.2 Le design conceptuel

Le design conceptuel consiste à créer un schéma conceptuel pour la base de données, à partir des besoins et des exigences déjà collectés et analysés dans l'étape précédente, en utilisant un modèle de données conceptuel de haut niveau, tel que le modèle entité-association ou bien le digramme de classe UML.

Ce schéma doit représenter les objets du monde réel de la manière la plus réaliste possible. Le modèle conceptuel doit s'assurer d'une bonne et claire compréhension des BF et BnF de l'entreprise. À ce niveau d'abstraction, le SGBD et/ou le type de matériel à utiliser pourrait ne pas avoir été identifié. Par conséquent, la conception doit être indépendante du logiciel et du matériel afin que le système puisse être installé dans n'importe quelle plate-forme matérielle et logicielle choisie ultérieurement [153].

2.2.3 La conception logique

La conception logique commence par le choix du du modèle de données choisi dans l'étape précédente. Ensuite, le schéma conceptuel sera transformé en un modèle de données afin de sa mise en œuvre à travers un SGBD. La génération du schéma logique peut être réalisée d'une manière automatisée via les règles de transformation dont le modèle entité-association ou le diagramme de classe UML est transformé en un schéma relationnel composé d'un ensemble de tables dans le cas des bases de données relationnelles. La conception logique doit contenir que des tables correctement normalisées (1FN, 2FN, 3FN, FNBC).

Après la définition des clés primaires et étrangères, et les domaines des colonnes, la normalisation est la dernière partie de la conception logique. Son objectif est d'éliminer la redondance et les anomalies potentielles lors de la mise à jour de la BD. L'anomalie de mise à jour est une

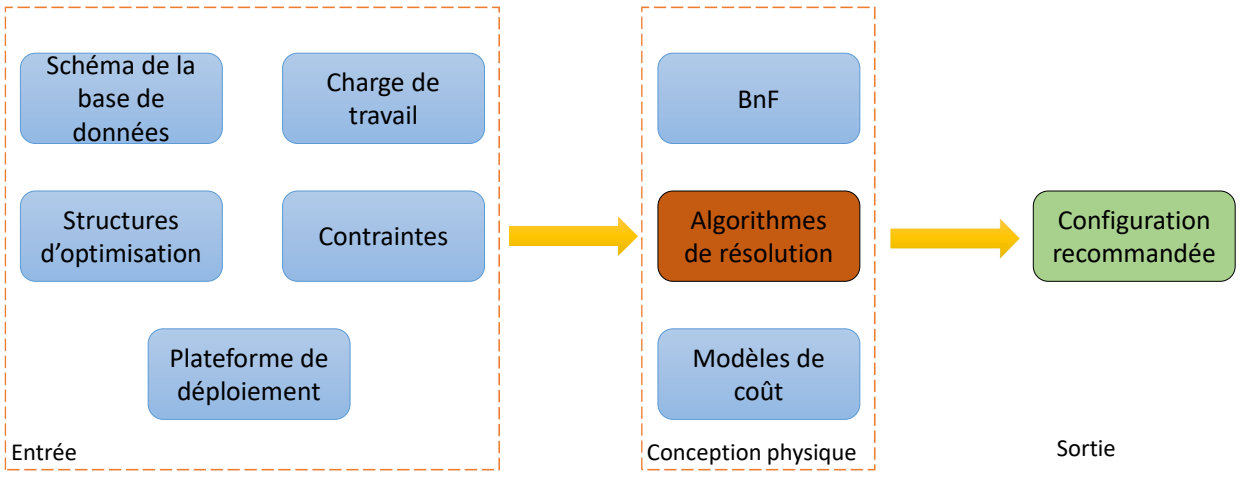


FIGURE 2.3 – La sélection d'une structure d'optimisation

conséquence de la redondance. Si une donnée est enregistrée en plusieurs endroits, cette même données doit être mise à jour en plusieurs endroits afin de s'assurer de la cohérence de la BD. La normalisation est une technique permettant de modifier le schéma relationnel afin de diminuer la redondance.

2.2.4 La conception physique

La conception physique consiste à définir la configuration physique des données, de leurs types, et surtout la sélection des techniques d'optimisation sur le support de stockage. Au cours de cette phase, les structures de stockage internes, les organisations de fichiers, les structures d'optimisation (les vues matérialisées, les indexes, la fragmentation de données.), les chemins d'accès, les contraintes d'intégrité, les droits d'accès des utilisateurs sont spécifiés.

Dans cette étape, les structures d'optimisation (SO) sont sélectionnées pour optimiser un ou plusieurs BnF tels que le temps d'exécution des requêtes et l'énergie consommée. Les différentes structures d'optimisation sont détaillées dans la section suivante (Cf. section 2.5.1). Les méthodes d'accès et de stockage des tables sur le disque, qui permettent à la base de données de fonctionner efficacement, sont définies dans cette phase. L'objectif de cette étape est de maximiser la performance de la BD sur l'ensemble du système en optimisant l'exploitation des ressources physiques (le CPU, les opérations E/S, l'accès à la mémoire, etc.) qui influencent sur la performance lors de l'exécution des requêtes.

Le problème de sélection d'une structure d'optimisation donnée est formalisé comme un problème d'optimisation afin de satisfaire un ou plusieurs BnF tout en respectant l'ensemble de contraintes telles que l'espace de stockage exigé par les SO à sélectionnées, si nécessaire.

Formellement, le problème de sélection des structures d'optimisation peut être défini comme suit :

Étant donné :

— un schéma d'une base de données,
— une charge de travail composée de n requêtes : $Q = \{q_1, q_2, \ldots, q_n\}$,
— une architecture matérielle avec ses composants,
— un ensemble des structures d'optimisation $S = \{s_1, s_2, \ldots, s_l\}$,
— un ensemble de contraintes : $C = \{c_1, c_2, \ldots, c_m\}$,
— un ensemble de besoins non-fonctionnels : $B = \{b_1, b_2, \ldots, b_k\}$,

Le problème de sélection des SO consiste à sélectionner un sous-ensemble de ces structures en satisfaisant l'ensemble B et en respectant l'ensemble des contraintes C.

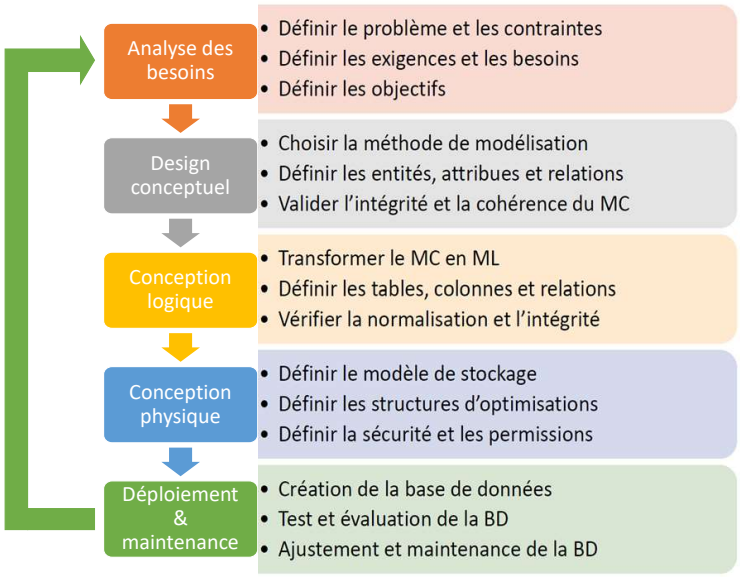


FIGURE 2.4 – Les étapes du cycle de vie d'une base données

La figure 2.3 illustre le schéma de la conception physique d'une base de données, dans laquelle la sélection des structures d'optimisation l'impact fortement et contribuent de manière positive ou négative à l'amélioration de sa performance, car elles deviennent une partie de la BD.

2.2.5 Déploiement et maintenance

À la fin de la conception logique et physique, la base de données devient prête pour qu'elle soit créée et déployée à travers la mise en œuvre des schémas précédents. La phase de déploiement consiste à choisir la plate-forme adéquate dans laquelle la base de données cible sera déployée.

Le résultat des différentes phases de conception est une série d'instructions de langage de définition de données. Ces instructions permettent la création des tables, des différentes structures d'optimisation, des contraintes de sécurité, des règles de stockage, de définir l'architecture adoptée (centralisée ou distribuée), etc. Ainsi, dans cette phase, toutes les spécifications de conception doivent être mises en œuvre.

Une fois que la base de données soit mise en production, le rôle de l'administrateur est de tester, évaluer et ajuster la BD afin de s'assurer qu'elle fonctionne comme prévu. L'administrateur doit périodiquement effectuer des activités de maintenance de la BD comme la sauvegarde, la restauration, la modification des attributs et des tables, la génération des statistiques, l'audit de sécurité, dans le but de garder son bon fonctionnement. Cette étape comporte aussi le tuning, qui est un processus de réglage continu de la BD dont le but est d'atteindre une performance maximale pour faire face à l'arrivée de nouvelles requêtes/exigences nécessitant des modifications mineures.

Le schéma illustré par la figure 2.4, représente les différentes étapes du cycle de vie de la base de données en précisant les tâches principales de chacune d'entre elles.

2.3 Le traitement des requêtes

Après les différentes étapes de conception et la création de la base de données, les utilisateurs peuvent exploiter les données à travers les requêtes. Le traitement des requêtes (TR) est un

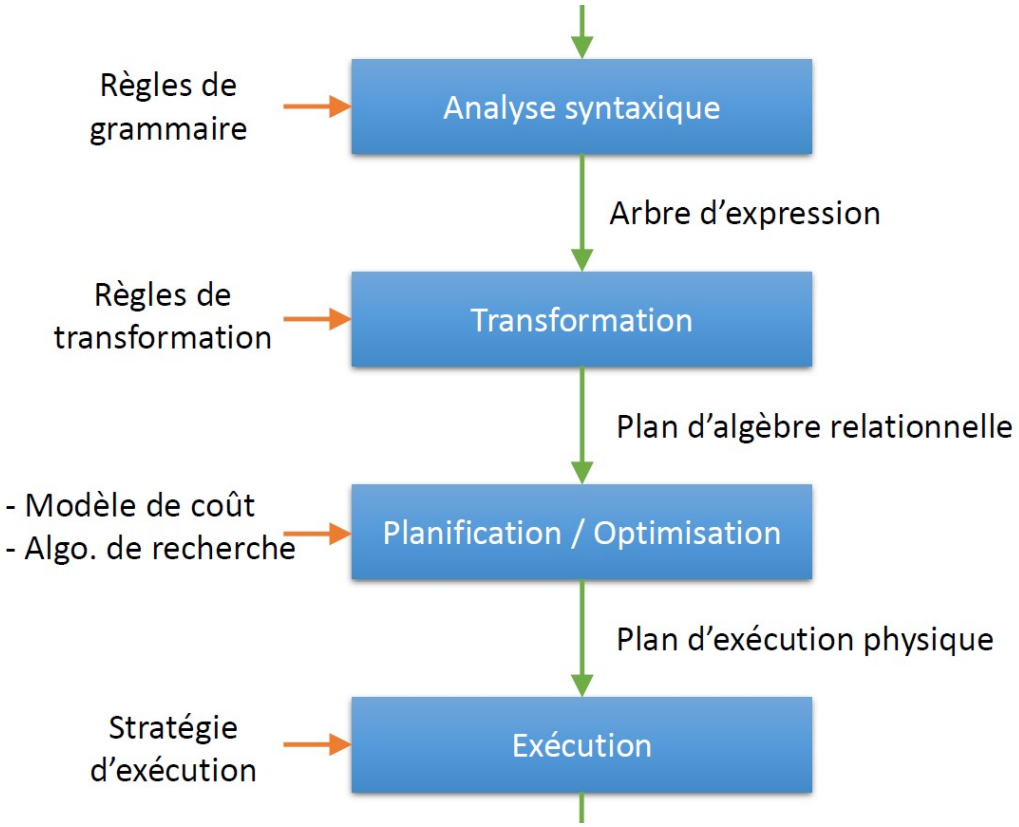


FIGURE 2.5 – Architecture globale du traitement de requêtes par un SGBD.

ensemble de composants d'un SGBD permettant de transformer les requêtes et les commandes de modification des données émises par les utilisateurs en une séquence d'opérations de base de données et de les exécuter. Dans cette section, nous abordons le traitement des requêtes, et nous considérons le cas des bases de données relationnelles.

Conventionnellement, le traitement des requêtes par un SGBD consiste à analyser les requêtes SQL et ensuite à produire une représentation interne de la requête. À partir de cette représentation, l'optimiseur génère un ensemble de plans physiques puis choisit le meilleur en termes de performance [3]. Comme les requêtes sont exprimées en langage SQL, l'analyseur commence par l'identification des mots-clés de la requête (les mots clés SQL, les noms des tables, les noms d'attributs, etc.) qui apparaissent dans le corps de la requête, puis vérifie qu'elles sont correctement formulées selon les règles syntaxiques (règles grammaticales) du langage, cette étape est appelée l'analyse syntaxique de la requête.

Ensuite, une validation sémantique est nécessaire pour s'assurer que la requête est valide, en vérifiant les noms des colonnes et des tables sont valides et sémantiquement significatifs par rapport au schéma de la BD (la colonne appartient à la table correspondante).

Une fois que la requête est valide syntaxiquement et sémantiquement, elle sera transformée en une structure de données arborescentes appelée *arbre algébrique* à travers des règles de transformation. En se basant sur cet arbre, le SGBD définit une stratégie d'exécution pour extraire les résultats de la requête à partir des fichiers de la BD. Une requête peut admettre plusieurs stratégies d'exécution, et le processus du choix de la stratégie optimale et la plus appropriée pour traiter la requête est connue sous le nom d'*optimisation de requête* [50]. La figure 2.5 donne un aperçu du processus de traitement des requêtes par un SGBD. En ce qui suit, nous détaillons les différentes étapes du traitement des requêtes.

3. Nous considérons l'objectif traditionnel des optimiseurs qui est la performance.

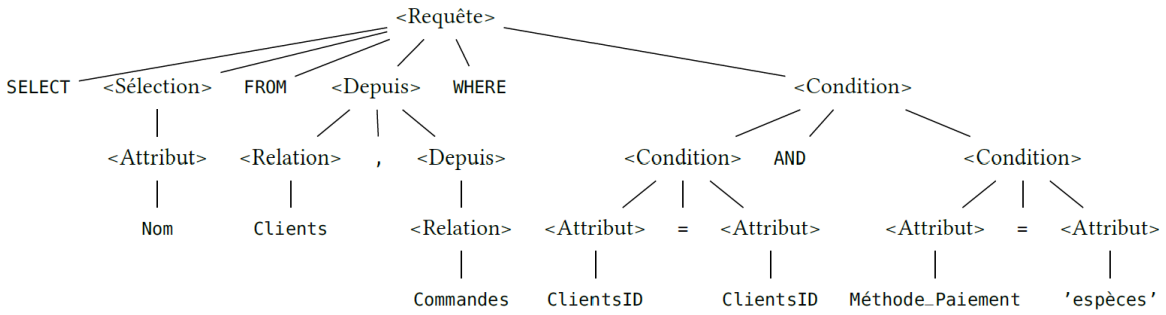


FIGURE 2.6 – Un arbre d'analyse.

2.3.1 L'analyse

Le rôle principal de l'analyseur est de récupérer du texte de la requête écrit en SQL et de le convertir en un arbre d'analyse. Les nœuds de cet arbre correspondent à des atomes, qui sont des éléments lexicaux tels que des mots-clés (par exemple, SELECT, FROM, WHERE), des noms des tables ou des attributs, des opérateurs tels que + ou <, des parenthèses, etc.

Une requête peut être syntaxiquement correcte, par contre, elle ne le soit pas sémantiquement. L'analyseur sémantique s'occupe de cette vérification pour que la requête soit sémantiquement correcte. Il s'agit de s'assurer que tous les noms après la clause $< From >$ réfèrent à des noms de tables appartenant au schéma de la BD, que les noms des colonnes dans les autres clauses existent dans les tables invoquées et qu'aucune contrainte d'intégrité n'est violée.

Une fois que les analyses syntaxique et sémantique soient réalisées, et à l'issu de cette phase, une structure algébrique sous forme d'un arbre est produite, dont les nœuds modélisent les opérateurs de l'algèbre relationnelle et les feuilles représentent les tables dans la BD. Des règles d'élimination de la redondance ou des règles de réécriture des expressions algébriques peuvent s'appliquer afin d'optimiser l'arbre algébrique [79].

Exemple 1 *La requête ci-dessous, permet de sélectionner les clients ayant passé une commande dans laquelle leur moyen de payement est "espèce". La figure 2.6 illustre l'arbre d'analyse de cette requête.*

```
SELECT Nom
FROM Clients, Commandes
WHERE Clients.ClientsID = Commandes.ClientsID
AND Methode_Paiement = especes';
```

2.3.2 La transformation

Dans cette étape, l'arbre produit à l'étape précédente est transformé en une expression d'algèbre relationnelle à travers l'application des heuristiques dans le but d'améliorer l'expression algébrique de la requête, en utilisant certaines des nombreuses règles algébriques issues de l'algèbre relationnelle. Ainsi, l'arbre d'expression sera transformé en un autre arbre équivalent permettant d'avoir un plan de requête physique plus optimisé. Le résultat des transformations appliquées est un plan de requête logique qui est la sortie de cette phase.

Nous présentons dans ce qui suit, un ensemble de règles algébriques qui servent à transformer un arbre algébrique en un arbre équivalent plus optimal en se basant sur l'algèbre relationnelle. Ces règles sont fondées sur les propriétés d'associativité et de commutativité des opérateurs algébriques. Le résultat de l'utilisation de ces heuristiques est une structure de données appelée plan de requête logique [56].

2.3.2.1 Règles de transformation algébriques

Nous pouvons scinder les opérateurs algébriques de base en deux : (1) les opérateurs binaires, qui permettent de retourner une nouvelle relation à partir de deux relations, tels que l'opération de l'union (U), la différence ($-$) et le produit cartésien (X) et, (2) les opérateurs un-aires tels que la projection (π), la sélection (σ), etc. Il est possible de déduire une combinaison d'évaluation des opérateurs permettant d'optimiser le résultat final en profitant de certaines propriétés mathématiques telles que la commutativité et l'associativité. Ci-dessous, nous présentons quelques règles de combinaison des opérateurs optimisant le processus de la transformation d'une requête [50][56].

1. **Décomposition de l'opérateur σ :**
   La sélection conjonctive peut être décomposée en une séquence d'opérations σ individuelles :
   $$\sigma_{c_1 \wedge c_2 \wedge \ldots \; and \; c_n}(\mathcal{R}) \equiv \sigma_{c_1}(\sigma_{c_2}(\ldots(\sigma_{c_n}(\mathcal{R})\ldots)))$$
2. **Commutativité de σ :**
   L'opérateur σ est commutative.
   $$\sigma_{c_1}(\sigma_{c_2}(\mathcal{R})) \equiv \sigma_{c_2}(\sigma_{c_1}(\mathcal{R}))$$
3. **Décomposition de l'opérateur π :**
   Dans une séquence de π, tous les π peuvent être ignorés sauf le dernier :
   $$\pi_{Liste_1}(\pi_{Liste_2}(\ldots(\pi_{Liste_n}(\mathcal{R}))\ldots)) \equiv \pi_{Liste_1}(\mathcal{R})$$
4. **Commuter σ avec π :**
   Si la contrainte de sélection c n'a besoin que des attributs $A_1, \ldots, A_n$ de la liste de projection, les deux opérations peuvent être commutées :
   $$\pi_{A_1,A_2,\ldots,A_n}(\sigma_c(\mathcal{R})) \equiv \sigma_c(\pi_{A_1,A_2,\ldots,A_n}(\mathcal{R}))$$
5. **Commutativité de $\bowtie$ (et de $\times$) :**
   L'opérateur de jointure et le produit cartésien sont commutatifs.
   $$\mathcal{R} \bowtie_c \mathcal{S} \equiv \mathcal{S} \bowtie_c \mathcal{R}$$
   $$\mathcal{R} \times \mathcal{S} \equiv \mathcal{S} \times \mathcal{R}$$
6. **Commuter σ avec $\bowtie$ (ou $\times$) :**
   Il est possible commuter les deux opérateurs dans le cas où les attributs de la contrainte (c) de sélection n'impliquent que les attributs d'une des relations jointes.
   $$\sigma_c(\mathcal{R} \bowtie \mathcal{S}) \equiv (\sigma_c(\mathcal{R})) \bowtie S$$
   Si la contrainte (c) peut être décomposée en $c_1 \wedge c_2$, où c_1 ne porte que les attributs de R et c_2 ne concerne que celui de S. Ainsi les opérateurs peuvent être commutés comme suit :
   $$\sigma_c(\mathcal{R} \bowtie \mathcal{S}) \equiv (\sigma_{c_1}(\mathcal{R})) \bowtie (\sigma_{c_2}(S))$$
7. **Commuter l'opérateur π avec $\bowtie$ (ou $\times$) :**
   Supposons que les attributs à projeter sont $L = \{A_1, \ldots, A_n, B_1, \ldots, B_n\}$, où $A_1, \ldots, A_n$ proviennent de la relation $\mathcal{R}$ et $B_1, \ldots, B_n$ de S. Dans le cas où la jointure n'implique que

les attributs de L, les deux opérateurs peuvent être commutés.

$$\pi_L(\mathcal{R} \bowtie_c \mathcal{S}) \equiv (\pi_{A_1,\ldots,A_n}(\mathcal{R})) \bowtie_c (\pi_{B_1,\ldots,B_m}(\mathcal{S}))$$

Si dans la condition c, il existe des attributs autres que ceux figurant dans L, un dernier opérateur de projection est ajouté contenant ces attributs. Par exemple, si les attributs $A_{n+1}, \ldots, A_{n+k}$ de $\mathcal{R}$ et $B_{m+1}, \ldots, B_{m+p}$ de $\mathcal{S}$ sont invoqués dans la condition c alors les opérateurs peuvent commuter comme suit :

$$\pi_L(\mathcal{R} \bowtie_c \mathcal{S}) \equiv \pi_L((\pi_{A_1,\ldots,A_n}(\mathcal{R})) \bowtie_c (\pi_{B_1,\ldots,B_m}(\mathcal{S})))$$

Pour $\times$, il n'y a pas de condition c, il faut juste remplacer $\bowtie_c$ par $\times$ dans la règle de transformation.

8. **Commutativité des opérateurs ensemblistes** :
Les opérations ensemblistes tels que $\cup$ et $\cap$ sont commutatives, par contre, la différence $(-)$ ne l'est pas.

9. **Associativité des opérateurs** $\bowtie, \times, \cup, \cap$:
Les opérateurs sont dits associatifs, si θ représente le même opérateur (un des quatre opérateurs), nous avons :
$$(\mathcal{R}\ \theta\ \mathcal{S})\ \theta\ \mathcal{T} \equiv \mathcal{R}\ \theta\ (\mathcal{S}\ \theta\ \mathcal{T})$$

10. **Commuter** σ **avec des opérations ensemblistes** $-, \cup, \cap$:
L'opération σ commute avec les opérations ensemblistes. Si θ représente l'une de ces trois opérations dans une expression, nous avons :
$$\sigma_c(\mathcal{R}\ \theta\ \mathcal{S}) \equiv (\sigma_c(\mathcal{R}))\ \theta\ (\sigma_c(\mathcal{S}))$$

11. **L'opération** π **commute avec** $\cup$:
$$\pi_L(\mathcal{R} \cup \mathcal{S}) \equiv (\pi_L(\mathcal{R})) \cup (\pi_L(\mathcal{S}))$$

12. **Conversion d'une séquence (**σ**,**$\times$**) en** $\bowtie$:
Si la condition c d'une σ qui suit un $\times$ correspond à une condition de jointure, alors, il est possible de convertir la séquence (σ,$\times$) en une $\bowtie$ qui peut s'écrire comme suit :
$$\sigma_c(\mathcal{R} \times \mathcal{S}) \equiv (\mathcal{R}\ \bowtie_c \mathcal{S})$$

13. **Éclater l'opérateur ensembliste** $-$ **avec** σ :
$$\sigma_c(\mathcal{R} - \mathcal{S}) = (\sigma_c(\mathcal{R})) - (\sigma_c(\mathcal{S}))$$

14. **Éclater l'opérateur ensembliste** $\cap$ **avec** σ **seulement sur un argument** :
$$\sigma_c(\mathcal{R} \cap \mathcal{S}) = (\sigma_c(\mathcal{R})) \cap (\mathcal{S})$$

15. **Élimination des doubles** :
L'opérateur δ élimine les doublons d'un ensemble. Si θ représente l'une de ces trois opérations : $\bowtie$, $\bowtie_c$, ou $\times$, dans une expression, alors :
$$\delta(\mathcal{R}\theta\mathcal{S}) \equiv \delta(\mathcal{R})\ \theta\ (\delta\mathcal{S})$$
$$\delta(\sigma_c\mathcal{R} \equiv \sigma_c(\delta\mathcal{R})$$

16. **Règles de groupement et d'agrégation** :
L'opérateur γ peut avoir plusieurs règles, par exemple, il peut absorber δ. Il est possible de projeter des attributs inutiles avant d'appliquer l'opération δ. Il est possible aussi de supprimer les doublons dans le cas d'agrégation de type MIN et MAX :

$$\delta\ (\gamma_L\ (\mathcal{R}))\ \equiv\ \gamma_L\ (\mathcal{R})$$

$$\gamma_L(\mathcal{R})\ \equiv\ \gamma_L(\pi_M(\mathcal{R}))$$

Si M est une liste contenant tous les attributs mentionnés dans L,

$$\gamma_L(\mathcal{R})\ \equiv\ \gamma_L(\delta(\mathcal{R}))$$

Lorsque L est MIN et/ou MAX.

2.3.2.2 Quelques règles d'amélioration du plan algébrique

En plus, des règles d'optimisation algébriques, il existe d'autres règles d'optimisation employées par l'optimiseur pour transformer un arbre algébrique initial en un plan logique plus optimal à s'exécuter. Dans le but de réduire la taille des données intermédiaires, les règles d'améliorations tentent d'évaluer le plus tôt possible les opérateurs de sélection afin de réduire le nombre de tuples et les opérateurs de projection, et par conséquent réduire le nombre d'attributs en les faisant descendre le plus loin possible dans l'arbre.

Nous présentons les règles les plus couramment utilisées :

1. Faire descendre l'opérateur de sélection ($SELECT$) vers le bas dans l'arborescence, dans la mesure du possible, à travers la décomposition de toute opération de $SELECT$. Dans le cas où une condition de sélection serait composée de AND et de plusieurs conditions, la condition peut être divisée en une séquence d'opérations de $SELECT$ et pousser chaque partie vers le bas de l'arbre d'une manière séparée.
2. L'opérateur de projection peut également être poussé vers le bas de l'arbre autant que possible, ou de nouvelles projections peuvent être ajoutées si nécessaire .
3. L'élimination des doubles peut être supprimée et/ou déplacée vers une position permettant d'avoir un résultat plus efficace dans l'arborescence.
4. La combinaison de certaines sélections avec un produit ci-dessous afin de transformer le couple d'opérations en une jointure, ce qui est généralement beaucoup plus pertinent que l'évaluation des deux opérations d'une manière séparée.

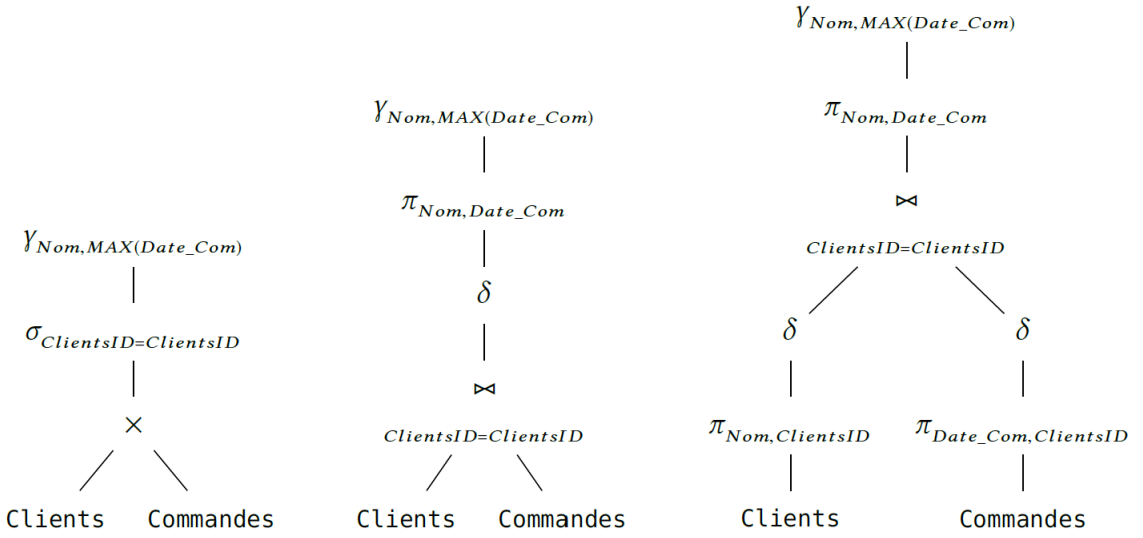
Exemple 2 *Soit la requête ci-dessous permettant d'afficher la date de commande la plus récente de chaque client.*

```
SELECT Nom, MAX(Date_Com)
FROM Clients, Commandes
WHERE Clients.ClientsID = Commandes.ClientsID
GROUP BY Nom;
```

En partant du plan de requête logique initial construit à partir de la requête (figure 2.7.a), nous pouvons appliquer les transformations suivantes :

— *Combiner la sélection et le produit dans une jointure,*
— *Générer un δ en dessous du γ,*
— *Générer un π entre le δ et le γ généré pour la projection sur Nom et Date_Com.*

Après les premières transformations, le nouveau plan est illustré dans la figure 2.7.b. Maintenant, il est devenu possible de pousser γ sous la $\bowtie$ et introduire l'opérateur π comme présenté dans la figure 2.7.c.



(a) Plan de requête logique initial.

(b) Plan de requête après la première transformation.

(c) Plan de requête après la deuxième transformation.

FIGURE 2.7 – Les transformations d'un plan de requête logique

2.3.3 La génération des plans et optimisation

Après l'analyse et la transformation de la requête en un plan de requête logique, ce dernier doit être ensuite traduit en un plan de requête physique (*PRP*). Le plan physique est modélisé à travers un arbre de requête qui est dérivé du plan logique. Il contient les informations des méthodes d'accès (séquentiel/indexé/parallèle) aux données et les algorithmes à utiliser pour exécuter les opérateurs relationnels appartenant à l'arbre (algorithme de jointure/algorithme de tri). Généralement, plusieurs plans physique sont générés et sont ensuite évalués. À ce niveau, l'optimiseur estime le coût de chaque opération et les ordonne afin d'obtenir le coût du plan. Cette méthode est appelée *la génération de plans basée sur le modèle coût*, qui permet de retourner le PRP ayant le coût optimal. La difficulté consiste à estimer les coûts des plans avec précision. Ces estimations sont basées sur (1) des statistiques sur données de la base (taille de la table, longueur d'un tuple, l'utilisation des indexes, etc), et (2) des statistiques du système (l'accès disques (entrées/sorties), le temps CPU pour exécuter les opérateurs, etc).

2.3.4 L'exécution

L'étape finale du processus du traitement d'une requête est son exécution. Lors de l'exécution, toutes les opérations d'E/S et de CPU indiquées dans le plan physique sont exécutées. Il existe 3 modes pour l'exécution d'un opérateur : (1) mode séquentiel (matérialisé) (2) pipeline et/ou, (3) parallèle.

Lors d'une exécution matérialisée, le résultat d'une opération est stocké physiquement sur le dispositif de stockage comme une relation temporaire avant d'être récupéré par l'opération suivante. Ce coût s'ajoute au coût global du traitement de requête. Le coût de ce mode peut s'avérer très élevé dans le cas où la taille de la relation intermédiaire produite est supérieure à la mémoire tampon et qu'elle doit être stockée sur les mémoires secondaires.

Le mode en pipeline, permet aux tuples résultants d'une opération à être envoyés directement à l'opération suivante dans la séquence de requête. Ce mode permet d'économiser le

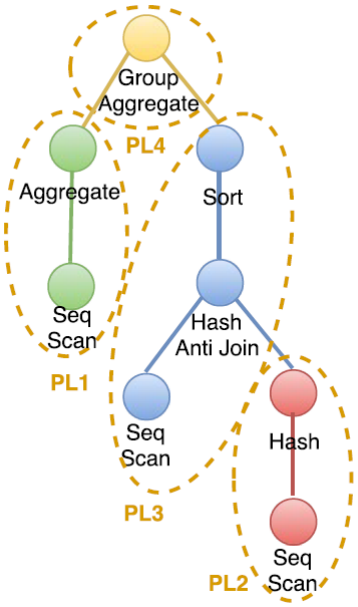


FIGURE 2.8 – La segmentation du plan d'exécution en pipelines

coût d'écriture des résultats intermédiaires sur le disque et la relecture pour l'opération suivante. Ce qui offre la possibilité de produire des résultats rapidement pour certaines opérations algébriques. Autres opérations telles que le tri et le groupement, ne peuvent pas prendre en charge ce mode parce qu'ils ne produisent aucune sortie tant qu'ils n'ont pas encore consommé complètement leurs entrées, ils sont dits des *opérateurs bloquants*. Dans un plan d'exécution, les pipelines sont délimités par les opérateurs bloquants. La figure 2.8 montre un exemple de la segmentation en pipeline d'une requête. Le mode parallèle est détaillé dans la section 2.5.2.

2.4 Les techniques d'évaluation des performances des bases de données

Les industriels, les développeurs, les chercheurs, ainsi que les utilisateurs intéressés par les $BnFs$ doivent être en mesure d'évaluer et de comparer les systèmes informatiques en général et les bases de données en particulier afin de prendre des décisions d'achat ou d'identifier les technologies prometteuses. Ainsi, beaucoup d'efforts ont été consacrés par les industriels et les chercheurs pour identifier des mesures et des techniques d'évaluation de BnF. L'évaluation de performance dans les systèmes informatiques concerne trois niveaux : (1) le matériel (2) le système d'exploitation et (3) l'application. Dans les applications de bases de données, deux métriques sont utilisées pour évaluer la performance, (*i*) les modèles de coût (MdC), et (*ii*) le benchmarking.

Modèles et métriques de coût La consommation réelle des ressources d'un système dépend de plusieurs facteurs, tels que la charge du travail, les composants matériels et les paramètres environnementaux. Dans le but de mesurer la consommation de ressources, des modèles d'évaluation précis pour des composants (processeur, mémoire, disque, carte graphique, réseau, etc.), des systèmes, et des applications, sont nécessaires. Dans la littérature, il existe deux types d'approches de modélisation (*i*) les approches basées sur les analyses détaillées dites approches analytiques, et (*ii*) les approches boîtes noires dites approches expérimentales [145].

Tableau 2.1 – Classification des benchmarks existants

Benchmark	Type de bases de données	BnF
TPC-C	Base de données transactionnelle	Débit des transactions Performance
TPC-H	Base de données décisionnelle	Performance
SSB	Base de données décisionnelle	Performance
OO7	Base de données Objet	Performance Efficacité de mise a jour
XOO7	Base de données XML	Utilisation espace mémoire Temps de conversion de données
YCSB	Base de données NoSQL	Latence de lecture Latence de mise a jour

— Les modèles de coût analytiques : Ce sont les approches classiques employées par les premiers travaux sur les modèles de coût. Ce type de MdC nécessite des connaissances détaillées du SGBD cible, de l'implémentation des opérateurs algébriques, des statistiques de la base de données, et des caractéristiques du matériel utilisé, afin de déterminer analytiquement et avec précision le coût d'exécution des requêtes. Cette approche est adaptée par le SGBD open-source PostgreSQL.
— Les modèles de coût expérimentaux : Ce type de MdC considère le SGBD comme une boîte noire. Il consiste à identifier, au premier lieu, un ensemble de caractéristiques de requêtes appelées des requêtes d'apprentissage, et de données qui déterminent potentiellement les coûts des opérateurs. Puis, ces requêtes seront exécutées sur le système. Ensuite, les modèles statistiques ou bien d'apprentissage automatique, des données collectées, sont employés, ce qui permet d'identifier les paramètres finaux du modèle de coût et la relation entre eux. Cette approche est également utilisée par les SGBD commerciaux comme Oracle.

Benchmarking Le banc de tests communément appelé par les anglophones les "*Benchmarks*", est un outil d'analyse, de sensibilisation et de motivation des chercheurs. Une définition formelle est donnée dans la définition 1. Son objectif est de définir des références pour comparer les solutions proposées en fonction des BnF définis. La mise en oeuvre d'un benchmark consiste à effectuer un ensemble de tests sur un SGBD afin d'estimer ses performances dans certaines conditions.

Définition 1 (Benchmarking) *Soumettre un système à une série de tests afin d'obtenir des résultats préétablis non-disponibles sur les systèmes concurrents". En d'autres termes, le benchmark est une mesure utilisée pour différencier deux ou plusieurs systèmes.*

Pour estimer les différents BnFs d'une base de données, les chercheurs, les agences gouvernementales et les consortiums standards de l'industrie pour les mesures de la performance ont proposé plusieurs benchmarks, où chacun permet de mesurer l'efficacité et la pertinence d'un BnF. Le tableau 2.1 classifie les différents benchmarks dans le domaine des bases de données.

Dans le but de qualifier un banc d'essai de bon, ce dernier doit satisfaire un ensemble de critères primordiaux. Un benchmark doit être simple et compréhensible pour gagner en crédibilité. Il doit être portable et facile à implémenter sur de différents SGBD. Un bon benchmark doit assurer le passage à l'échelle en donnant la possibilité d'étudier des bases données de tailles diverses et de permettre l'augmentation l'échelle. Et le plus important,

un bon benchmark doit être pertinent et il permet de répondre aux besoins du plus grand nombre d'utilisateurs possible.

Malgré l'efficacité des benchmarks, surtout dans l'évaluation des algorithmes de sélection des structures d'optimisation (sélection des indexes, vues matérialisées, etc.), ils présentent, néanmoins, quelques limites. La majorité d'entre eux ont un schéma figé, ce qui limite leur utilisation. Les concepteurs des benchmarks ont opté pour la conception de bancs de tests simple afin de satisfaire les critères d'un bon benchmark, ce qui les a conduits à concevoir des benchmark très ciblés, rarement réutilisables dans d'autres contextes que celui dans lequel ils ont été créés.

2.5 Les techniques d'optimisation dans les bases de données

Dans les premières générations des BD, la majorité des travaux étaient principalement orientés vers l'optimisation logique. Ceci est dû à la simplicité des requêtes de cette époque. En revanche, ces optimisations se sont révélées insuffisantes face aux BD de plus en plus larges, et des requêtes complexes impliquant des jointures et des agrégations [32]. Dans cette section, nous présentons les différentes techniques adoptées.

2.5.1 Les structures d'optimisations

Le réglage des performances (tuning) dans le système de gestion des bases de données signifie améliorer les performances de la base de données, c'est-à-dire minimiser le temps de réponse à un coût très optimal. Comme le temps de réponse d'une requête est la première métrique en matière de performance de base de données, l'optimisation des requêtes est l'un des aspects importants du réglage des performances [69]. L'objectif de l'optimisation des requêtes est de fournir un temps de réponse minimum et un débit maximum avec une utilisation efficace des ressources. L'optimisation des requêtes signifie principalement la sélection, suivie d'un séquençage dans un ordre spécifique, des différentes clauses SQL pour formuler une requête efficace, à partir des multiples plans, en établissant une comparaison des plans de requête en fonction du coût des ressources impliquées et du temps de réponse [69].

La conception physique de la base de données vise à maximiser l'efficacité du traitement des données. Lors de la conception d'une base de données, l'accent est mis sur la réduction du temps d'exécution des requêtes. Le processus de conception physique de la BD nécessite de prendre plusieurs décisions cruciales qui auront un impact sur les performances des applications utilisant la base de données. Ces décisions incluent le format de stockage, la conception des colonnes, l'utilisation appropriée des types de données, etc, comme précédemment expliqué dans le section 2.2.4.

Dans la littérature, les structures d'optimisation existantes sont classées en deux catégories en se basant sur la redondance des données liée à leur implémentation sur le support physique. Ainsi, si une structure ne génère pas un coût de stockage supplémentaire, alors, elle est dite non-redondante, et elle est dite redondante dans le cas contraire comme illustré dans la figure 2.9.

La sélection des structures d'optimisation peut s'effectuer selon deux modes, (*i*) mode isolé, et (*ii*) mode multiple. Dans le mode isolé, l'administrateur de la BD sélectionne une seule structure d'optimisation afin de satisfaire les $BnFs$ de sa charge de travail. Par contre, dans le mode multiple, deux ou plusieurs structures d'optimisation sont sélectionnées. Dans ce mode de sélection, deux solutions principales sont adoptées (*a*) une solution séquentielle (naïve) et (*b*) une solution conjointe.

Le problème de sélection multiple des SO est formulé comme un problème d'optimisation comme suit :

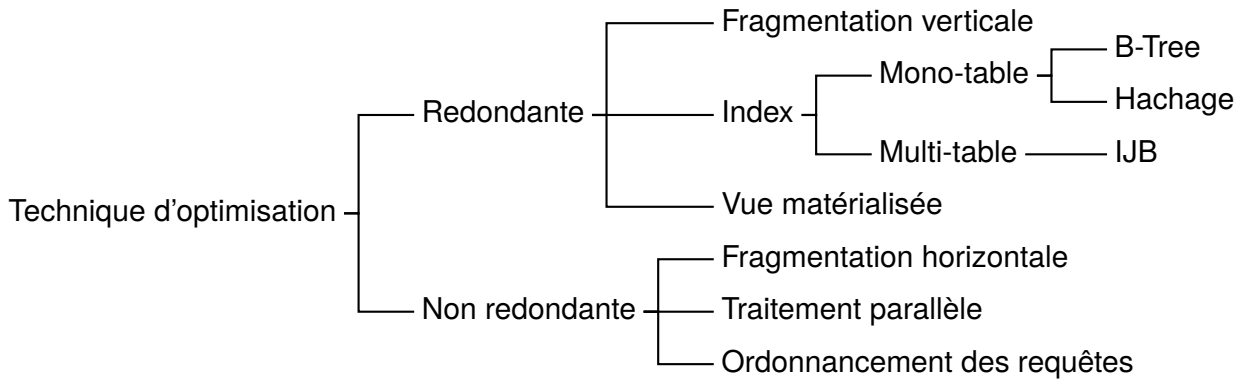


FIGURE 2.9 – Classification des techniques d'optimisation

Soit :

— Une charge de travail composée de m requêtes $Q = \{Q_1, Q_2, \ldots, Q_m\}$;

— Un ensemble de contraintes $C = \{C_1, C_2, \ldots, C_p\}$, par exemple l'espace de stockage pour les SO redondantes ou le nombre de fragments générés pour le cas de la fragmentation horizontale.

Le problème de sélection multiple des structures d'optimisation consiste à sélectionner un ensemble de SO selon les préférences de l'administrateur tels que :

— Le coût d'exécution des requêtes Q en présence de l'ensemble des SO sélectionnées soit réduit ;

— Les contraintes appartenant à C soient satisfaites.

Généralement, les structures d'optimisation sont sélectionnées dans le seul but d'améliorer la performance, cependant, nous avons vu qu'il existe plusieurs BnFs à optimiser (dans cette thèse, nous considérons la consommation énergétique comme un BnF). Ce qui nous mène à migrer le problème de sélection des SO, d'un problème mono-objectif à un problème multi-objectif qui consiste à optimiser plus d'un BnF simultanément. Afin de résoudre ce nouveau problème, il est nécessaire de développer des nouvelles techniques qui permettent d'offrir le meilleur compromis entre la minimisation du temps de réponse et la réduction de la consommation d'énergie du système. La plupart du temps, la solution optimale dans le cas de l'optimisation mono-objectif, est clairement définie, ce qui n'est pas valable pour le problème d'optimisation multi-objectif, où au lieu d'un seul optimum, il existe plutôt un ensemble de compromis alternatifs. Pour résoudre ce problème, plusieurs méthodes sont disponibles, une description de ces méthodes est donnée dans la section 4.3.2.2.

Dans cette section, nous analysons la sélection isolée, et nous présentons les différentes structures d'optimisations en détaillant les plus répandues et celles que nous allons utiliser durant cette thèse.

2.5.1.1 Les indexes

La très grande taille des bases de données, rend la recherche de données un processus très lent, ce qui conduit à chercher des solutions pour éviter les accès à des tables entières. Pour accéder rapidement au bloc de données, des indexes peuvent être utilisés. Ces derniers fournissent des pointeurs vers le bloc de données où les données prévues sont disponibles. Par exemple, un index peut être créé pour trouver les détails de l'employé à partir d'un identifiant d'un employé particulier. L'administrateur de base de données (DBA) crée des indexes dans une base de données dans le cadre des techniques d'amélioration de performance. Cependant,

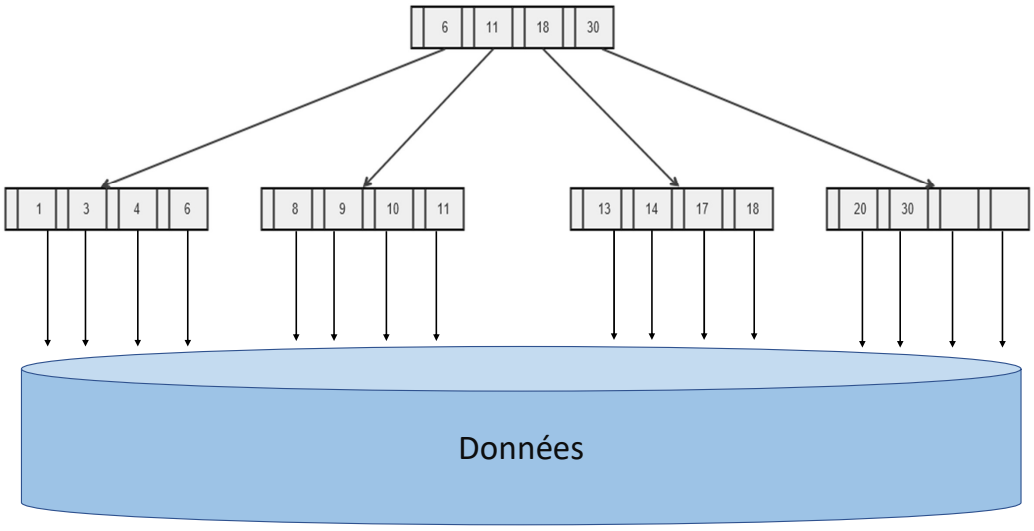


FIGURE 2.10 – Index B-tree

l'utilisation des indexes doit être faite avec précaution, car un trop grand nombre d'index sur une table peut dégrader les performances.

En raison des différents types d'applications, le type de données stockées dans la BD, le format des données et, surtout, le type de requêtes exécutées, le choix du type d'index adéquat devient une nécessité primordiale. Le DBA doit être bien conscient aux types d'index afin de prendre une décision appropriée dans le choix de l'index pertinent. L'administrateur doit choisir l'index correspondant en fonction du type d'informations à extraire de la base de données. Il existe plusieurs types d'index, les indexes par hachage, les indexes B-Tree, les indexes de jointure, les indexes binaires, les indexes de jointure binaire. En ce qui suit, nous présenterons les indexes les plus utilisés et nous détaillerons l'index de jointure binaire que nous allons proposer une approche énergétique de sa sélection dans cette thèse.

2.5.1.1.1 L'index B-Tree L'index B-Tree est l'index utilisé par défaut dans la majorité des SGBD commerciaux, par exemple, Oracle crée par défaut un index B-tree sur la clé primaire de chaque table. Comme son nom l'indique, il est organisé sous forme d'un arbre à plusieurs niveaux, où chaque nœud d'un niveau pointe vers un niveau inférieur. Les nœuds du niveau le plus bas, appelés feuilles, contiennent les entrées d'index ainsi qu'un pointeur vers l'emplacement physique de l'enregistrement correspondant (généralement un identifiant physique, ROWID) comme illustré dans la figure 2.10.

2.5.1.1.2 L'index par hachage L'index par hachage utilise une fonction de hachage qui permet, à partir d'une valeur de la clé c, de donner l'adresse $f(c)$ d'un espace de stockage où l'élément est placé. La figure 2.11 montre un index de hachage construit sur l'attribut PID de la table Produit. Dans ce type d'index, le choix de la fonction de hachage est très délicat et important pour garantir l'efficacité de l'index. Par exemple, si la fonction de hachage donne la même valeur à un nombre important d'éléments, alors l'accès à l'index devient un balayage séquentiel, ce qui rend l'index inutile. La fonction de hachage utilisée dans l'exemple de la figure 2.11 calcule le reste de la division de l'attribut clé par 7.

2.5.1.1.3 Les indexes de jointure binaires Les indexes de jointure binaires (IJB) sont une combinaison entre les indexes de jointure et les indexes binaires [129], ils permettent d'optimiser à la fois les opérations de sélection et les opérations de jointure. Dans les applications d'entrepôt de

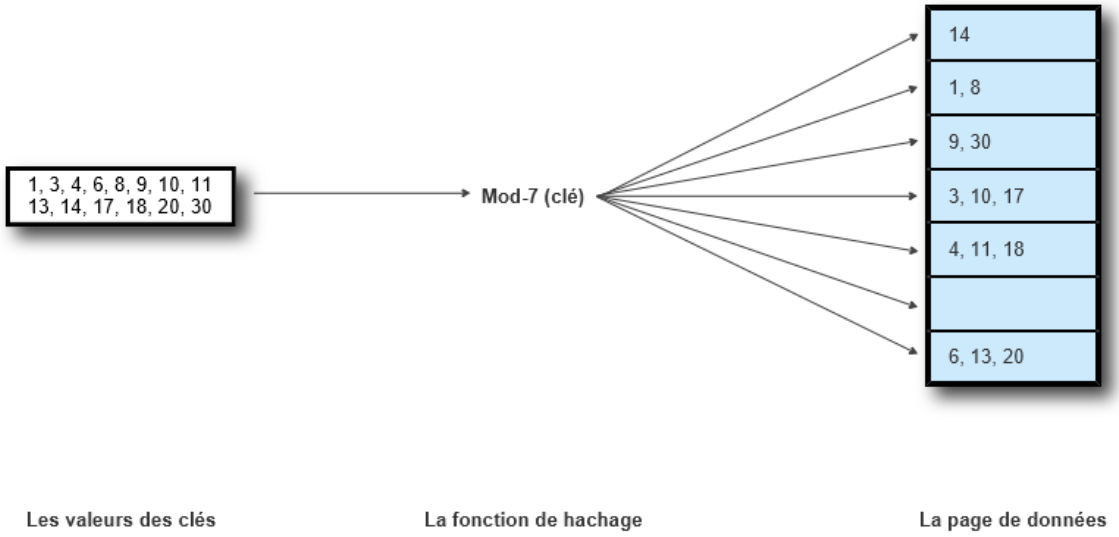


FIGURE 2.11 – Index par hachage

données, ils sont définis entre une table de faits et une ou plusieurs tables de dimensions basées sur une ou plusieurs colonnes (appelées attributs indexables) avec une cardinalité raisonnable (comme l'attribut sexe) issus d'une ou plusieurs tables de dimensions, contrairement aux indexes binaires et les indexes B-Tree, où les attributs indexés ne peuvent appartenir qu'à une seule table. Bien que la faible cardinalité permet de gagner en espace de stockage, cependant, elle signifie que la sélectivité devient supérieure à celle des attributs ayant une cardinalité plus élevée.

Indexer tous les attributs d'une table de grande taille avec un index B-tree traditionnel peut être interdit en termes d'espace de stockage occupé par les indexes en raison que ces derniers peuvent être plusieurs fois plus grands que les données de la table. Généralement, les IJB représentent seulement une petite portion de la taille des données indexées dans la table. De plus, et contrairement à la plupart des autres types d'index, les IJB incluent les lignes qui ont des valeurs NULL, l'indexation de ces valeurs peut être utile pour certains types d'instructions SQL, comme les requêtes utilisant la fonction d'agrégation COUNT. Ils aident en fait l'optimiseur à fournir une estimation suffisamment bonne, car ils réduisent les données extraites de la table de faits et permettent à l'optimiseur de corriger/calculer la cardinalité des jointures de manière moyenne.

L'encodage binaire de ces indexes favorise leur compression [109]. Ils sont largement utilisés dans les environnements caractérisés par une grande quantité de données à gérer, des requêtes d'interrogation de données du type "SELECT" et ayant un niveau faible de requêtes de manipulation de données. Les entrepôts de données présentent un environnement idéal pour l'utilisation des IJB. Pour ces environnements, l'IJB permet de (*i*) réduire considérablement le temps d'exécution des requêtes, (*ii*) réduire le coût de stockage par rapport à d'autres techniques d'indexation pour la même quantité de données grâce à sa nature binaire, (*iii*) améliorer la performance même sur un matériel peu performant (mémoire limitée/nombre faible d'unité de traitement), (*iv*) avoir une maintenance facile et efficace. Par conséquent, ils offrent un compromis entre la performance et le stockage. D'autre part, il est nécessaire de mentionner que les IJB sont généralement plus faciles à détruire et à recréer qu'à maintenir [4]. Un exemple de la création et de l'utilisation de l'IJB est fourni dans l'exemple 3.

4. https://docs.oracle.com/cd/E18283_01/server.112/e16579/indexes.htm

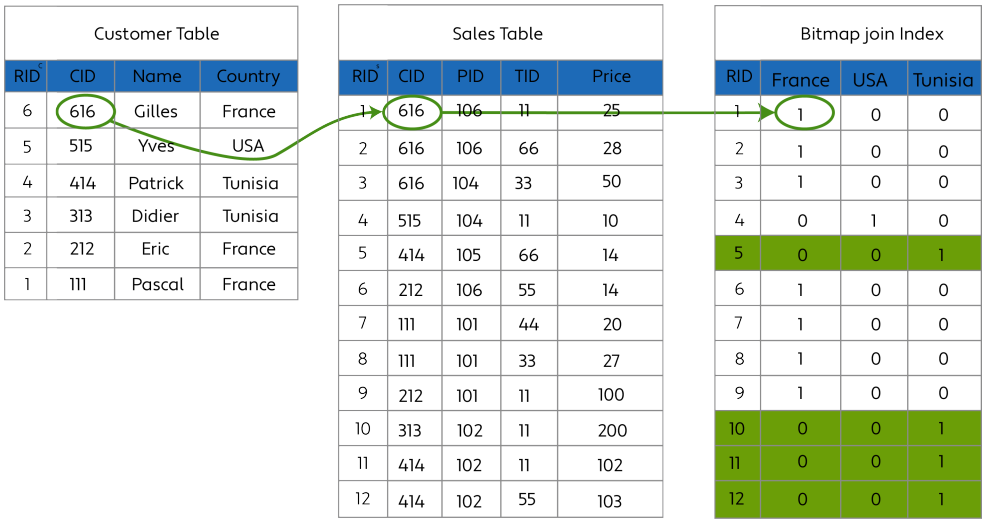


FIGURE 2.12 – Index de jointure binaire

Exemple 3 *La requête SQL suivante crée un IJB de l'attribut Country de la table de dimension Customer et de la table de faits Sales.*

```
CREATE BITMAP INDEX ijb_Country
ON Sales(Customer.Country)
FROM Sales S, Customer C
WHERE S.CID= C.CID
```

L'instruction ci-dessus crée un fichier physique comme suit :

1. *Créer un fichier avec $p+1$ colonnes représentant l'identifiant de ligne (ROWID) de la table de faits et les différentes valeurs de l'attribut Country (trois dans notre cas : France, USA et Tunisia) ;*
2. *Chaque vecteur correspondant à une valeur de l'attribut Country est défini par une valeur binaire. Précisément, le i^{me} bit du vecteur correspondant à une valeur v_k, est mis à 1 si le tuple de la table de faits ayant le ROWID i correspond à v_k . Sinon, le i^{me} bit du vecteur est mis à 0.*

La figure 2.12 montre un échantillon de la population de notre IJB. Cette illustration montre l'intérêt de cet index en termes de réduction de l'espace de stockage, de compression [6, 94] et d'accélération des requêtes analytiques du type $Count$ et SUM, et leur gestion des opérations booléennes grâce a leur nature binaire, où l'accès à l'index peut retourner le résultat de la requête.

Pour illustrer le principe fondamental de fonctionnement des IJB, considérons leur contexte favori représenté par un entrepôt de données relationnel avec une table de faits TF et un ensemble de tables de dimensions $Dim = \{DT_1, \ldots DT_L\}$. La formalisation orientée performance du problème de sélection des IJB ($PSIJB$) peut être présentée comme suit :
étant donné :

(a) un entrepôt de données relationnel avec une table de faits TF et L tables de dimensions $Dim = \{DT_1, \ldots, DT_L\}$,

(b) une charge de travail avec m requêtes $Q = \{Q_1, Q_2, \ldots, Q_m\}$, et

(c) un ensemble de contraintes telles que le coût de stockage et le coût de maintenance.

Le PSIJB vise à sélectionner une configuration d'indexes qui minimise le temps d'exécution global des requêtes et satisfait les contraintes fixées. Trouver les IJB optimaux nécessite de tester $2^n - 1$ configurations d'IJB, où n représente le nombre d'attributs indexables. Les applications commerciales actuelles d'entrepôt de données peuvent impliquer un grand nombre de tables et

de requêtes. Par exemple, l'article [3] dont les auteurs sont tous de la société Oracle publié dans VLDB 2020 montre des expérimentations impliquant un schéma de base de données avec 200 tables et un ensemble de 650 requêtes. Dans les requêtes analytiques dans le cloud, le nombre d'attributs indexables peut être très élevé, ainsi, trouver les meilleurs IJB pour une charge de travail donnée devient une tâche très difficile [9].

Le problème de sélection d'index de jointure binaire est connu pour être NP-difficile [7] en raison de la grande quantité de données et du nombre exponentiel d'attributs candidats pouvant être utilisés dans le processus de sélection. Par conséquent, des techniques de recherche heuristiques sont utilisées pour résoudre ce problème [31]. La majorité des approches de sélection des IJB utilise une sélection automatique des attributs indexables de la charge de travail puisque le choix manuel des attributs à indexer, est à la fois subjectif et un peu difficile à atteindre surtout lorsque le nombre de requêtes de la charge de travail est important [9].

Pour réduire cette complexité, il est nécessaire de passer, dans un premier temps, par une phase d'élagage pour réduire le nombre d'attributs candidats. Plusieurs travaux se sont intéressés à cette phase d'élagage qui s'appuie sur certains critères de sélection privilégiant des attributs par rapport à d'autres tels que la fréquence d'utilisation, la cardinalité des attributs, la taille des tables, etc.

Une panoplie d'approches a été proposée dans la littérature pour résoudre le problème de sélection des IJB. La majorité de ces approches commence par l'identification d'attributs indexables, qui peut être manuelle ou bien automatique. Ensuite, ils utilisent de différentes techniques pour élaguer l'espace de recherche et sélectionner la configuration finale d'indexes. Parmi ces techniques, nous pouvons citer, les algorithmes de fouille de données [7, 9, 17, 122, 123], les algorithmes basés sur la programmation linéaire [171], les systèmes immunitaires artificiels [54], les algorithmes génétiques [24], ou la traverse minimale d'un hypergraphe [62].

Aouiche et al.[7] ont proposé une approche basée sur la fouille de données où ils exploitent des itemsets fermés fréquents (FCI) en utilisant l'algorithme Close [135]. Un itemset est fermé s'il n'a pas de sur-ensemble avec le même *support*. Ce dernier, est le nombre de transactions qui contiennent l'attribut a. Ainsi, un itemset est fréquent lorsque $sup(a)$ atteint au moins un seuil spécifié par l'utilisateur appelé *minsup*. Ensuite, pour chaque FCI, un algorithme glouton basé sur un modèle de coût mathématique est exécuté pour sélectionner une configuration finale d'IJB. Dans leur proposition, les auteurs ne considèrent que la fréquence d'accès aux attributs comme critère pour élaguer l'espace de recherche. Le modèle de coût définit par les auteurs prend en entrée une requête et un plan d'exécution et retourne le coût de ce plan en termes d'opérations d'E/S. Ce coût est calculé en utilisant des formules qui prennent en compte un certain nombre de paramètres dont les plus importants sont représentés sur le tableau 2.2, et des statistiques collectées sur la base de données. En ce qui suit, nous détaillons ce modèle de coût.

Tableau 2.2 – Les paramètres du modèle de coût

Symbole	Description
$\|T\|$	Nombre de n-uplets de la table T ou cardinalité de l'attribut X
S_p	Taille en octets d'une page disque
PT	Nombre de pages nécessaires pour stocker la table T
$S_{pointeur}$	Taille en octets du pointeur d'une page
m	Ordre d'un B-tree
d	Nombre de bitmaps utilisés pour évaluer une requête donnée
$w(X)$	Taille en octets d'un n-uplet de la table T oude l'attribut X

L'espace de stockage occupé par un IJB dépend de deux facteurs : (*i*) le nombre de lignes de la table de faits, et (*ii*) la cardinalité de l'attribut indexable.

Soit l'index I défini sur n attributs $A_1, A_2, .., A_n$, l'espace de stockage en octets de I est calculé par la formule suivante :

$$Taille(I) = \frac{(|RowID| + \sum_{j-1}^{n} |A_j|) \times |F|}{8} \tag{2.1}$$

Le coût d'exécution d'une requête est exprimé en nombre d'entrées-sorties nécessaires pour son exécution. En se basant sur les attributs indexés de la configuration d'indexes, trois scénarios sont possibles pour l'exécution de Q : *(i)* aucun attribut indexable de Q n'appartient à CI ; *(ii)* tous les attributs indexables de Q appartiennent à CI ; et *(iii)* quelques attributs indexables de Q appartiennent à CI.

1. **Scénario 1 : Aucun attribut de Q n'appartient à CI :** dans le cas d'absence d'IJB utilisés par Q, toutes les jointures de Q peuvent être calculées en utilisant la jointure par hachage. Le nombre d'E/S nécessaires pour joindre la table R avec la table S en utilisant la jointure par hachage est :
$$C-_{hash} = 3(pS + pR) \tag{2.2}$$
où pX est le nombre de pages nécessaires pour stocker la table X.
2. **Scénario 2 : Tous les attributs de Q appartiennent à CI :** ce cas représente la situation idéale, dans laquelle toutes les jointures de Q sont pré-calculées. Dans ce cas, l'exécution de Q passe par deux étapes importantes : le chargement des indexes, ensuite, l'accès aux données. Par conséquent, deux coûts sont considérés : le coût de chargement des indexes et le coût d'accès aux données.
   — *Le coût de chargement de l'index :* le coût de chargement d'un IJB noté $CC(I)$ correspond au nombre de pages lues pour le charger. Il est calculé par la formule suivante :
   $$CC(I) = \frac{Taille(I)}{PS} \tag{2.3}$$
   Le coût de chargement de l'ensemble des indexes utilisés par une requête Q est égal à la somme des coûts de chargement de tous les indexes.
   — *Le coût d'accès aux n-uplets :* soit N_t le nombre de n-uplets de la table de faits référencés par la requête Q, le coût de lecture (CL) de ces n-uplets est donné par la formule suivante :
   $$CL = \|F\|(1 - e^{-\frac{N_t}{\|F\|}}) \tag{2.4}$$
   où $\|F\|$ désigne le nombre de pages nécessaires pour stocker la table de faits F.
3. **Scénario 3 : Quelques attributs de Q appartiennent CI :** dans ce cas, l'exécution de Q se fait en deux phases. Lors de la première phase, les IJB utilisés par Q sont chargés et utilisés pour trouver un ensemble de n-uplets de la table de faits, qui seront utilisés par la suite. Le coût de cette phase est calculé comme celui du scénario 2. Dans la deuxième phase, les jointures non-effectuées à cause de l'absence d'IJB, sont effectuées entre les n-uplets de faits résultants de la première phase, et les tables de dimension non encore jointes. Le coût de cette étape est calculé en utilisant la formule du premier scénario.

Bellatreche et al.[17] ont proposé l'algorithme DynaClose, qui est une amélioration de l'approche d'Aouiche et al.[7] en introduisant des paramètres d'élagage tels que la taille de la table de l'attribut et sa fréquence et une fonction fitness à calculer pour chaque FCI. L'algorithme proposé pénalise les FCI définis sur des tables de petite taille.

Dans [123], les auteurs ont introduit une approche distribuée basée sur l'itemset fréquent maximal (MFI). Un itemset est maximal s'il n'a pas de sur-ensemble fréquent. Pour élaguer l'espace de recherche, les auteurs ont introduit une nouvelle métrique d'élagage appelée $Weight$.

La configuration finale d'indexes est construite à travers les MFI qui optimisent la performance en se basant sur le modèle de coût et sans violer la contrainte de stockage.

Dans [24], les auteurs ont proposé une approche qui s'appuie sur un algorithme génétique de sélection mono et multiple des IJB. Ils ont représenté le chromosome à travers un tableau binaire d'attributs indexables où une entrée est égale à 1 si l'attribut est sélectionné et 0, sinon. Pour sélectionner la CFI, la fonction objective est basée sur un modèle de coût et la taille de l'index sélectionné.

Gacem et al. [54], ont utilisé les systèmes immunitaires artificiels (SIA) pour résoudre le problème de sélection des IJB. Dans leur approche, les auteurs ont considéré l'IJB comme des anticorps, et les requêtes comme des antigènes. Les expérimentations montrent que cette approche est mieux adaptée aux environnements disposant de grands espaces de stockage.

Dans [62], les auteurs ont proposé une approche qui se base sur une modélisation en hypergraphe de l'espace de recherche et qui utilise les traverses minimales et une fonction fitness pour déterminer l'ensemble d'attributs candidats et pour choisir la configuration finale d'index. Les expérimentations menées sur deux benchmarks différents montrent l'efficacité de cette méthode.

Les auteurs dans [173], ont proposé GBPSO qui est une approche parallèle basée sur l'essaim de particules binaire et le GPU pour résoudre le $PSIJB$ dans les entrepôts de données. Ils ont testé leur approche sur trois classes de problème en variant la taille à chaque fois. Yu et al. [193] ont proposé un mécanisme hybride (CPU/GPU), où le GPU est utilisé comme conteneur et processeur de l'IJB pour améliorer les performances de jointure en étoile. Le tableau 2.3 récapitule les différentes approches proposées dans la littérature et montre leurs ressemblances et leurs différences.

Pour réduire la taille des indexes binaires ce qui permet de les construire sur des attributs de cardinalité moyenne, les techniques de compression ont également été largement utilisées. La compression réduit la taille de l'index binaire et permet également d'effectuer les opérations logiques sans décompresser entièrement l'index. Cependant, la compression génère un coût d'exécution supplémentaire par rapport aux indexes non compressés et peut rendre les mises à jour plus coûteuses [81]. Dans la littérature, plusieurs techniques de compression ont été proposées [6, 81, 94, 182]. Une revue des différentes techniques a été proposée dans [35], dans laquelle un examen minutieux de ces approches a été proposé.

Tableau 2.3 – Les approches de sélection des IJB

Approche	Algorithme de sélection	Plateforme	Type d'IJB	Mis à jour	Benchmark
Aouiche et al. [7]	Technique de fouille de données (FCI) algorithme glouton guidé par un modèle de coût 3 fonctions objectives	Centrale	Mono-attribut	Non	APB
Bellatreche et al. [17]	Technique de fouille de données (FCI) algorithme glouton guidé par un modèle de coût fonction Fitness	Centrale	Mono-attribut	Non	APB
Azefack et al. [9]	Technique de fouille de données MFI algorithme glouton guidé par un modèle de coût	Centrale	Mono-attribut	Oui	-
Necir et al. [122]	Technique de fouille de données MFI + algorithme glouton guidé par un modèle de coût Fonction Weight	Centrale	Mono-attribut	Non	APB
Necir et al. 2015 [123]	Technique de fouille de données MFI + multi agent modèle de cout	Distribuée	Mono-attribut	Non	APB
Gacem et al. [54]	Systèmes immunitaires artificiels modèle de coût	Centrale	Mono-attribut	Non	APB
Bouchakri et al. [24]	Algorithme génétique basé sur un modèle de coût	Centrale	Mono-attribut + multi-attributs	Non	APB
Bellatreche et al. [14]	algorithme glouton guidé par un modèle de coût	Centrale	Mono-attribut + multi-attributs	Non	APB
Toumi et al. [171]	Programmation linéaire modèle de coût	Centrale	Mono-attribut	Non	APB
Ghabry el a. [62]	Traverse minimale fonction Fitness	Centrale	Mono-attribut	Non	TPC-H & SSB1
Toumi et al. [173]	Essaim de particules binaire	Centrale	Mono-attribut	Non	APB

2.5.1.2 La fragmentation horizontale

La fragmentation horizontale des données (FH) est une structure d'optimisation non-redondante qui cherche en premier lieu à minimiser le temps de réponse de la requête. La fragmentation des données est l'une des techniques les plus utilisées dans les bases de données aux niveaux logique, physique et déploiement. Contrairement à d'autres techniques de bases de données, la FH a été utilisée dans toutes les générations de bases de données. Elle permet aux méthodes d'accès telles que les tables, les indexes et les vues matérialisées d'être partitionnées en un ensemble disjoint de lignes qui sont stockées et accessibles séparément [13]. Cette technique présente une partie cruciale de la conception physique, et la plupart des systèmes de bases de données commerciaux d'aujourd'hui offrent un langage de définition de données natif pour définir les partitions horizontales d'une table comme la requête ci-dessous [5]. Cette dernière permet de partitionner la table T en deux partitions $p0$ et $p1$ par rapport aux valeurs 2021 et 2022 de l'attribut $Year$.

```
CREATE TABLE T
(
Year integer,
A varchar(15),
B varchar(15),
C  varchar(15)
)
PARTITION BY LIST (Year)
(
    PARTITION p0 VALUES  (2021),
    PARTITION p1 VALUES  (2022)
);
```

La fragmentation améliore la performance, la gestion et la disponibilité d'une grande variété d'applications et contribue à réduire le coût total du stockage de grandes quantités de données. La FH permet de subdiviser les tables en plus petits éléments dits fragments/partitions, ce qui permet de gérer et d'accéder à ces objets de base de données à un niveau de granularité plus fin [6]. Les performances des requêtes peuvent être considérablement améliorées dans certaines situations, en particulier lorsque la plupart des lignes fortement consultées de la table se trouvent dans une seule partition ou un petit nombre de partitions [7]. La FH divise une relation en sous-ensembles de tuples à l'aide de prédicats de requête comme illustrée dans la figure 2.13, par conséquent, elle réduit les coûts de traitement des requêtes en minimisant l'accès aux instances non pertinentes [111]. D'autres techniques d'optimisation (par exemple, index, vue matérialisée, partitionnement vertical) nécessitent des coûts de stockage et de maintenance supplémentaires, alors que la FH n'entraîne pas ces coûts, puisqu'elle ne réplique pas les données.

Deux principaux types de FH existent : la *fragmentation horizontale primaire* (mono-table) et la *fragmentation horizontale dérivée*. Dans la fragmentation horizontale primaire (FHP), une table est partitionnée en utilisant ses propres attributs via des prédicats simples. Dans la fragmentation horizontale dérivée (FHD), une table hérite des caractéristiques de partitionnement d'une autre table [24]. La FHD est bien adapté aux systèmes dirigés par les données comme les entrepôts de données relationnels puisque la table de faits est connectée à toutes les tables de dimension, par conséquent, FHD permet d'optimiser à la fois les opérations de jointure et de sélection.

Trois modes de distribution sont proposés pour contrôler la manière dont les données sont placées dans les partitions : *Range*, *Hash* et *List*. Chaque mode de distribution présente des

5. Nous avons utilisé la syntaxe d'Oracle
6. https://docs.oracle.com/database/121/VLDBG/GUID-EA7EF5CB-DD49-43AF-889A-F83AAC0D7D51.htm
7. http://postgresql.org/docs/current/ddl-partitioning.html

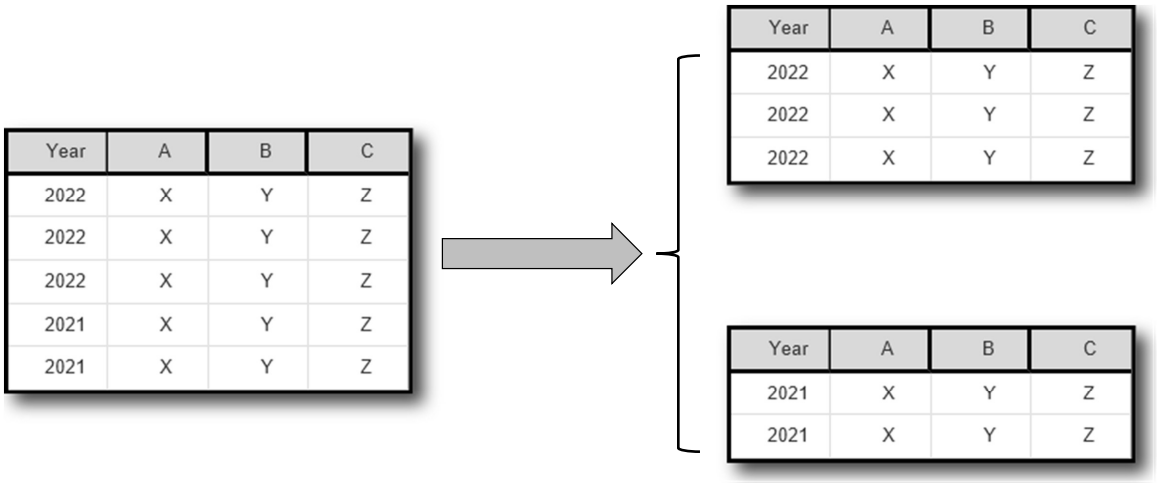


FIGURE 2.13 – La fragmentation horizontale

avantages différents et chacun est plus adapté à une situation particulière. Une table peut être partitionnée en tant que table partitionnée composite qui est une combinaison des modes de distribution de données, où la table est partitionnée par un mode de distribution, puis chaque partition est subdivisée en sous-partitions à l'aide d'un second mode de distribution de données.

Historiquement, le problème de sélection d'un schéma de fragmentation était piloté par les performances des requêtes, et il peut être formalisé comme suit : Soit : **(a)** un schéma de base de données avec une table jointe JT et des tables environnantes $\{ST_1, \ldots, ST_L\}$, **(b)** un ensemble de requêtes $Q = \{Q_1, Q_2, \ldots, Q_m\}$, constituant la charge de travail, et **(c)** un ensemble de contraintes telles que le nombre de partition et la taille maximale d'une partition. Le FH vise à trouver *un schéma de fragmentation* qui minimise le temps de réponse global à la requête et satisfait les contraintes fixées par rapport au schéma.

Pour une fragmentation horizontale valide d'une table T en n partitions, trois règles doivent être vérifiées [131] :

1. Complétude : toute instance de T doit être présente dans au moins une partition. La complétude garantit qu'aucune instance ne sera perdue après le partitionnement de la table.
2. Disjonction : toute instance appartenant à une partition ne doit pas appartenir à une autre partition.
3. Reconstruction : une fois, la fragmentation est effectuée, il doit toujours être possible de reconstruire la table d'origine à partir de ses partitions en utilisant l'opérateur d'union.

Le défi dans la résolution du problème de sélection d'un schéma de FH est de trouver quelles tables partitionner, sur quels attributs et sur quels prédicats des attributs choisis.

Le nombre de partitions m_i d'une table T dépend du nombre d'attributs partitionnés de T ainsi que du nombre de sous-domaines de ces attributs. Le nombre de partitions peut être calculé à l'aide de la formule suivante :

$$m_i = \prod_{j=1}^{r_i} n_{ij} \tag{2.5}$$

où r_i et n_{ij} représentent respectivement le nombre d'attributs partitionnés de T et le nombre de sous-domaines de l'attribut A_j de T. L'explosion du nombre de partitions est due à deux paramètres, le nombre de tables partitionnées et le nombre d'attributs partitionnés de chaque table.

Le problème de trouver le schéma de partitionnement optimal est prouvé comme un problème NP-Hard [13]. Par conséquent, de nombreux efforts académiques et industriels ont été consacrés à le résoudre. la FH a été largement étudiée, et son efficacité a été prouvée dans toutes les générations de bases de données (les bases de données relationnelles [2, 30, 131] les bases de données orientées objet [16], les entrepôts de données relationnels [15, 92] et les entrepôts de données XML [41, 111]) et toutes les architectures de déploiement (ex : centralisé, distribué [125] ou parallèle). Les performances s'améliorent rapidement lorsque la FH est appliquée, mais ont tendance à se dégrader lorsque le nombre de partitions augmente, un sur-partitionnement doit être détecté.

Une panoplie d'approches a été proposée dans la littérature pour résoudre le problème de sélection de la FH. De nombreux algorithmes et heuristiques ont été utilisés (PBA, ABA, modèle de coût, algorithmes de fouilles de données, algorithmes génétiques, etc.). Ces approches s'appuient sur des informations qualitatives et quantitatives définies sur les tables à partitionner. Les informations qualitatives fournissent les prédicats de sélection définis sur les tables, quant aux informations quantitatives, elles fournissent les facteurs de sélectivité des prédicats de sélection et les fréquences d'accès des requêtes.

Les approches basées sur les prédicats (PBA) sont simples, mais ont une grande complexité lorsque le nombre de prédicats est énorme. Pour n prédicats simples, cette approche génère 2^n minterms. Un *minterm* est une conjonction de plusieurs prédicats. Par conséquent, son utilisation est limitée à un nombre raisonnable de prédicats. L'idée de base des approches basées sur l'affinité (ABA) est que les prédicats simples ayant une haute affinité sont regroupés. Une affinité entre deux prédicats est la somme des fréquences de requêtes accédant simultanément à ces prédicats. Ces approches ne prennent en compte que les fréquences d'accès pour produire des partitions horizontales. Cependant, pour fournir une bonne fragmentation horizontale pour un schéma de base de données, plusieurs paramètres doivent être pris en compte, comme la taille des tables, les facteurs de sélectivité des prédicats, etc. Sauf que ces approches ne contrôlent pas non plus le nombre de partitions finals. Pour surmonter les limites des approches précédentes, une approche basée sur les coûts a été proposée pour évaluer le coût de chaque schéma de partitionnement généré. Un modèle mathématique de coût est proposé sur la base d'informations relatives à la BD, telles que la taille des tables, les facteurs de sélectivité, la taille du tampon, etc. Le modèle de coût estime le nombre d'opérations d'entrée/sortie requis pour exécuter la charge de travail sur une base de données partitionnée. Les techniques de fouille de données se soient déjà avérées extrêmement utiles pour sélectionner des structures d'optimisation qui améliorent la performance, ce qui a encouragé les chercheurs à en tirer profit et certains travaux ont adopté des algorithmes de fouille de données pour résoudre le problème de FH. Généralement, ces algorithmes contiennent deux phases : la collecte des métadonnées et des statistiques et la sélection du schéma de fragmentation. Dans le travail d'Agrawal et al. [2], les auteurs présentent des techniques de sélection de structures d'optimisation multiples qui permettent une solution évolutive pour résoudre les problèmes de fragmentation verticale et horizontale conjointement avec des indexes et des vues matérialisées pour améliorer les performances et la gestion. Curino et al. [40] ont présenté *The Schism*, qui a des stratégies de fragmentation de graphe qui augmentent l'évolutivité. Schism modélise la charge de travail sous forme d'un graphe et applique un algorithme de fragmentation de graphe *k-way min-cut* pour atténuer l'effet des transactions distribuées et augmenter le débit. La FH est utilisée pour améliorer l'évolutivité et la réplication pour la disponibilité des bases de données distribuées. Les partitions sont organisées en fonction du graphe qui contient des nœuds et des arêtes. Dans [172] les auteurs ont proposé une méthodologie basée sur l'optimisation par essaim de particules. Tout d'abord, ils calculent le coefficient d'attraction entre les prédicats à l'aide de l'indice de $Jaccard$, puis ils regroupent l'ensemble des prédicats et enfin, ils utilisent une optimisation par essaim de particules discrètes pour sélectionner le meilleur schéma de fragmentation. Dans [13], les auteurs proposent deux

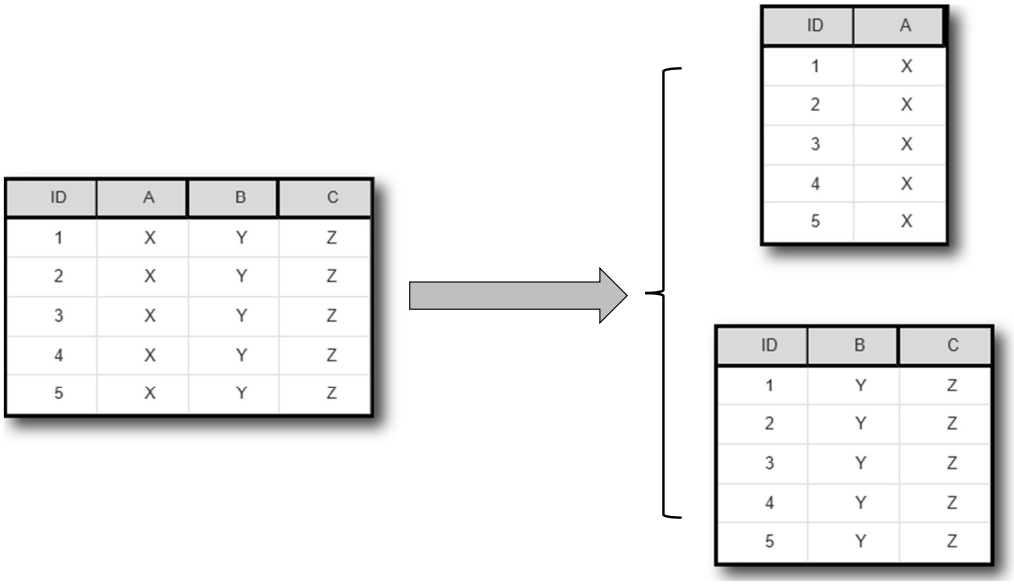


FIGURE 2.14 – La fragmentation verticale

algorithmes, Hill Climbing et Simulated Recuit. Ils ont également utilisé un modèle de coût pour évaluer la qualité du schéma de fragmentation généré. Leurs expérimentations montrent que l'algorithme de recuit simulé surpasse celui de Hill Climbing pour un seuil faible, alors que le Hill Climbing donne de meilleures performances pour des valeurs plus élevées du seuil. Mahboubi et al. [111] ont utilisé une approche de fouille de données, c'est-à-dire le clustering. Dans lequel ils utilisent l'algorithme K-MEANS dans un entrepôt de données XML où K est utilisé comme contrainte de maintenance. Bellaterche et al. [13] ont introduit une approche reposant sur un algorithme génétique. Pour modéliser leurs chromosomes, les auteurs utilisent une structure de données qui est une juxtaposition de tableaux représentant des attributs de sélection, où chaque tableau représente la décomposition du domaine d'un attribut, puis ils utilisent une fonction fitness basée sur un modèle de coût pour estimer le coût d'E/S pour chaque solution.

2.5.1.3 La fragmentation verticale

La fragmentation verticale (FV) divise une table en plusieurs tables contenant moins de colonnes. Il existe deux types de FV, la normalisation et le partitionnement des colonnes. La normalisation est un processus qui consiste à supprimer les colonnes redondantes d'une table et de les placer dans d'autres tables, en se basant sur les formes normales, liées à travers la relation de clé étrangère. Le partitionnement des colonnes divise la table d'origine verticalement en plusieurs autres tables plus petites liées à travers des relations de clé étrangère.

L'utilisation la plus courante de la fragmentation verticale consiste à réduire les coûts d'E/S et de performances associés à la récupération d'éléments fréquemment consultés. La figure 2.14 montre un exemple de partitionnement vertical. Dans cet exemple, des différentes colonnes de la table sont stockées dans différentes partitions. Une partition contient des données qui sont accédées plus fréquemment, notamment la colonne A, et une autre partition contient des colonnes moins fréquemment consultées. Grâce à la FV, les données sensibles peuvent être stockées dans une partition séparée avec des contrôles de sécurité supplémentaires, cette manœuvre peut également réduire le nombre d'accès simultanés.

La fragmentation des données peut être un mécanisme très efficace qui permet aux administrateurs et aux concepteurs de bases de données de gérer facilement les systèmes de plusieurs

téraoctets où la disponibilité des données devient une exigence clé. Ces partitions sont également utiles dans les environnements de traitement parallèle où par exemple le serveur S_1 peut fonctionner sur la partition 1, le serveur S_2 peut traiter la partition 2, etc. Ce qui augmente les performances de la base de données. La décomposition d'une très grande table en un ensemble de tables de taille réduite convient parfaitement aux bases de données orientées colonnes telles que HBase et Cassandra.

Malgré l'importance de la FV, il n'y a pas de syntaxe SQL spécifique pour créer des partitions verticales, il suffit de créer une nouvelle table avec des colonnes particulières et d'y copier les données. Ensuite, il sera possible de déposer les colonnes souhaitées dans la partition séparée. Et en cas de besoin d'accéder entièrement à la table initiale, il est possible de le faire en utilisant la jointure via clause JOIN entre les différentes partitions à travers la clé primaire.

La FV peut également causer des anomalies. L'utilisateur de la BD peut rencontrer des confusions lors de la récupération des différentes partitions en raison de la vitesse d'accès aux partitions différentes les unes des autres. Les anomalies de la troisième forme normale sont possibles telles que l'insertion, la suppression et la modification. D'autre part, et en raison de la réplication des données sur les partitions, plus d'espace de stockage est occupé en le comparant aux données stockées dans des fichiers normalisés. La modification des données affecte les données de plusieurs partitions, car elle prend plus de temps lorsqu'un seul fichier est utilisé.

2.5.1.4 Les vues matérialisées

Une vue logique ou bien une vue tout court peut être définie comme une requête nommée, et elle est dite matérialisée si son résultat est stocké physiquement, plus formellement, les vues sont définies comme suit :

Définition 2 (Une vue logique) *Une vue est une relation logique qui utilise la définition d'une requête pour extraire les données à partir des tables ou bien d'autres vues.*

Définition 3 (Une vue matérialisée) *Une vue matérialisée est une table sur disque qui contient les résultats d'une requête. Elle peut s'agir d'une copie locale de données située à distance, ou un sous-ensemble des lignes et/ou colonnes d'un résultat de requête ou de jointure avec ou sans une fonction d'agrégation.*

Les vues matérialisées (VM) réduisent le temps d'exécution des requêtes, en pré-calculant les opérations les plus coûteuses, comme la jointure et l'agrégation, et en stockant leurs résultats. Ce qui permet à certaines requêtes de s'exécuter et de retourner le résultat en accédant seulement aux vues matérialisées sans l'accès aux tables, ce qui rend l'exécution plus rapide. Dans les BD, les vues sont largement utilisées pour satisfaire plusieurs objectifs comme la sécurité, la confidentialité, l'intégrité référentielle, etc.

Généralement, les VM sont de taille réduite par rapport à la table réelle et elles ne sont constituées qu'à partir de données importantes, ce qui offre une exécution plus rapide des requêtes. Lorsque une requête est émise, les VMs sont accédées en premier lieu, si le résultat n'est pas trouvé dans les VM, dans ce cas les tables réelles du système seront recherchées. La requête ci-dessous permet de créer une vue matérialisée qui permet de retourner le prix total par pays dans un ordre décroissant.

```
CREATE MATERIALIZED VIEW resume_ventes AS
SELECT Country, SUM(Price) as Prix_Total
FROM Sales S, Customer C
WHERE S.CID = C.CID
GROUP BY Country
ORDER BY DESC Prix_Total;
```

L'utilisation des vues matérialisées est une technique courante pour réduire le temps de réponse des requêtes. En effet, matérialiser un ensemble approprié de vues et répondre aux requêtes à l'aide de ces vues peut considérablement accélérer le traitement des requêtes puisque l'accès aux vues matérialisées peut être beaucoup plus rapide que le recalcule des jointures. Par conséquent, la matérialisation de toutes les requêtes peut entraîner une réduction du coût de traitement, par contre, le coût de maintenance des vues sera très élevé, parce que les vues matérialisées doivent être maintenues et mises à jour afin de les garder cohérentes avec les données des sources.

En outre, le résultat de la requête peut être trop volumineux pour tenir dans l'espace mémoire disponible. Il est donc nécessaire de sélectionner un ensemble de vues à matérialiser en prenant en compte trois paramètres importants : le coût de traitement des requêtes, le coût de maintenance des vues et l'espace de stockage. Le problème de choisir les vues à matérialiser en gardant un certain équilibre entre les trois coûts est connu sous le nom de problème de sélection de vues matérialisées $PSVM$. C'est l'un des problèmes les plus difficiles dans les systèmes dirigés par les données et il est connu pour être un problème NP-complet [113].

Il n'est pas possible de matérialiser toutes les vues possibles, en raison des trois paramètres que nous avons précisés précédemment. Ainsi, le $PSVM$ peut être défini comme suit :

Soit :

— une charge de travail Q composée de n requêtes $Q = \{q_1, q_2, \ldots, q_n\}$;
— un ensemble de contraintes C, tel que le coût de stockage et/ou de maintenance ;
— un ensemble de besoins non fonctionnels B à l'exemple du coût de traitement, énergie consommée, voire coût de stockage et/ou de maintenance [8].

Le problème consiste à sélectionner un ensemble de vues à matérialiser en respectant les contraintes C et en satisfaisant l'ensemble B.

Dans la littérature, plusieurs travaux se sont consacrés à la résolution du $PSVM$, ces derniers se différencient par les structures de données utilisées pour la modélisation des VMs candidates, les algorithmes utilisés, le type de sélection, les contraintes à satisfaire et les besoins non-fonctionnels. Les structures de données sont généralement utilisées pour représenter l'espace de recherche, souvent large et en fonction de la charge de travail, des vues candidates à la matérialisation. Parmi ces structures, nous pouvons citer les treillis, les graphes et les multi view processing plan (MVPP). Une classification des différentes approches et des différentes méthodes utilisées est illustrée dans la figure 2.15.

Traditionnellement, la charge de travail en entrée considère seulement un ensemble fixe de requêtes. Une tendance plus réaliste considère une arrivée dynamique des requêtes, et par conséquent, la charge de travail est construite progressivement et change avec le temps. L'augmentation de la taille des VM, a incité la considération de l'espace de stockage et le coût de la maintenance comme des contraintes.

Une panoplie d'algorithmes a été proposée dans le cadre de la résolution du $PSVM$, inspirée de différents domaines et différents types comme les algorithmes déterministes et les algorithmes évolutionnaires. Une revue de l'état de l'art est présentée dans [113].

Comme toutes les générations des bases de données, le besoin non-fonctionnel principal est la performance traduite par la réduction du temps d'exécution. Et plus récemment, des travaux ont intégré la consommation d'énergie en tant que BnF additionnel [150]. Au fil des années, et à cause de la nécessité de la mis à jour (entière ou incrémentale) d'une manière régulière des VMs afin de les garder cohérentes, le coût de leur mise à jour peut être très élevé, ainsi, ce coût s'est imposé comme une contrainte.

Les tables de la BD évoluent et changent en fonction des mises à jour survenues, cependant, ces dernières ne sont pas reportés dans les VM, leurs contenus deviendront obsolètes et ne représenteront plus la réalité de la BD. Ainsi, la maintenance des VM permet de reporter les

8. Les coûts de stockage et/ou de maintenance peuvent être considérés comme contraintes ou comme BnF.

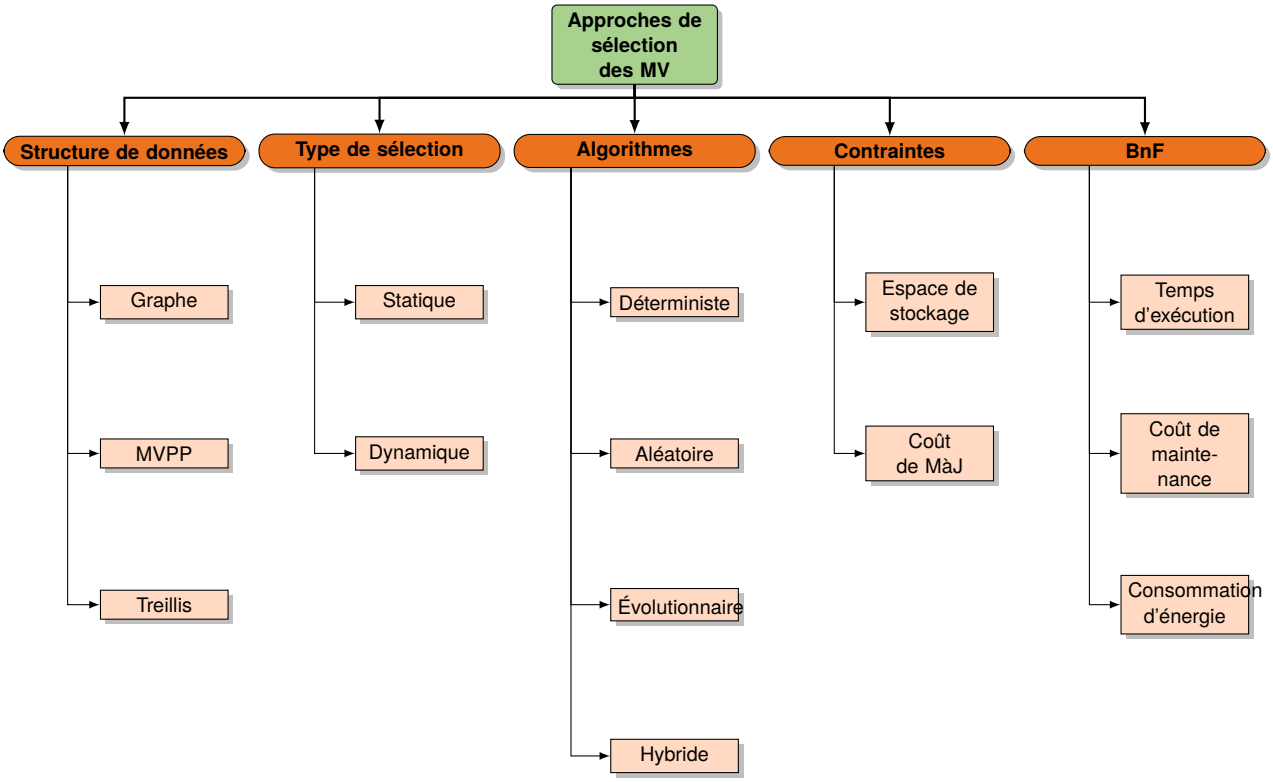


FIGURE 2.15 – Classification des approches de sélection des vues matérialisées

modifications effectuées sur les tables au niveau des vues. Cette manipulation peut se faire à travers trois méthodes : immédiate, périodique, et différée. Dans la première méthode, les VM sont mises à jour immédiatement à la fin de chaque transaction modifiant les tables sur lesquelles les VM sont définies. Dans la seconde, et comme son nom l'indique, les VM sont mises à jour à des périodes précises. Dans le mode différé, les VM ne sont mises à jour que lorsqu'elles sont utilisées par une requête.

Une fois, les VM sont sélectionnées, les requêtes de la charge de travail doivent être réécrites en fonction de ces VM afin d'en tirer profit. Le processus de réécriture de requêtes est supporté par la plupart des SGBD. Le problème de réécriture des requêtes a attiré l'attention des chercheurs grâce à sa relation avec plusieurs autres problèmes dans les bases données comme l'optimisation de requêtes et l'intégration des données, etc. La réécriture des requêtes est également considérée comme une technique d'optimisation pour réduire le coût d'évaluation d'une requête.

Formellement, nous pouvons définir le problème de réécriture des requêtes comme suit :
Soit
— une requête Q définie sur un schéma d'une base de données,
— un ensemble de vues matérialisée $\{V_1, V_2, \ldots, V_i\}$
Le problème de réécriture des requêtes consiste à répondre à la requête Q en utilisant les VM, et que le plan d'exécution soit le moins cher pour Q.

2.5.2 Le traitement parallèle

De nos jours, les entreprises doivent gérer une énorme quantité de données avec un taux de transfert très élevé. Pour de telles exigences, l'architecture client-serveur ou centralisée ne sont plus efficaces. Avec la nécessité d'améliorer l'efficacité du système, le concept de la base de

données parallèle entre en scène, qui cherche à améliorer les performances du système grâce au concept du traitement parallèle (parallélisation). Plusieurs ressources telles que les processeurs et les disques sont utilisées en parallèle. Les opérations sont effectuées simultanément, par opposition au traitement séquentiel, un serveur parallèle peut permettre l'accès à une seule base de données par des utilisateurs sur plusieurs machines. Il également possible d'effectuer plusieurs opérations en parallèle telles que le chargement des données, le traitement des requêtes, la création d'index et l'évaluation des requêtes. En connectant les différentes ressources telles que le processeur et les disques en parallèle, la performance peut augmenter considérablement et offre la possibilité d'exécuter les tâches en moins de temps. Dans les bases de données réparties, les nœuds ont moins de contacts les uns avec les autres, de sorte que la défaillance d'un nœud n'entraîne pas la défaillance de l'ensemble du système. Cela équivaut à une disponibilité de la base de données nettement plus élevée.

Le traitement parallèle permet à de nombreux processus de travailler simultanément dans le but d'exécuter entre eux une partie de la requête, et ce, à travers une répartition du travail nécessaire entre plusieurs unités de traitement ce qui permet au moteur d'exécution de traiter la requête plus rapidement que sur une seule unité de traitement. Ainsi, les opérations de base de données, qui prennent souvent beaucoup de temps et qui impliquent une grande quantité de données, peuvent être parallélisées si le matériel le permet.

La répartition des données est l'étape la plus importante du traitement parallèle. Une répartition uniforme des données permet de répartir les accès sur les différents disques, et d'optimiser ainsi le temps d'accès. La génération d'un plan parallèle doit être à la fois efficace, correcte et elle permet d'inclure des opérations de contrôle pour la synchronisation des accès aux données et la détection des fins des opérations.

2.5.3 La dé-normalisation

La normalisation des tables est effectuée pour supprimer la redondance des données afin de maintenir l'intégrité de la base de données et d'éliminer les données redondantes à travers les formes normales (1FN, 2FN, 3FN, FNBC, 4FN). La normalisation d'une *BD* fait partie d'un processus de conception de bases de données efficace dans lequel une table est décomposée en deux tables ou plus dans le but d'éviter les anomalies. Une base de données normalisée peut éviter diverses anomalies, cependant, cela peut avoir un impact négatif sur les performances de la base de données. Le temps d'exécution des requêtes sur la base de données normalisée est considérablement élevé en raison des multiples jointures entre les tables. Les tables normalisées peuvent conduire à un traitement inefficace des données.

Pour améliorer les performances de la base de données, la dé-normalisation des tables peut être délibérément effectuée lorsque les tables sont transformées en une forme non normalisée. La dé-normalisation consiste à un processus de regroupement de plusieurs tables liées par des références, en une seule table, à travers la réalisation statique des opérations de jointure adéquates. La dé-normalisation permet d'améliorer les performances de la BD lors de la recherche sur les tables grâce à l'implémentation des jointures à la place de les calculer. Ceci peut être réalisé en ajoutant des colonnes redondantes aux tables. Des colonnes dérivées peuvent également être ajoutées pour dé-normaliser une base de données, par exemple, le cas de la transformation du schéma en flocon de neige en schéma en étoile dans le cas des entrepôts de données.

2.5.4 L'optimisation des jointures

L'une des étapes les plus cruciales et les plus coûteuses lors de l'exécution des requêtes est l'optimisation des requêtes. La tâche de l'optimiseur consiste à trouver une stratégie d'exécution optimale. Si ces requêtes nécessitent une opération de jointure entre plusieurs tables, l'ordre de jointure aura un impact significatif sur le coût d'exécution. Rappelons que la jointure est

l'opération qui permet de relier deux tables sur la base de propriétés partagées. Dans les requêtes comportant 6 tables ou moins, le meilleur ordre peut être trouvé rapidement à travers une recherche exhaustive de l'espace de recherche, par contre, il est difficile de trouver le meilleur plan lorsqu'il y a plus de 8 relations. Dans le cas des bases de données relationnelles typiques, le nombre de tables jointes des requêtes ne dépasse pas les 10 tables. Pour trouver l'ordre de jointure dans ces environnements, des approches comme la programmation dynamique ont été adoptées. Sauf que, dans des systèmes comme les systèmes d'aide à la décision, l'opérateur de jointure peut inclure plus de 100 tables [82].

La sélection d'un ordre de jointure efficace, c'est-à-dire un ordre permettant d'évaluer efficacement les prédicats de jointure d'une requête donnée, est un sujet crucial dans l'optimisation des requêtes relationnelles. Lorsque nous joignions trois tables A, B, et C de 10, 10000 et 1000000 de lignes (respectivement), par exemple, un plan de requête qui joint B et C en premier peut avoir un coût supérieur à celui qui joint A et C en premier. Au fur et à mesure que la taille des données augmente et que le nombre de relations participantes à l'opération de jointure augmente, le nombre de plans d'exécution disponibles augmente, ce qui empêche les approches traditionnelles de fournir de solution satisfaisante à ce problème qui est considéré communément un problème NP-Hard [82].

Le problème d'optimisation d'ordre de jointure prend en entrée un graphe de requête (graphe de jointure) qui contient toutes les tables participantes dans la jointure, et les liens entre les tables sont appelés nœuds du graphe. L'espace de recherche, également connu sous le nom d'espace de solution, est une collection de stratégies qui fournissent les mêmes résultats pour un problème donné. Les feuilles et les nœuds intérieurs de l'arbre de traitement sont respectivement les tables de la BD et les opérateurs de jointure. Le transfert des données des nœuds à la racine est déterminé par les arêtes. Bien qu'il soit théoriquement possible de générer plusieurs plans d'exécution de requête optimaux, les optimiseurs ne fournissent souvent qu'un seul plan d'exécution efficace et acceptable [82].

Outre l'ordre de jointure, l'implémentation de l'algorithme de jointure est un autre facteur qui a un impact significatif sur le coût de l'exécution finale. En général, les SGBD supportent une variété d'implémentations des algorithmes de l'opération de la jointure parmi lesquelles l'optimiseur de requêtes peut choisir. Chaque algorithme de jointure a des caractéristiques spécifiques qui le rendent plus ou moins approprié pour une requête donnée et un environnement d'exécution donné. Il existe trois algorithmes pour calculer l'opération de jointure : Nested loop join, Hash join et Sort Merge Join [161].

Dans l'algorithme Nested Loop (boucles imbriquées), chaque tuple de la relation est comparé avec tous les n-uplets de l'autre relation, et à la fin, seuls les pairs de tuples qui satisfont la condition sont ajoutés au résultat de la jointure [161].

La jointure par hachage est un type de jointure qui utilise une table de hachage pour identifier les lignes correspondantes entre deux entrées (une entrée est une ou plusieurs tables). Ce type de jointure est généralement plus efficace que les jointures en boucle imbriquées, surtout si l'une des entrées est suffisamment petite pour tenir dans la mémoire. Selon la littérature, cet algorithme est généralement divisé en deux phases : la phase de construction et la phase d'exploration [161].

La jointure par tri-fusion est un algorithme de jointure courant dans les systèmes de bases de données utilisant le tri. Le prédicat de jointure doit être un prédicat d'égalité. L'algorithme trie les deux relations en fonction de l'attribut de jointure, puis fusionne les relations triées en les analysant séquentiellement et en recherchant les tuples correspondants.

L'efficacité des structures d'optimisation provient de leur capacité de pré-calculer les jointures et de stocker leurs résultats comme le cas des indexes de jointures binaires et les vues matérialisées, ou bien, de réduire la taille des tables jointes, les tuples à joindre plus précisément, dans le cas de la fragmentation horizontale.

2.5.5 L'ordonnancement des requêtes

Principalement, la technique d'ordonnancement a été étudiée et employée par les entreprises dans le contexte d'optimisation pour établir les emplois du temps et d'autres planifications des tâches. Ensuite, elle a été intégrée dans les systèmes d'exploitation pour la gestion des processus sous la forme d'une composante dite *ordonnanceur*. Le rôle de l'ordonnanceur est de définir l'ordre de l'exécution des processus et de s'assurer de l'exécution tout en optimisant les ressources physiques [11]. Les SGBD, comme DB2 et Oracle, ont tiré profit de cette technique et ils l'ont adopté à deux niveaux : (1) dans l'ordonnanceur interne du SGBD, afin de permettre l'ordonnancement des processus dans le but de s'assurer qu'ils soient tous exécutés en un temps fini, (2) dans l'ordonnanceur de requêtes, et dans ce cas l'ordonnancement des requêtes (OR) est considéré comme une technique d'optimisation non redondante qui vise à réduire le temps d'exécution des requêtes [108].

Dans le contexte de bases de données, il existe une forte interaction entre les requêtes, et qui est très fréquente. Cette interaction peut être utile en exploitant les sous-expressions communes entre les requêtes afin de réduire le nombre d'accès au disque. Si la taille des résultats de ces sous-expressions est réduite, ils peuvent être stockés en mémoire principale de façon temporaire sous forme d'objets du buffer, ou bien, ils peuvent être stockés sur le disque sous la forme de vues matérialisées, si leur taille est considérable. Ainsi, le rôle de l'ordonnanceur est de choisir la meilleure planification pour l'exécution des requêtes en maximisant l'utilisation des résultats pré-calculés existants. Il permet également de traiter la concurrence des utilisateurs et des transactions en assurant l'accès aux données sans perte de temps.

L'OR est souvent corrélé avec d'autres techniques d'optimisation telle que la gestion du buffer et les vues matérialisées. Phan et al. [136] ont montré que l'ordonnancement n'est pas utile si les données ne subissent aucun remplacement. Autrement dit, si les mêmes données résident sur le support de stockage durant l'exécution des requêtes, l'ordre d'exécution n'a pas un impact sur la performance. Les auteurs ont proposé une approche qui permet de créer des vues matérialisées d'une manière dynamique. Un algorithme génétique est utilisé pour la sélection des vues matérialisées ce qui a permis d'augmenter la performance. Cette approche a montré l'importance de l'interaction entre les supports de stockage, les requêtes et les différentes techniques d'optimisation.

2.6 Conclusion

Les technologies de l'information se développent à un rythme très élevé. Récemment, de nouveaux types d'exigences en matière de capacité de traitement des bases de données ont émergé, et d'autre part, une variété de techniques d'optimisation sophistiquées ont été proposées. Ce qui a permis aux bases de données de devenir un concept qui évolue et qui change progressivement au fil des années depuis que le terme a été inventé. Ces évolutions ont rendu possible la mise en œuvre des BD plus performante et plus scalable à travers l'exploitation des nouvelles technologies matérielles et logicielles.

Dans le but d'intégrer l'aspect énergétique dans une base de données, il est primordiale, en premier lieu, de l'étudier afin d'identifier les composantes ayant un impact sur la consommation énergétique du système. Ainsi, ce chapitre a été dédié à la présentation du cycle de vie d'une base de données et il présente un survol historique des différentes techniques d'optimisation. Ces dernières se distinguent et dépendent du contexte de leur utilisation, le types des requêtes utilisées et la configuration matérielle. Pour chaque structure d'optimisation présentée, les méthodes de sélection et les algorithmes proposés sont détaillés.

Dans le chapitre suivant, nous allons présenter les notions de base relatives à l'énergie, et nous donnerons un état de l'art des travaux dédiés à l'optimisation de la consommation énergétique dans les systèmes d'informations en général et dans les bases de données en particulier.

Chapitre 3

La gestion de l'énergie dans les bases de données

3.1 Introduction

Les centres de données consomment plus de 416 térawatts d'électricité par an, ce qui représente près de 3% de l'électricité totale générée à l'échelle mondiale. Selon Energy Technologies Area (ETA) [1], plus de 73 milliards de kWh ont été consommés par les États-Unis en 2020 [160]. Cette surconsommation a mis en évidence la nécessité de minimiser la consommation d'énergie. Face à cette situation, la consommation d'énergie des logiciels et du matériel informatique est devenue un problème émergeant pour les développeurs, les chercheurs et les industriels. Elle est devenue un domaine très attractif soulevant plusieurs défis, et ce, dû à la complexité des systèmes informatiques et ses différents composants, d'où l'énergie doit être optimisée afin d'utiliser efficacement les logiciels sans aucun gaspillage. Ainsi, les développeurs ont essayé de produire des logiciels économes en énergie en supprimant les fonctionnalités optionnelles, en limitant les capacités des logiciels, en produisant des logiciels plus petits et centrés sur les fonctionnalités ou en réduisant l'utilisation de leurs applications. L'émergence des concepts tels que les systèmes éco-énergétiques, a transformé l'objectif des concepteurs des systèmes informatiques de la performance à la puissance électrique et l'efficacité énergétique.

Pour identifier les défis ouverts dans ce domaine et faciliter des futurs progrès, il est essentiel de synthétiser et de classer les travaux de recherche dédiés à la conception éco-énergétique à ce jour. Dans ce chapitre, nous présentons une revue de l'état de l'art des différentes techniques et moyens utilisés qui ont pour objectif la réduction de la consommation d'énergie dans les systèmes informatiques en général et dans le domaine de bases de données en particulier. Ce chapitre commence par la définition des notions de base relatives à l'énergie ainsi que les différentes méthodes pour la mesurer. Le traitement de l'aspect énergétique dans les systèmes informatiques ainsi que son intégration dans tous ses niveaux sont discutés dans la deuxième section en présentant une taxonomie des travaux sur la conception économe en énergie des systèmes informatiques couvrant les niveaux : matériels, systèmes d'exploitation, et applications. Ensuite, nous détaillons les approches logicielles et matérielles permettant de réduire l'énergie dans les bases de données.

3.2 Préliminaires

Avant de détailler les travaux existants sur l'optimisation de la consommation énergétique, et pour mieux appréhender les problèmes liés à ce sujet, nous commençons, dans cette section, par la présentation de quelques définitions et par l'introduction de quelques notions fondamentales sur l'énergie. Ensuite, nous détaillons les outils matériels et logiciels permettant de mesurer/estimer l'énergie.

3.2.1 Définitions

Bien que les notions de l'énergie et la puissance sont complètement différentes, et chacune est utilisée pour quantifier une grandeur bien déterminée, elles sont souvent utilisées d'une manière interchangeable pour identifier la même grandeur.

Le mot *énergie* est dérivé du mot grec **energeia** qui signifie *la force en action*. Ce terme vient d'une approche physique du problème, qui décrit l'énergie comme :

Définition 4 (Énergie) *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 [149].*

1. https://eta.lbl.gov/

D'où, l'énergie (E), est la quantité totale de travail effectuée par un système sur une période de temps, et elle est mesurée en *Joules* (J) et elle peut être définie formellement comme suit :

$$E = P \times t \tag{3.1}$$

où P, t et E représentent respectivement la puissance, la période et l'énergie consommée.

L'énergie peut être présente sous diverses formes telles que l'énergie électrique, l'énergie mécanique, l'énergie de la lumière, l'énergie sonore, l'énergie thermique, l'énergie magnétique, et l'énergie nucléaire. En ce qui suit, nous nous focalisons seulement sur l'énergie électrique.

La puissance reflète la vitesse à laquelle un travail est fourni ou la dérivée du travail au fil du temps, plus précisément :

Définition 5 (Puissance électrique) *La puissance électrique est la quantité d'énergie consommée par le système par unité de temps, ou le rythme de faire un travail.*

La puissance est mesurée en *Watts* (W) et formellement, elle est définie comme suit :

$$P = \frac{W}{t} \tag{3.2}$$

où W, t et P représentent respectivement le travail total effectué, la période et la puissance.

Les concepts de l'énergie, puissance, et travail sont utilisés d'une manière différente qui correspond à son contexte d'utilisation. Par exemple, dans le contexte informatique, la notion de l'énergie représente l'énergie électrique totale consommée par l'ordinateur au fil du temps, la puissance désigne le taux avec lequel un ordinateur consomme de l'énergie électrique (ou la dissipe sous forme de chaleur), et le travail implique les activités et les instructions associées à l'exécution des applications/programmes (addition, soustraction ou opération dans la mémoire, etc.) [177].

Le temps écoulé et l'énergie consommée par un programme lors de son exécution nous permettent de calculer la puissance moyenne de cette exécution, qui est le ratio de l'énergie par le temps. Malgré cette relation, la réduction de la puissance n'implique pas nécessairement une réduction de l'énergie totale consommée, plus précisément, lorsque cette réduction de puissance nécessite une augmentation du temps d'exécution pour terminer le traitement demandé.

En électronique, pour un système donnée, il existe deux catégories de puissance électrique : (a) la puissance base (aussi appelée statique ou passive) et (b) la puissance active (aussi appelée puissance dynamique) [59].

1. **La puissance de base :** est la puissance consommée lorsque le système est en état d'inactivité, c'est-à-dire, aucun programme n'est en cours d'exécution. C'est la puissance minimale nécessaire pour le fonctionnement des différents composants matériels de l'ordinateur (processeur, mémoires, périphériques d'E/S, ventilateurs, et les autres composants électroniques de la carte mère) lorsqu'ils sont en un état d'inactivité.
2. **La puissance active :** est la puissance requise pour l'ordinateur lors de l'exécution d'un programme. C'est la puissance consommée par tous les composants intervenant dans l'exécution d'un programme. Cette puissance dépend fortement de la tâche à exécuter et les composants nécessaires.

Lors de l'évaluation de la consommation d'énergie dans un système, il faut tenir en compte deux concepts de puissance électrique : (*i*) la puissance moyenne qui représente la moyenne de la puissance consommée durant d'exécution, et (*ii*) la puissance maximale (*peak power*) qui représente la valeur maximale de la puissance atteinte durant l'exécution. Tout au long de cette thèse, nous utilisons la puissance moyenne pour les mesures de la puissance.

Lorsque nous parlons de la réduction de la consommation d'énergie, nous utilisons la notion de l'*efficacité énergétique* (EE). EE est définie par le ratio de la performance, mesurée en taux de

travail effectué (TE) (dans le contexte des bases de données nous parlons de requêtes exécutées) par la quantité d'énergie utilisée [63].

Définition 6 (Efficacité énergétique) *L'efficacité énergétique est la consommation d'énergie d'un système par rapport au service rendu.*

$$EE = \frac{TE}{E} = \frac{TE}{P \times t} = \frac{t}{P} \tag{3.3}$$

où TE, t, E et P désignent le travail effectué, le temps, l'énergie et la puissance respectivement.

Il existe deux manières pour optimiser l'efficacité énergétique, soit de réduire la consommation énergétique pour réaliser le même taux de travail demandé, ou bien, par l'utilisation de la même quantité d'énergie pour effectuer plus de travail [65].

3.2.2 Les outils de mesure

Selon le consultant en management d'entreprise *Peter Drucker*, *"If you can't measure it, you can't improve it"*. Ainsi, afin de réduire l'énergie à n'importe quel niveau d'un système tout en offrant une précision minimale, il est nécessaire de mesurer l'énergie disponible et consommée. En particulier, pour une gestion de l'énergie à un niveau supérieur, surveiller et estimer la consommation d'énergie des ressources matérielles et logicielles est une condition **sine qua non**. Cependant, la définition de l'énergie dans les systèmes informatiques est basique et très générale, bien que l'énergie dépend du logiciel, du matériel, et du contexte d'exécution.

L'estimation de la consommation d'énergie d'un système est souvent réalisée soit, en modélisant l'utilisation des ressources pour obtenir des informations sur l'énergie, soit, à travers des équipements (wattmètre/capteurs) adéquats. Ainsi, une première étape d'optimisation de la consommation d'énergie consiste à la mesurer avec précision. Dans cette section, nous décrivons les outils matériels et logiciels permettant de mesurer l'énergie d'un système.

3.2.2.1 Les outils matériels

L'évaluation de la consommation d'énergie des composants matériels nécessite généralement un investissement matériel, comme un multimètre ou un circuit intégré spécialisé (capteur).

Pour mesurer l'énergie, les multimètre/wattmètre présentent une solution fiable et précise. Un wattmètre, comme son nom l'indique, est un instrument qui permet de mesurer la consommation électrique d'un appareil (récepteur), mais aussi la puissance émise par un courant électrique de la source (émetteur). Cette puissance sera traduite par la suite en $watts$. Le wattmètre est souvent branché entre la source d'électricité et l'appareil à mesurer. Sur le marché, il existe une multitude de modèles de wattmètre : les analyseurs de précision de puissance (LMG) [2], Brand Electronics 20-1850 CI [3], CHROMA 66200 [4], WT1800E [5], PA3000 [6], PowerSpy 2 [7]. Ces multimètres permettent de mesurer l'énergie d'un système, où chaque wattmètre se caractérise par son taux d'erreur, la puissance maximale et minimale mesurée, sa vitesse d'échantillonnage, sa fréquence, le nombre de grandeurs mesurées, et la mémoire pour le stockage des valeurs. Dans ce qui suit, nous allons présenter deux wattmètres (i,e. WattsUp ?Pro et Yocto-Watt) que nous avons utilisés pour les études expérimentales que nous avons réalisées durant cette thèse.

2. https://www.zes.com/en/Products/Precision-Power-Analyzers?gclid=EAIaIQobChMIiYj-4t-B9gIViOd3Ch39Hg-ZEAAYASAABwE
3. https://www.brandelectronics.com/meters.html?ref=GoogleYuz.Com
4. https://www.chromaate.com/en/test_solutions/power_electronics_test_solution
5. https://tmi.yokogawa.com/solutions/products/power-analyzers/wt1800e-high-performance-power-analyzer/
6. https://www.tek.com/fr/products/power-analyzers/pa3000
7. https://www.alciom.com/nos-metiers/produits/powerspy2/

FIGURE 3.1 – Le wattmètre WattsUp ? Pro ES

Watts'up ?Pro [8] est un appareil permettant de mesurer dix-huit grandeurs différentes, dont : la puissance actuelle, la puissance minimale, la puissance maximale achevée, le facteur de puissance, wattheures cumulés, Kilowattheures (Kwh) mensuel moyen, temps écoulé, la fréquence, le coût mensuel moyen en dollars, la tension, volts minimum, volts maximum, ampères actuels, ampères minimum et ampères maximum, etc. Avec son écran numérique, comme le montre la figure 3.1, il permet de visualiser, en temps réel, la consommation de l'équipement branché. Watts'up ?Pro offre une précision de 1.5%, par contre, en dessous de 60 $watts$, le courant et la puissance diminuent en précision. La puissance minimale mesurable est $0,5$ $watts$. Il dispose d'une mémoire interne qui peut être utilisée pour enregistrer des données. Il peut enregistrer pendant 8000 secondes avec toutes les grandeurs mesurées, ou pendant 120000 secondes, avec seulement la puissance enregistrée. Une application est fournie permettant d'enregistrer toutes les mesures effectuées par $Watts'up?Pro$ sur un ordinateur, via un câble USB.

Yocto-watt [9] comme illustré dans la figure 3.2, est un module électronique permettant de mesurer à travers un câble USB la puissance consommée par un appareil électrique. Il permet de mesurer la tension entre deux câbles électriques ainsi que le courant passant dans un câble. Il est aussi capable de mesurer la consommation électrique à un instant t (en Watt) d'un appareil pour calculer sa consommation horaire en $Wattheure$.

Yocto-Watt peut fonctionner avec des tensions continues ou bien des tensions alternatives. Pour cette dernière, il est capable de calculer le facteur de charge qui est le décalage tension/courant.

Il offre des mesures avec une précision de $0.002W/1.5\%$ pour le courant continu et de 0.02 $W/1.5\%$ pour le courant alternatif. Une caractéristique importante de ce module, est d'être un wattmètre isolé, c'est-à-dire, que la partie mesure est électriquement isolée de la partie USB. Ce qui permet de mesurer la consommation de charges électriques qui ne partagent pas la même référence de terre que le bus USB de l'ordinateur. Lorsqu'il est monté dans un boîtier protecteur adéquat, il peut être utilisé pour mesurer la puissance d'appareils connectés directement sur le secteur. Une application de visualisation est également fournie permettant de visualiser en temps réel la consommation électrique.

8. http://www.wattsupmeters.com
9. https://www.yoctopuce.com/EN/products/usb-electrical-sensors/yocto-watt

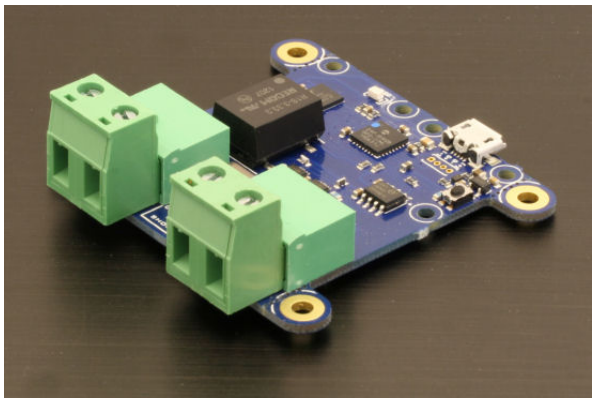


FIGURE 3.2 – Le wattmètre Yocto-watt

3.2.2.2 Les approches et les outils logiciels

Les chercheurs et les industriels se sont intéressés par la réduction de l'énergie consommée par un système informatique. Cependant, comprendre le comportement énergétique d'un composant matériel ou bien logiciel présente la première étape pour améliorer sa consommation énergétique. Ainsi, de nombreux travaux ont été introduits pour estimer et mesurer la consommation énergétique des logiciels. Cette estimation peut être réalisée en modélisant l'utilisation des ressources matérielles. Dans cette section, nous présentons les principales approches et outils de mesure de l'énergie introduits dans la littérature.

Pour mesurer la consommation énergétique des applications, une méthodologie d'estimation basée sur le logiciel est adoptée. Cette approche se réalise comme suit : la consommation des ressources matérielles est mesurée avant d'être mappée aux logiciels via des modèles énergétiques. Concrètement, cette consommation des ressources matérielles est mesurée, soit directement depuis les appareils, soit via les primitives du système d'exploitation. Ces informations sont ensuite exploitées dans des modèles énergétiques définis afin d'estimer la consommation énergétique des logiciels. Cette méthodologie est composée de quatre étapes qui sont résumées dans la figure 3.3.

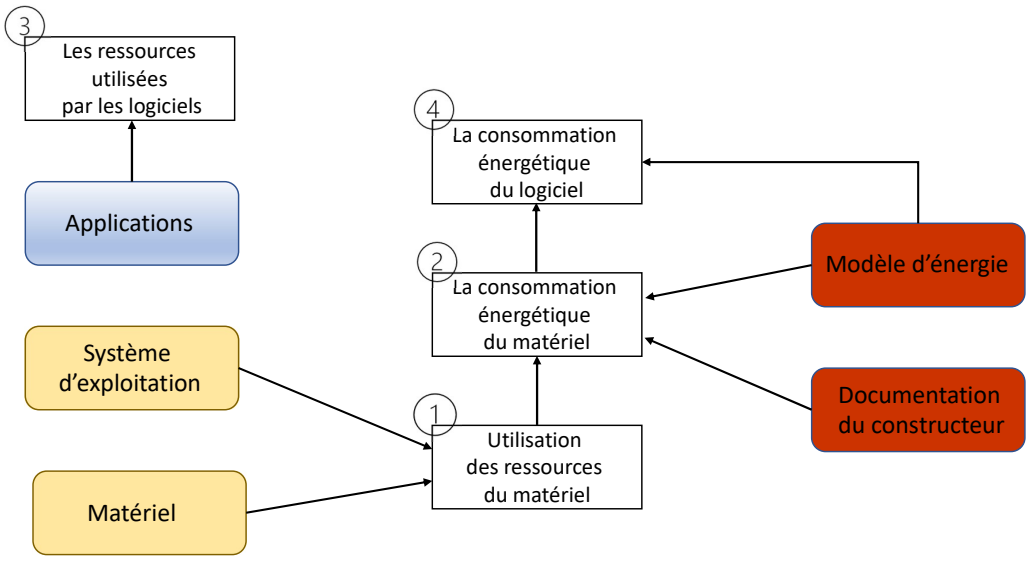


FIGURE 3.3 – La méthodologie d'estimation basée sur un logiciel

Seo et al. [157] ont proposé une approche basée sur l'estimation qui calcule la consommation d'énergie d'un composant logiciel en utilisant la formule de l'équation 3.4 :

$$E_{Component} = E_{Computational} + E_{Communication} + E_{Infrastructure} \tag{3.4}$$

Où $E_{Computational}$ est l'estimation énergétique de l'utilisation d'une ressource matérielle (le traitement du processeur, l'accès à la mémoire, les requêtes exécutées par le disque). $E_{Communication}$ est l'estimation énergétique des données transférées sur le réseau et $E_{Infrastructure}$ représente l'estimation énergétique de la machine virtuelle. L'avantage de cette approche est qu'elle permet d'estimer la consommation d'énergie des applications Java s'exécutant dans des machines virtuelles.

Kansal et al. [88] ont également proposé une approche pour calculer la consommation énergétique d'une application. La principale différence entre leur approche et celle de Seo et al. [157] est qu'elle permet d'estimer la consommation d'énergie même lorsque l'application est en état d'attente et d'inactivité. Les applications consomment de l'énergie lorsqu'elles attendent des requêtes du disque. La consommation énergétique totale d'une application est donnée par l'équation 3.5 [128].

$$E_{App} = E_{Active} + E_{Wait} + E_{Idle} \tag{3.5}$$

où E_{Active} représente l'énergie dépensée par l'application lorsqu'elle est en cours d'exécution. E_{Wait} est le coût énergétique de l'application à l'état d'attente et E_{Idle} représente le coût énergétique de l'application à l'état inactif.

Trefethen et al. [174] ont introduit une approche basée sur des capteurs. Les capteurs sont utilisés entre l'application sur laquelle les mesures sont effectuées et la source d'alimentation. Les capteurs collectent des données précises à partir de la source d'alimentation. Les données collectées sont stockées dans un serveur de collecte de données. Le fonctionnement des capteurs n'est pas toujours continu mais périodique sur des intervalles de temps. Après avoir exécuté le logiciel, les parties lu des données des capteurs et le logiciel sont corrélées. L'énergie consommée par l'application est donnée par l'équation 3.6.

$$E = \int_T P_w dt - P_I T \tag{3.6}$$

où P_w représente le profil de puissance en une fraction de seconde. P_I est le profil d'inactivité du logiciel et T représente le temps d'exécution du logiciel en cours d'exécution.

Parmi les outils développés pour l'estimation et le profilage de l'énergie, nous pouvons citer :

pTop est un outil de profilage de puissance au niveau des processus [47]. Similaire à l'utilitaire Top de GNU/Linux qui fournit l'usage des ressources pour chaque processus, pTop fournit la consommation d'énergie (en Joules) des processus en cours d'exécution. Pour chaque processus, il donne la consommation électrique du CPU, de l'interface réseau, de la mémoire de l'ordinateur et du disque dur. L'outil consiste en un démon s'exécutant dans le noyau et estimant en continu l'utilisation des ressources de chaque processus. Il obtient ces informations en accédant au répertoire $/proc$. Pour le processeur, il utilise également la puissance thermique de conception (Thermal Design Power TDP), qui est la quantité maximale de puissance que le système de refroidissement doit dissiper, fournie par les constructeurs dans les calculs de consommation d'énergie. Il calcule ensuite la quantité d'énergie consommée par chaque application durant un intervalle de temps t. Il se compose également d'un utilitaire d'affichage similaire à l'utilitaire Top de Linux. Une version Windows est également disponible, appelée $pTopW$, et propose des fonctionnalités similaires, mais en utilisant des API Windows. Le modèle énergétique de $pTop$ est une somme

de l'énergie consommée par les ressources individuelles en plus de l'énergie consommée par l'interaction de ces ressources. Par conséquent, la consommation énergétique d'une application peut être calculée selon la formule suivante :

$$E_{appi} = \sum U_{ij} \times E_{ressourcej} \times E_{interaction} \tag{3.7}$$

où U_{ij} est l'utilisation de l'application i sur la ressource j, $E_{ressourcej}$ est la quantité d'énergie consommée par la ressource j, et $E_{interaction}$ est la quantité indirecte d'énergie consommée par l'application en raison de l'interaction entre les ressources système. Les auteurs ont proposé également un modèle général pour la consommation d'énergie d'une ressource particulière (CPU, disque, interface réseau).

PowerScope permet de profiler la consommation énergétique des applications [116]. Cet outil utilise un multimètre numérique pour échantillonner la consommation d'énergie et un ordinateur séparé pour contrôler le multimètre et stocker les données collectées. $PowerScope$ peut échantillonner la consommation d'énergie par processus. Cet échantillonnage est plus précis que l'estimation d'énergie, bien qu'il nécessite encore un investissement matériel. En particulier, $PowerScope$ mappe la consommation d'énergie sur la structure du programme. Il peut donc déterminer l'énergie consommée par un processus spécifique, et même jusqu'à la consommation d'énergie des différentes fonctions au sein du processus. La mise en œuvre de l'outil utilise un échantillonnage statistique de la consommation d'énergie et de l'activité du système. L'outil génère un profil énergétique qui est ensuite analysé hors ligne. Ainsi, l'outil n'a pas de surcharge de profilage, mais il ne donne aucune valeur en ligne. Lors de l'échantillonnage, un multimètre est utilisé pour échantillonner le courant consommé par l'ordinateur profilé. Un ordinateur séparé est également utilisé pour stocker les informations collectées et contrôler le multimètre (bien que cela puisse également être fait sur le même ordinateur). $PowerScope$ utilise trois composants logiciels :

1. un moniteur système qui échantillonne l'activité du système en utilisant un démon au niveau de l'utilisateur et les modifications du noyau du système d'exploitation. Le moniteur enregistre la valeur du compteur de programme (PC) et l'identificateur de processus (PID). Il enregistre également, via l'instrumentation, des informations systèmes supplémentaires tels que le chemin d'accès associé à l'exécution des processus ou le chargement des bibliothèques partagées.
2. un moniteur d'énergie qui s'exécute sur une machine distincte et collecte les échantillons de courant du multimètre. Ce dernier transmet de manière asynchrone les échantillons en cours au moniteur où ils seront stockés.
3. un analyseur d'énergie qui utilise les données collectées par le moniteur de système et le moniteur d'énergie pour générer le profil énergétique de l'activité du système. La consommation d'énergie est calculée à l'aide de la formule de l'équation 3.8, et l'analyseur génère ensuite un résumé de la consommation d'énergie par processus. Le processus d'analyse est effectué hors ligne après l'exécution du programme.

$$E \approx V_{meas} \sum_{t=0}^{n} I_t \Delta t \tag{3.8}$$

où E est l'énergie totale sur n échantillons en utilisant une seule mesure de tension V_{meas}, I_t est le courant et Δt est l'intervalle de temps.

En appliquant leurs outils sur des scénarios de vidéo adaptative, les auteurs ont réussi à obtenir une réduction de 46% de la consommation d'énergie totale lors de l'application de la compression vidéo, d'une taille d'affichage plus petite et d'optimisations de l'alimentation du réseau et du disque. Cependant, l'outil est relativement ancien. De nombreuses

techniques modernes de gestion de l'énergie du matériel, du système d'exploitation et du logiciel n'étaient pas encore mises en œuvre il y a plus de vingt ans. Sur un système moderne, la réduction d'énergie peut être inférieure au nombre obtenu par les auteurs dans leurs recherches.

PowerAPI & Jalen PowerAPI [10] [25] est une interface de programmation d'application (API) pour surveiller la consommation d'énergie des applications, en temps réel, au niveau des processus système. Jalen [127] est une architecture de profilage des logiciels, conçue pour surveiller la consommation d'énergie des applications au niveau du code logiciel (les méthodes). Les deux outils utilisent des modèles de puissance pour estimer la consommation d'énergie des processus et des blocs de code. Les modèles (les estimations) sont divisés par ressources matérielles et par applications logicielles. Concrètement, $PowerAPI$ peut estimer la consommation d'énergie d'un processus en cours d'exécution pour le CPU, ou bien pour le disque dur, ou pour les deux ensemble ou plusieurs autres ressources matérielles. $Jalen$ peut également proposer des estimations de ressources par matériel. De plus, il estime la consommation de logiciels à une granularité plus fine.

En particulier, le modèle CPU est basé sur l'équation standard CMOS [11] [137] :

$$Power_{CPU}^{f,v} = c \times f \times V^2 \tag{3.9}$$

où f est la fréquence du processeur, V est la tension et c est une valeur constante en fonction des matériaux du matériels (tels que la capacité et le facteur d'activité).

Le modèle CPU pour un processus logiciel est défini comme la moyenne de la puissance CPU de chaque fréquence par le temps CPU de toutes les fréquences :

$$P_{comp} = \frac{\sum_{f \in frquencies} P_{CPU}^{f} \times t_{CPU}^{f}}{\sum_{f \in frquencies} t_{CPU}^{f}} \tag{3.10}$$

Et enfin, le modèle CPU d'une méthode dans une application est lié au temps CPU utilisé pour l'exécution de la méthode. L'équation est résumée comme suit :

$$Power_{method}^{CPU} = \frac{Time_{method}^{CPU} \times Power_{thread}^{CPU}}{Duration_{cycle}} \tag{3.11}$$

où $Duration_{cycle}$ est la durée du cycle.

Le modèle de disque suit une tendance similaire en basant les calculs sur le nombre d'octets lus et écrits sur le disque principal à partir d'un processus spécifique. L'équation est la suivante :

$$Power_{process}^{disk} = Bytes_{read} \times Power_{reading} + Bytes_{write} \times Power_{writing} \tag{3.12}$$

où $Bytes_{read/write}$ est le nombre d'octets lus/écrits sur le disque par le processus, et $Power_{reading/writing}$ est la puissance nécessaire pour lire/écrire un octet depuis/vers le disque. Ces dernières, sont spécifiques au matériel, elles sont fournies par les constructeurs.

Pour les méthodes, la puissance consommée est liée à la taille des données échangées avec le disque. L'équation est définie comme une multiplication croisée comme montré dans la formule 3.13 :

10. https://powerapi-ng.github.io/introduction.html
11. Complementary Metal Oxide Semiconductor

$$Power^{disk}_{method} = \frac{Bytes^{disk}_{method} \times Power^{disk}_{process}}{Bytes^{disk}_{process}} \tag{3.13}$$

La précision de *PowerAPI* est mesurée à l'aide d'un wattmètre et le taux d'erreur calculé peut varier entre 0,5% et 3%. Grâce à *PowerAPI* et *Jalen*, les auteurs ont réussi à détecter les *hotspot* énergétiques dans les logiciels. Les auteurs ont considéré l'exemple du serveur web *Jetty* où ils décrivent la répartition de l'énergie entre les classes et les méthodes dans leur scénario d'expérimentation. Cependant, l'approche est limitée par seulement les modèles mis en œuvre (par exemple, CPU, disque et carte réseau Ethernet). *Jalen* utilise également une instrumentation en bytecode, donc une surcharge non négligeable est présente (calculée à environ 57% pour les requêtes individuelles du serveur *Tomcat*). Ce qui a mené les auteurs à développer une nouvelle version en utilisant uniquement l'échantillonnage statistique, ce qui réduit les charges. Dans leurs travaux, les auteurs discutent également de la pertinence des valeurs brutes pour comparer la consommation d'énergie des logiciels sur un ensemble différent de périphériques matériels.

En plus des approches précédentes, d'autres outils proposés par les industriels offrent des informations énergétiques. *PowerTop* [12] est un outil Linux pour diagnostiquer les problèmes de consommation et de la gestion de l'alimentation. Il rapporte une estimation de la consommation d'énergie des applications logicielles et des composants du système. Il offre également un mode interactif où les utilisateurs peuvent appliquer de différents paramètres de gestion de l'alimentation non activés par défaut dans la distribution Linux utilisée. *PowerTop* partage donc des similitudes avec *pTop*, mais aussi des limitations telles que le manque de résultats fins. *EnergyChecker* [13] est un SDK [14] créé par Intel qui fournit des fonctions pour exporter et importer des compteurs depuis une application. Ces compteurs mesurent le temps passé pour un événement ou un processus particulier, comme la lecture d'un fichier ou la conversion d'une vidéo. Les compteurs sont ensuite utilisés pour estimer la consommation électrique de l'application. Cependant, l'estimation de puissance nécessite un wattmètre matériel, limitant ainsi la flexibilité de l'approche. *Joulemeter* [15] est un outil logiciel qui estime la consommation énergétique des ressources matérielles et des applications logicielles dans un ordinateur. Il surveille l'utilisation des ressources, telles que l'utilisation du processeur ou la luminosité de l'écran, afin d'estimer la consommation d'énergie de ces ressources. *Joulemeter* utilise des modèles de puissance spécifiques à la machine pour la configuration matérielle. Leur modèle actuel prend en compte les états P du processeur, l'utilisation de l'alimentation, les niveaux d'E/S du disque et si l'écran est allumé ou éteint. Les modèles, cependant, sont appris par le calibrage. Cela entraîne une limitation en termes de flexibilité, car les modèles de puissance ne peuvent pas être estimés sans des références de laboratoire préalables.

D'autre part, les statistiques des applications sont proposées à travers les outils de profilage, qui peuvent être commerciaux ou bien open-source. Ces derniers dépendent généralement du langage de programmation, comme GNU Gprof [16] et C Profiler [17] pour le langage *C*, ANTS Performance profiler [18], AQtime Pro [19] ou SlimTune [20] pour le langage *.NET*. En Java, des outils

12. https://01.org/powertop
13. http://software.intel.com/en-us/articles/intel-energy-checker-sdk
14. Software development kit : Kit de développement.
15. http://research.microsoft.com/en-us/projects/joulemeter
16. http://www.gnu.org/software/binutils/
17. http://www.semdesigns.com/Products/Profilers/CProfiler.html
18. http://www.red-gate.com/products/dotnet-development/ants-performance-profiler/
19. http://smartbear.com/products/development-tools/performance-profiling/
20. http://code.google.com/p/slimtune/

tels que VisualVM [21], Java Interactive Profiler (JIP) [22], JProfiler [23], et Oktech Profiler [24], offrent des informations générales sur l'application et des statistiques d'utilisation des ressources. Sauf que, ces outils de profilage ne fournissent pas d'informations sur la consommation d'énergie de l'application au niveau des threads ou des méthodes. Cependant, ces outils ne fournissent pas d'informations liées au réseau, telles que le nombre d'octets transmis par les méthodes, et ainsi ne peuvent pas offrir des statistiques précises de l'énergie consommée.

De nos jours, la mesure de l'énergie peut être divisée en trois catégories : la mesure matérielle, les modèles de puissance et la mesure logicielle. La mesure matérielle offre une grande précision, mais à un niveau de granularité plus épais. Elle nécessite, comme son nom l'indique, du matériel supplémentaire qu'il soit embarqué ou bien branché en externe. La principale limite d'une telle approche est l'impossibilité d'évoluer et la difficulté de mise à l'échelle. Les modèles de puissance fournissent des modèles pour calculer ou estimer la consommation d'énergie du matériel et des logiciels. Les modèles sont soit trop génériques, soit dépendants de la plate-forme ce qui est valable aussi pour les outils basés sur des modèles énergétiques qui souffrent également de la dépendance à la plate-forme.

La mesure de la consommation énergétique des appareils et des logiciels est pertinente lorsque les informations collectées sont réutilisables. Les valeurs de consommation d'énergie brute dépendent du matériel, ainsi, elles ne peuvent pas être utilisées telles quelles avec du matériels différent ou des configurations différentes. Elles peuvent également, dans une moindre mesure, fluctuer même sur la même machine et la même configuration en raison d'imperfections électromécaniques. Par conséquent, une méthodologie uniquement logicielle offre suffisamment d'avantages pour surpasser cette limitation de la réutilisabilité, tout en conservant une précision raisonnable. Une revue de l'état de l'art et une comparaison des différentes approches pour mesurer l'énergie est donnée dans [128]. Dans cette revue, les auteurs détaillent les approches de mesure et de modélisation de l'énergie, dont ils agissent de compteurs matériels ou de modélisation logicielle. Ils ont également décrit les approches de modélisation au niveau du système et les outils utilisés pour estimer l'utilisation des ressources matérielles et la consommation d'énergie.

Les approches précises nécessitent un investissement matériel sous la forme de wattmètres, tandis que les approches logicielles manquent de précision, de convivialité et de flexibilité, et les profileurs logiciels ne traitent pas la dimension énergétique. Les compteurs d'énergie ont l'inconvénient d'être difficiles, voire impossibles, à améliorer. Ils ne fournissent également que des informations au niveau matériel, pas logiciel. Enfin, les compteurs matériels tombent dans l'oubli, car les utilisateurs cesseraient de les utiliser après un certain temps. Nous soutenons que les solutions efficaces nécessitent des approches de mesure de l'énergie uniquement logicielles, et que l'estimation logicielle basée sur un modèle de la consommation d'énergie fournie une estimation précise pour les approches de gestion et d'optimisation de l'énergie. Dans la section suivante, nous présenterons les différentes approches d'intégration de l'énergie des les systèmes informatiques.

3.3 Intégration de l'énergie dans les systèmes informatiques

De nos jours, les enjeux de l'informatique éco-énergétique sont de plus en plus étudiés par les industriels et les chercheurs à tous les niveaux de l'infrastructure informatique. En effet, de nombreux efforts ont été consacré à l'amélioration de l'efficacité énergétique des systèmes informatiques. Dans cette section, nous présentons une brève description des méthodes d'évaluation

21. http://visualvm.java.net
22. http://jiprof.sourceforge.net/
23. http://www.ej-technologies.com/products/jprofiler/overview.html
24. http://code.google.com/p/oktech-profiler

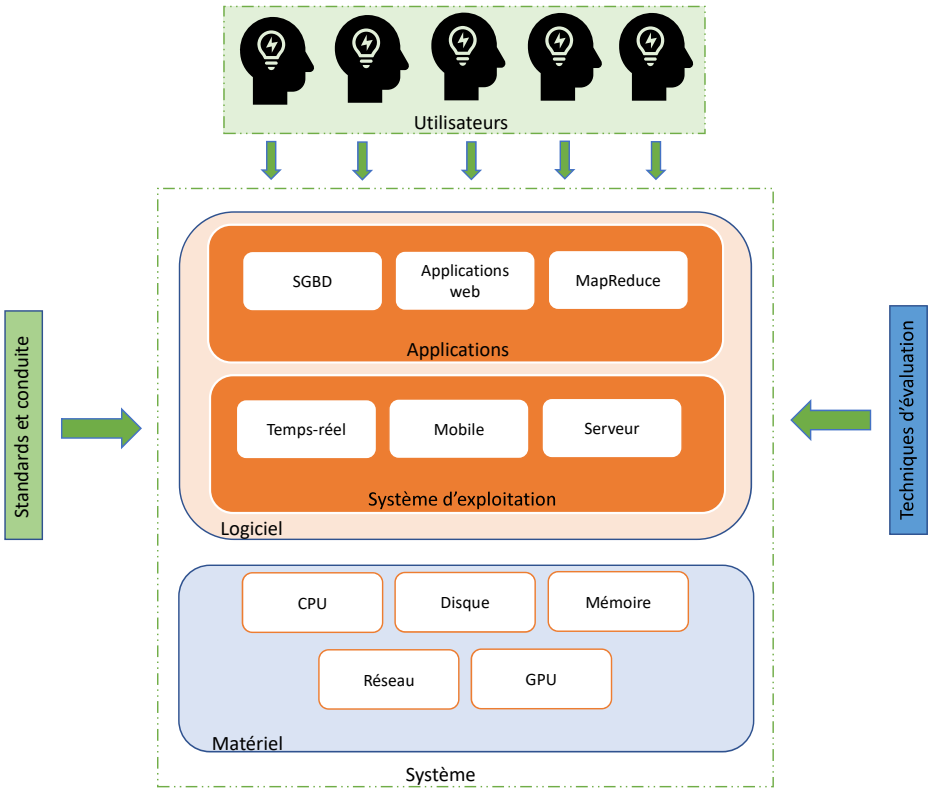


FIGURE 3.4 – La consommation d'énergie dans les systèmes informatiques

d'énergie dans les *SI*s, nous introduisons une revue des principaux travaux d'EE des systèmes informatiques en les classant en deux niveaux : (1) matériel, (2) logiciel. Nous présentons également les initiatives et les standards introduits par les différentes organisations internationaux. La figure 3.4 illustre l'interdépendance des différents influenceurs sur la consommation d'énergie des systèmes informatiques.

3.3.1 Méthodes et métriques d'évaluation de l'efficacité énergétique

Afin de pouvoir évaluer et comparer les différentes méthodes éco-énergétique et les approches développées et de comprendre les différences existantes entre elles, les méthodes et les métriques d'évaluation ont une très grande importance et constituent l'argument principal lors du choix et la préférence d'un système par rapport un autre. Dans le but d'évaluer l'efficacité énergétique des systèmes informatiques, une multitude de métriques ont été proposées dans la littérature. Ces métriques sont fondées : (*i*) sur l'énergie totale consommée (*ii*) sur l'exploitation des ressources du système, (*iii*) sur la chaleur dégagée par le système lors de son fonctionnement [39]. D'autre part, les méthodes d'évaluation permettent aux fabricants d'ordinateurs, aux chercheurs et aux utilisateurs finaux sensibles à l'énergie, de prendre une décision fondée sur des bases scientifiques lors de l'achat ou bien l'identification des technologies prometteuses. Pour fournir des comparaisons normalisées, justes et précises, des règles bien définies sont primordiales [146]. C'est pour cette raison, que les scientifiques et les industriels ont fourni des efforts considérables afin de proposer des mesures d'évaluation et des techniques de modélisation de l'énergie. Ces dernières peuvent être regroupées en trois catégories :

les approches basées sur les modèles mathématiques : qui sont les plus utilisées dans la littérature. Ces approches modélisent le système par un modèle mathématique (ensemble

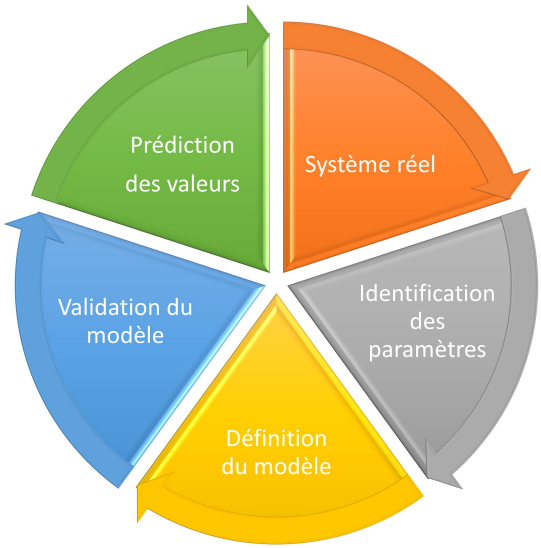


FIGURE 3.5 – La conception d'un modèle mathématique

d'équations) qui prend en entrée des mesures collectées en temps réel sur les données et des mesures sur l'énergie consommée et retourne des prédictions de la consommation d'énergie du système. Pour estimer le coût d'exécution ($Cost$) d'une tâche (ts) dans un système informatique, l'équation 3.14 peut être utilisée.

$$Cost(ts) = \alpha \times I^{ts}_{cpu} + \beta \times I^{ts}_{cache} + \gamma \times I^{ts}_{mem} + \delta \times I^{ts}_{io} + \sigma \times I^{ts}_{com} + \epsilon^{ts} \tag{3.14}$$

où I^{ts}_{cpu}, I^{ts}_{cache}, I^{ts}_{mem}, I^{ts}_{io}, I^{ts}_{com}, et ϵ^{ts} représentent respectivement, le nombre d'instructions du processeur, les défauts de cache du processeur, l'utilisation de la mémoire, le taux d'E/S du disque, la quantité des données transférées sur le réseau et les erreurs d'estimation. Les constantes α, β, γ, δ, et σ sont obtenues à partir de l'apprentissage automatiques en utilisant les techniques de machine learning [118]. Ces modèles permettent d'améliorer l'efficacité énergétique en l'utilisant pour l'optimisation des algorithmes existants ou bien pour la conception des composants et des systèmes éco-énergétique.
En partant d'un système réel, la construction d'un tel modèle se fait en quatre étapes comme le montre la figure 3.5 :

1. **l'identification des paramètres** : dans cette étape, nous mesurons la consommation d'énergie des composants pour identifier les paramètres et les composants sensibles à l'énergie et détecter ceux qui sont les plus énergivores.
2. **la définition du modèle** : les paramètres sélectionnés sont utilisés pour la construction du modèle à travers les techniques d'apprentissage. Le choix de ces derniers est très important pour la précision du modèle.
3. **la validation du modèle** : le modèle est validé expérimentalement pour la validation de son adéquation aux fins prévues, l'estimation de l'énergie dans notre cas.
4. **la prédiction des valeurs** : enfin, le modèle construit est utilisé pour prédire la consommation d'énergie du système.

Une panoplie de modèles ont été proposés dans la littérature pour la modélisation de l'énergie en traitant les différents niveaux d'un système informatique (architecture, composants, circuit, système d'exploitation, application). Les auteurs dans [18, 42, 58, 100, 142], ont proposé des revues de la littérature et des taxonomies pour l'efficacité énergétique dans les centres de données et dans le cloud.

les outils de simulation : ces outils sont utilisés pour calibrer et prédire la consommation énergétique d'un système. Ce genre d'outil est utilisé sur un large éventail de domaines d'application. Les outils de simulation permettent de réduire la complexité de l'extraction des informations détaillées du fonctionnement des différents composants du système, par la modélisation de chaque composant de manière indépendante plutôt qu'en un ensemble. Ce qui permet de minimiser le coût mais n'offre pas la probabilité, par ce que la simulation repose sur des connaissances spécifiques de chaque composant [178]. Parmi les outils de simulation, nous pouvons citer comme : *Wattch* proposé par Brooks et al. [26] depuis les années 2000, *Tempo Simulator* proposé par Shafi et al. [158], et *SoftWatt Simulator* proposé par Gurumurthi et al. [66] qui permet de modéliser la consommation énergétique des différents composants (CPU, mémoire, support de stockage), et ensuite de quantifier l'énergie nécessaire pour le fonctionnement des applications et du système d'exploitation.

le benchmarking : ou bien, les bancs de tests, sont des outils d'analyse, utilisés comme des références pour se comparer aux meilleurs techniques (voir la définition 1). Dans le contexte de l'EE les benchmark sont utilisés pour l'estimation de la consommation énergétique d'un système ou bien de ses composants. Ainsi, les chercheurs, les agences gouvernementales et les consortiums standard de l'industrie ont développé des benchmarks énergétiques permettant de comparer l'EE des systèmes informatiques. Parmi les benchmarks proposés, nous pouvons citer : $TPC-Energy$ [25] développé par Transaction Processing Performance Council (TPC), qui permet d'introduire des mesures énergétique pendant le traitement des requêtes pour assister les utilisateurs à identifier les équipements éco-énergétiques [190], $SPECpower\_ssj2008$ [26], introduit par Standard Performance Evaluation Corporation (SPEC), permet de mesurer la performance et la consommation d'énergie d'un système exécutant des charges de travail basées sur le langage Java. Il exerce le CPU, les caches, et la hiérarchie de mémoire. The Storage Performance Council (SPC) qui permet d'évaluer les composants de stockage, a inclut la mesure de la consommation d'énergie en plus des performances de stockage dans ses bancs de tests SPC Benchmark 2(SPC-2) et SPC Benchmark 2C(SPC-2C) pour proposer les deux nouveaux benchmark sensible à l'énergie SPC-2/E et SPC-2C/E respectivement. Rivoire et al. [143] ont proposé *JouleSort* qui est un benchmark sensible à l'énergie dédié aux algorithmes de tri.

3.3.2 Les approches matérielles

Actuellement, l'aspect énergétique est de plus en plus étudié à tous les niveaux dans les infrastructures d'un système informatique. Afin d'améliorer l'EE, de nombreux travaux ont été proposés. Ces travaux peuvent être répartis en deux parties en se basant sur la méthode de la gestion d'énergie. Ainsi, nous pouvons trouvez les techniques de gestion statique de l'énergie, qui cherchent à améliorer l'EE lors de la conception et la production des composants à travers leur miniaturisation ou bien l'optimisation de leurs circuits électroniques. Ou bien, les techniques de gestion dynamique, qui cherchent à optimiser l'EE lors de l'exécution du système [107]. Dans ce qui suit nous détaillons les travaux traitant la gestion dynamique de l'énergie du matériel dans les systèmes informatiques. Ces travaux sont scindés en quatre familles : (*i*) l'ajustement dynamique de la tension, (*ii*) la désactivation Dynamique des Composants, (*iii*) le co-traitement, et (*iv*) les supports de stockage.

L'ajustement dynamique de la tension et de la fréquence (DVFS [27]) permet au CPU de faire varier sa fréquence et sa tension afin de réduire sa consommation d'énergie. L'équation 3.15 décrit le modèle standard de la consommation énergétique du CPU.

25. http://www.tpc.org/tpc_energy/
26. http://www.spec.org/power_ssj2008/
27. Dynamic Voltage and Frequency Scaling

$$Puissance_{CPU}^{f,v} = C \times f \times V^2 \tag{3.15}$$

où C est une valeur constante en fonction du matériels (comme la capacité, généralement mesurée en farads F, et le facteur d'activité), f est la fréquence en hertz (Hz), et V est la tension de l'alimentation en volt (V) [59]. Cette équation montre que la puissance consommée par un composant dépend fortement de sa tension et de sa fréquence. Les valeurs de la fréquence f et la tension V, sont obtenues par le système d'exploitation lors de l'exécution. Cependant, la variable statique C ne peut pas être obtenue à l'exécution. Cette dernière valeur est un ensemble de données décrivant les caractéristiques physiques du CPU (par exemple, capacité ou facteur d'activité). Les fabricants peuvent fournir cette constante bien que dans la plupart des cas, elle soit manquante.

Grâce à la formule de l'équation 3.15, nous constatons que la consommation électrique n'est pas toujours linéairement dépendante du pourcentage d'utilisation du CPU. Cela est dû à la mise à l'échelle dynamique de la tension et de la fréquence (DVFS) et également au fait que la puissance dépend de la tension (et par conséquent de la fréquence) du processeur. Par exemple, un processus qui utilise 100% du processeur ne consommera pas nécessairement plus d'énergie qu'un processus fonctionnant à 50% d'utilisation du processeur, mais avec une tension plus élevée. Cette technique est très appliquée dans les processeurs multi-cœur et les mémoires DRAM [28].

Wesier et al. [180] ont eu l'idée de contrôler d'une manière dynamique la vitesse d'horloge du CPU pour réduire sa consommation d'énergie pour une tâche particulière. Les résultats obtenus étaient encourageants, cependant, cette manoeuvre a engendré une augmentation du temps d'exécution. Et depuis, plusieurs chercheurs se sont intéressés à la réduction de l'énergie à travers le contrôle dynamique du CPU. Shin et al. [162] ont proposé une méthode dans laquelle ils tentent de ralentir le CPU à chaque fois qu'il y a une seule tâche à exécuter. Dans [75], les auteurs proposent un algorithme éco-énergétique pour l'ordonnancement. La majorité des algorithmes d'ordonnancement proposés, cherchent à trouver le meilleur compromis entre la performance et l'énergie.

Appliquer la technique de DVFS sur un processeur multi-cœur est loin d'être une tâche simple. Souvent, elle est simplifiée en forçant chaque nœud d'un package à travailler à la même fréquence et à la même tension [103]. Cette technique, qui est appelée le DVFS global, engendre une dégradation de l'efficacité énergétique. Ainsi, les architectures dites DVFS par cœur ont été proposées, dans lesquelles, les cœurs d'un même package peuvent partager la même tension et la même fréquence, mais chaque package peut être exécuté à différentes tensions et fréquences [132]. Dans [155], les auteurs proposent un ordonnanceur éco-énergétique à travers un framework permettant de satisfaire les contraintes d'équité de fonctionnement des unités de traitement. L'implémentation de ce framework sur un processeur multi-cœurs hétérogène a permis d'améliorer considérablement l'efficacité énergétique et l'équité par rapport à l'ordonnanceur standard utilisé au niveau du système Linux.

Pour les mémoires centrales, la DVFS est également utilisée, où deux méthodes sont présentées dans [4], permettant d'améliorer l'EE au niveau de la DRAM. Ces deux méthodes réduisent les demandes d'accès des ranks (rangée des barrettes), afin de prolonger leur inactivité. La méthode Rank-Aware REplacement (RARE) utilise une politique pour le remplacement du dernier niveau de cache (LLC) [29]. Cette méthode bloque le remplacement des blocs de mappage pour les ranks prioritaires, ce qui réduit les accès globaux à ces derniers. La seconde méthode Rank-Aware Write Buffer (RAWB) retient les demandes des tampons allant vers certaines barrettes, ensuite, elle les envoie en mode groupé (bach) lorsque la DRAM est en mode pleine utilisation, ce qui permet de prolonger le temps d'inactivité des ranks. Le Copy-Row DRAM (CROW) [73] ,

28. Dynamic Random Access Memory
29. Last Level Cache

est un algorithme qui permet de partitionner en deux chaque zone mémoire de la RAM (lignes régulières et lignes de copie). Les auteurs proposent deux nouveaux mécanismes, CROW-cache et CROW-ref. Pour réduire la latence d'activation de la DRAM, CROW-cache utilise une ligne de copie pour dupliquer une ligne régulière et active simultanément une ligne régulière avec sa ligne de copie dupliquée. D'autre part, pour réduire le taux d'actualisation de la DRAM, CROW-ref remappe les lignes régulières faibles en rétention vers les lignes de copie forte. Ces deux mécanismes, augmentent la vitesse de 20% et améliorent l'efficacité énergétique de 22, 3% par rapport à la DRAM standard.

Certains composants informatiques ne disposent pas de la capacité d'ajuster automatiquement leurs performances. Par conséquent, ces derniers, doivent être désactivés durant les périodes d'inactivités pour réduire leur consommation énergétique. Ainsi, la désactivation des composants peut être vue comme un cas particulier de DVFS, dans lequel la tension et la fréquence sont mis à 0. Les architectures modernes des composants offrent généralement la possibilité de s'exécuter avec de différents modes de fonctionnement ce qui permet de réduire la consommation énergétique du système. Park et al. [134] ont proposé de basculer une partie de la mémoire ou bien sa totalité en mode faible consommation, durant l'exécution des processus, afin de réduire la consommation énergétique.

Récemment, les unités de traitement graphique (GPU) sont devenues le socle des centres de données grâce à leur efficacité de calcul des fonctions à virgule flottante. De plus, les GPUs sont très économes sur le plan énergétique en les comparant aux CPU. Ce qui rend les deux principaux supercalculateurs à haut rendement énergétique sont basés sur des GPU [30]. Ainsi, les architectures modernes tirent profit des performances des GPU en permettant de combiner les unités de traitement et en admettant le co-traitement. Une étude comparative est effectuée dans [168], où les auteurs comparent le CPU et le GPU d'un point de vue énergétique et d'un point de vue performance. Les résultats obtenus montrent que le GPU surpasse le CPU en offrant un avantage significatif en termes de performances. Une autre étude qui compare le CPU avec les CPU multi-cœurs et le FPGA [31] est donné dans [53], dans laquelle les auteurs ont évalué la performance et le comportement énergétique des différentes architectures. Dans cette étude, la FPGA a prouvé son efficacité en termes de la consommation énergétique et également pour la performance.

Afin d'améliorer l'efficacité énergétique des supports de stockage, deux techniques sont proposées : (*i*) rendre le matériel plus écologique (*ii*) réduire la duplication des données [189]. Les disques SSD (Solid-State Drive) utilisent une mémoire Flash. Cette dernière, est une mémoire non-volatile ayant des caractéristiques similaires à la mémoire morte effaçable électriquement et programmable (EEPROM) [151]. Les SSD possèdent des propriétés proportionnelles à l'énergie, c'est à dire l'énergie consommée est proportionnelle aux opérations d'E/S effectuées par seconde [163].

3.3.3 Les approches logicielles

Comme indiqué dans [165], les fabricants des appareils mobiles cherchent à augmenter la durée de vie de la batterie de ces derniers. De grandes améliorations ont été apportées aux technologies de la batterie, des processeurs et des écrans pour réduire leur consommation d'énergie. En plus de ces évolutions matérielles, le logiciel (application) peut aider facilement à réduire la consommation d'énergie sur les appareils mobiles et également à prolonger la durée de vie de la batterie. Ainsi, les chercheurs ont présenté les caractéristiques des logiciels verts et les méthodologies de conception de logiciels pour améliorer l'efficacité énergétique des logiciels.

30. https://www.top500.org/green500/
31. Field-Programmable Gate Array

Selon [28], l'*EE* en génie logiciel est définie comme les pratiques qui appliquent les principes d'ingénierie aux logiciels en tenant compte de l'aspect énergétique. Le développement, l'exploitation et la maintenance des logiciels sont donc effectués de manière verte et produisent un produit logiciel vert. D'après les auteurs dans [167], un logiciel vert doit répondre à trois exigences abstraites :

— Les processus d'ingénierie logicielle requis pour le développement, la maintenance et le déploiement des logiciels doivent économiser les ressources et réduire le gaspillage.
— L'exécution du logiciel doit économiser les ressources et réduire le gaspillage.
— Les logiciels doivent soutenir le développement durable.

De ces exigences, les auteurs en déduisent trois facteurs verts abstraits :

— Faisabilité : L'efficacité des ressources pour développer et maintenir le logiciel.
— Efficacité : L'efficacité des ressources pour exécuter le logiciel.
— Durabilité : comment le logiciel soutient le développement durable.

où le développement durable est défini par la capacité de produire peu ou pas de dégâts à l'environnement et donc capable de durer longtemps [46], ou bien, par la voie du développement qui répond aux besoins du présent sans compromettre la capacité des générations futures à répondre à leurs besoins [37].

Les exigences précédemment détaillées, sont utilisées lors du développement de tous les logiciels verts, en ce qui suit, nous présenterons, les approches éco-énergétique permettant de réduire l'énergie au niveau des applications, systèmes d'exploitation et dans le cloud.

3.3.3.1 Système d'exploitation et applications

Le système d'exploitation (SE) est un ensemble de programmes qui orchestre l'utilisation des ressources d'un ordinateur à travers des logiciels applicatifs, et vu son importance, les chercheurs ont rapidement réalisé son impact significatif sur la consommation de l'énergie. Benini et al. [19] ont proposé une classification des composants du système en se basant sur leur consommation d'énergie, et ils ont présenté également une revue de l'état de l'art des différentes techniques éco-énergétiques autour de trois phases principales de la conception des systèmes : la conception et la modélisation, la mise en œuvre, et la gestion d'exécution. Les auteurs ont montré aussi que les SE ont une consommation d'énergie hétérogène et qu'elle peut varier même entre les versions du même SE. Par exemple, la consommation d'énergie d'un même ordinateur en utilisant des différentes versions de Windows ou bien des noyaux Linux différents, présente des variations non-négligeables [130].

Le gouverneur à la demande, est un gestionnaire d'alimentation en temps réel développé dans [133] au sein du noyau d'un système Linux. Il permet de surveiller en permanence l'utilisation du processeur en modifiant la fréquence d'horloge pour satisfaire les exigences de performances demandées. Cet ajustement de la fréquence d'horloge minimise la consommation d'énergie. D'autres techniques sont développées pour les systèmes d'exploitation destinés aux dispositifs alimentés à travers les batteries, tels que les mobiles. Parmi ces techniques, nous pouvons citer, ECOsystem [192], qui offre une interface permettant de définir la durée de vie de la batterie, ainsi que les priorités d'application utilisées afin d'estimer la quantité d'énergie allouée aux applications à chaque période. Des techniques similaires ont été proposées pour les mobiles comme Nemesis OS [124], et Coda/Odyssey [52], pour les serveurs nous pouvons trouver, PowerNap [117], et Barely Alive [5].

Gurumurthi et al. [66] ont introduit un simulateur de la puissance du système qui représente le processeur, la hiérarchie de la mémoire et un sous-système de disque à faible consommation d'énergie. Ce simulateur calcule les performances énergétiques au niveau du système d'exploitation et au niveau des applications.

Les auteurs dans [170] ont défini une méthode d'analyse de puissance au niveau des instructions, qui fournit un moyen précis et pratique de mesurer le coût énergétique des applications et

d'analyser la puissance au niveau des instructions basée sur l'évaluation qui permet d'analyser efficacement la consommation d'énergie des logiciels.

Plusieurs applications peuvent être exécutées avec plusieurs modes et de différentes manières pour effectuer la même tâche. Ainsi, les applications qui sont aptes d'adapter leurs manières d'exécution en fonction des états liés à l'alimentation du système, offrent une alternative additionnelle dans le but d'économiser d'énergie. Ces raisons ont rendu l'amélioration de l'EE au niveau des applications un domaine de recherche actif. De nombreuses approches ont été proposées dans la littérature pour optimiser les applications web [34], Map-Reduce [101, 112, 169], les compilateurs [76] et les environnements de développement intégrés (EDI) [114]. Xian et al. [183] ont développé un framework de programmation appelé SEEDS qui permet aux développeurs informatiques de concevoir des applications éco-énergétique. SEEDS permet aux applications d'être exécutées en utilisant des plans différents en fonction de leurs consommations d'énergie. Ce framework offre une analyse automatisée, une prise de décision et une mise en œuvre dans le but d'améliorer l'efficacité énergétique d'une application donnée.

Dans [67], les auteurs ont étudié l'impact des requêtes d'insertion massive de données en utilisant le langage PL/SQL du point de vue de la consommation énergétique globale, la consommation énergétique du processeur et le temps de réponse. Pour les expérimentations, les auteurs ont utilisé PowerAPI, qui utilise la puissance de conception thermique (TDP) [32] qui peut être décrite comme la quantité maximale de chaleur que le CPU peut émettre pendant le traitement. Les expérimentations ont montré que pour insérer le même volume de données, un algorithme utilisant la boucle FOR a un temps de réponse significativement plus élevé que la deuxième méthode utilisant l'instruction $FORALL$, également pour la consommation d'énergie, l'instruction FORALL est plus efficace que FOR.

Récemment, la consommation d'énergie dans les bases de données en général a commencé à attirer l'attention de la communauté des chercheurs. L'objectif de ces travaux est d'intégrer l'aspect énergétique et de concevoir ainsi des SGBDs sensibles à l'énergie. Notre étude s'inscrit dans cette catégorie, et nous allons détailler dans une section séparée (cf. 3.4) les différents travaux éco-énergétique de la littérature dans les bases de données. La Figure 3.6 représente une classification des travaux d'EE dans les systèmes informatiques sur les niveaux discutés.

3.3.3.2 Le cloud

Le cloud computing a changé l'approche classique qui consistait à acquérir ses propres ressources matérielles et logicielles pour satisfaire les besoins des utilisateurs, à une approche fondée sur l'allocation des infrastructures, des plateformes et des logiciels en tant que services en se basant sur la notion d'élasticité qui évolue au cours du temps selon les besoins. Cette flexibilité et facilité d'acquérir et de libérer des ressources présentent d'énormes avantages pour les administrateurs. Ce paradigme est fondé sur l'utilisation des technologies comme la virtualisation et le traitement distribué pour offrir les ressources aux utilisateurs. Il se présente comme un moyen pour optimiser l'efficacité énergétique à travers la réduction des émissions de gaz à effet de serre [90]. Le cloud offre plusieurs avantages : (a) utilisation accrue des ressources (b) indépendance de l'emplacement, c'est à dire, les machines virtuelles peuvent être déplacées vers un endroit où l'énergie est moins chère (ou plus verte), (c) adaptation de l'utilisation des ressources aux besoins de l'utilisateur [18].

Afin d'améliorer l'EE dans les systèmes informatiques, la technique de virtualisation est largement appliquée. Cette technique consiste à assembler plusieurs machines physiques sur un seul serveur, et ensuite, créer un ensemble de machines virtuelles en se basant sur les machines physiques. Cette méthode permet l'exécution de nombreuses machines virtuelles d'une façon indépendante et elle désactive ou bien elle met en mode de veille prolongée les machines

32. Thermal Design Power

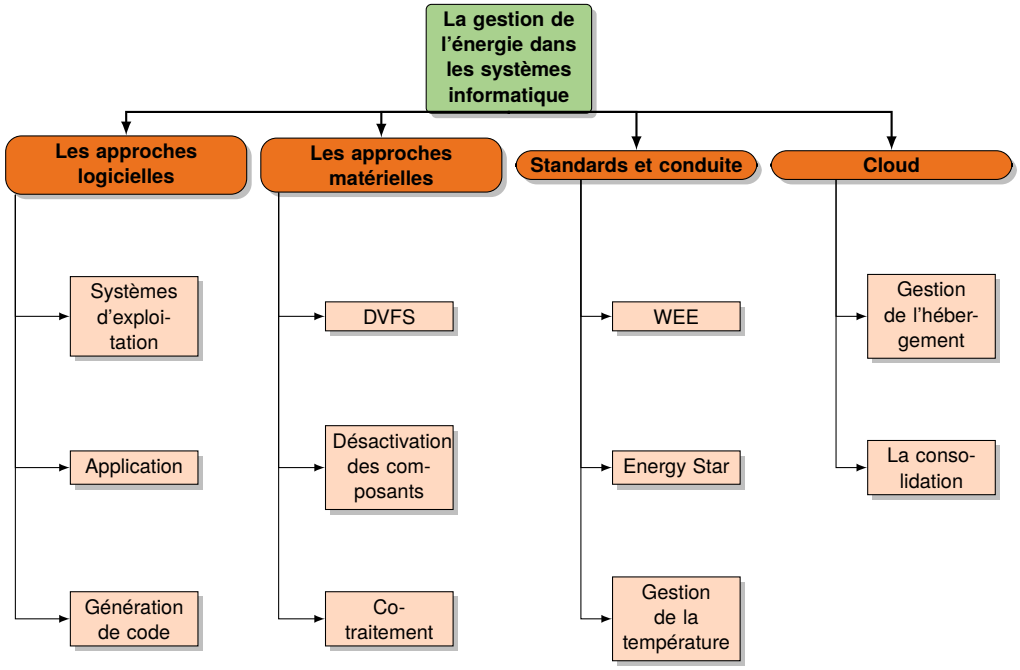


FIGURE 3.6 – Classification des approches d'EE dans les systèmes informatique

non utilisées [159]. VirtualPower [121] est un système qui assure la gestion de l'énergie dans les environnements virtualisés. Ce système se repose sur des méthodes logicielles et sur l'ajustement dynamique de l'énergie des plateformes. La gestion de l'énergie prend en compte les politiques de gestion indépendantes des machines virtuelles installées et les coordonne pour atteindre l'objectif énergétique. Nguyen et al. [141] ont introduit l'EPOBF (Energy-aware and Performance-per-watt Oriented Bestfit), qui est une heuristique d'allocation des machines virtuelles. EPOBF utilise la métrique de performance-par-watt dans le but de choisir la configuration optimale en termes d'efficacité énergétique parmi les machines disponibles. Les expérimentations ont montré qu'une réduction de 35% de l'énergie consommée peut être atteinte.

Nous pouvons diviser les approches de réduction de l'énergie dans les environnements Cloud en deux catégories : (*i*) les approches d'optimisation des machines consacrées à l'hébergement des machines virtuelles, et (*ii*) les approches se basant sur la consolidation des machines virtuelles dans un même cluster. Les approches de la première catégorie essaient de réduire l'énergie des serveurs par le biais de la technique du DVFS, par contre, les approches de la deuxième catégorie visent à migrer les MVs sur un nombre minimal de machines [27]. La consolidation consiste à migrer les MVs entre les serveurs afin de libérer les serveurs pour effectuer des travaux de maintenance ou pour les éteindre, ou bien pour réorganiser dynamiquement les MVs en fonction de leur utilisation des ressources. La migration de MV entre les clusters peut rencontrer deux défis : (1) Optimiser le placement des MVs et (2) Minimiser les délais de placement et la consommation énergétique [179]. Plusieurs travaux se sont consacré au problème de placement éco-énergétique des MVs, nous pouvons citer [55, 93, 33, 51]

3.3.3.3 Standards et Conduite

Pour remédier à la consommation excessive de l'énergie, et afin de diminuer les émissions de gaz à effet de serre et surtout d'assurer l'approvisionnement de l'énergie qui a un impact direct sur l'économie des nations, les entreprises et les gouvernements ont imposé des standards et des règles. Par exemple, en 2008, la Commission Européenne (CE) a crée un code de conduite (The

European Code of Conduct for Data Centres) [33] afin d'améliorer l'efficacité énergétique dans les centres de données. Il s'agit d'une initiative volontaire qui présente les bonnes pratiques aux fabricants, fournisseurs, consultants et services publics, afin de réduire leur consommation d'énergie. La CE a également introduit la directive sur la gestion des déchets des équipements électriques et électroniques (WEEE) [34] afin de contrôler les impacts environnementaux des équipements électriques et électroniques usés. D'autre part, le programme ENERGY STAR lancé en 1992 par l'agence de protection de l'environnement des États-Unis, propose des lignes directrices pour promouvoir les économies d'énergie et offre des avantages environnementaux et des gains financiers grâce à une efficacité énergétique supérieure. ENERGY STAR permet de fournir des informations simples, crédibles et impartiales sur lesquelles les consommateurs et les entreprises s'appuient pour prendre des décisions. Récemment, ce programme s'est intéressé aux centres de données et cherche à réduire la consommation annuelle de plus de 70 milliards de kWh de ce secteur pour lutter contre la crise climatique [35].

Dans les centres de données, la gestion de la température présente un sérieux défit aux industriels, en partant du fait que l'augmentation de la chaleur des composants matériels d'un système augmente sa consommation d'énergie et peut engendrer une défaillance de ces derniers. D'autre part, la température ambiante autour du système peut aussi affecter sa consommation globale [144]. Ainsi, les unités de calcul, l'infrastructure de déploiement, la position géographique, et les systèmes de refroidissement sont très importants pour réduire l'énergie. ASHRAE [36] recommande que la température environnementale idéale des systèmes de stockage de données de classe A1 à A2 doit être comprise entre 18° et 27° Celsius. De nos jours, les techniques de refroidissement sont devenues un sujet de recherche actif ce qui a incité les chercheurs et les industriels à développer de nouvelles technologies de refroidissement offrant une meilleure efficacité énergétique [70]. Une revue des systèmes de refroidissement est donnée dans [49], dans laquelle les auteurs classent l'efficacité énergétique de ces systèmes en trois classes (1) refroidissement par air, (2) refroidissement par liquide et (3) refroidissement en deux phases. Les systèmes de refroidissement à base d'eau ont prouvé leur efficacité en termes d'économie d'énergie par rapport aux systèmes traditionnels de refroidissement par air, ce qui a encouragé les entreprises à opter pour cette technologie.

3.4 Intégration de l'énergie dans les bases de données

L'explosion des données numérisées a placé la communauté des bases de données au carrefour du débat sur l'efficacité énergétique. Les centres de données, qui représentaient environ 1% (ou 205 TWh) de la consommation mondiale d'électricité en 2018 [115], doivent stocker et traiter le déluge de données. Dans cette situation, la conception des systèmes basés sur les données doit concilier deux besoins non fonctionnels, réduire l'énergie consommée sans dégrader la performance. La réduction de la consommation énergétique des unités de traitements des requêtes est considérée comme un acteur essentiel pour réduire la gourmandise énergétique de ces systèmes. Historiquement, la satisfaction des performances des requêtes est la marque de fabrique des produits des bases de données. Cependant, et face à l'avènement des besoins écologiques, la communauté de bases de données ne peut pas rester inactive, et la conception des SGBD sensibles à l'énergie est devenue un sujet de recherche actif au cours des dernières années. Le rapport Claremont [37] publié en 2008 a incité la communauté des bases de données

33. https://joint-research-centre.ec.europa.eu/energy-efficiency/energy-efficiency-products/code-conduct-ict/code-conduct-energy-efficiency-data-centres_en
34. https://ec.europa.eu/environment/waste/weee/index_en.htm
35. https://www.epa.gov/newsreleases/energy-star-expands-efforts-improve-energy-efficiency-us-data-centers
36. American Society of Heating, Refrigeration and Air Conditioning
37. https://dsf.berkeley.edu/claremont/clarremontreport08.pdf

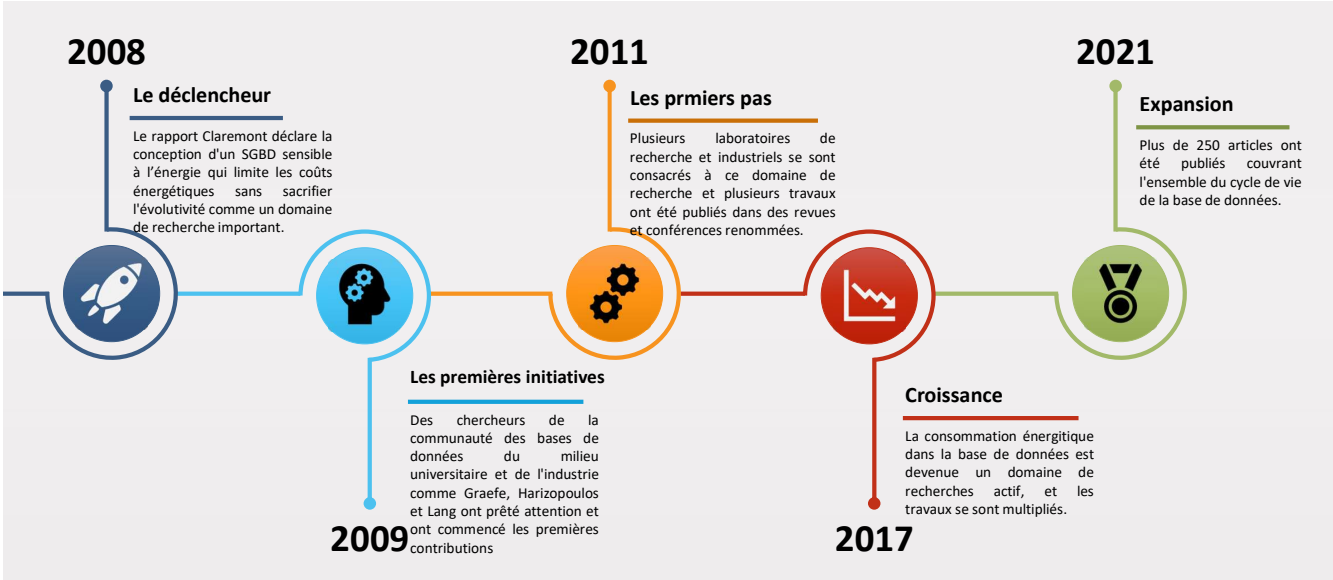


FIGURE 3.7 – L'intérêt de la communauté de bases de données à l'énergie

à réagir face à cette urgence afin de renforcer l'efficacité énergétique des produits des bases de données. Depuis cette date, une pléthore d'études et d'initiatives qui se sont principalement concentrées sur la modélisation et la gestion de la consommation énergétique des opérations de données (telles que les jointures, les sélections, les tris), sont proposées par des chercheurs comme le montre la figure 3.7.

La consommation énergétique d'un SGBD dépend généralement de la base de données qu'il héberge, plus précisément, du schéma de la *BD* (les tables et les attributs), sa population (en termes d'instances) et son degré d'exploitation via les requêtes. La plus grosse consommation d'un SGBD est due au calcul effectué par ses composantes principales telles que l'optimiseur de requêtes, le gestionnaire de stockage, le gestionnaire des méthodes d'accès, le gestionnaire de buffer et le contrôleur de concurrence [151]. Cependant, la majorité des solutions EE développées par les académiques et dans l'industrie se concentrent principalement sur le côté du matériel [164]. En ce qui suit, nous présentons les approches matérielles et logicielles proposées par la communauté des bases de données afin de réduire l'énergie.

3.4.1 Les approches matérielles

La communauté des bases de données a tiré profit des nouvelles technologies et des capacités modernes des architectures du matériels informatique afin d'améliorer son efficacité énergétique. Pour atteindre cet objectif, des techniques naïves comme la désactivation de composants électroniques pendant les périodes d'inactivité ou bien des techniques plus avancées comme l'ajustement dynamique de la tension et la fréquence, et le traitement parallèle, ont été utilisés. Tous les composants intervenant dans l'exécution d'une requête ont été concerné par la réduction de l'énergie. Dans cette section, nous présentons une revue des différentes approches pour améliorer l'efficacité énergétique des bases de données d'un point de vue matériel en regroupant ces approches par composant comme le montre la figure 3.8.

3.4.1.1 Les unités de traitement

Xu et al. [184] ont identifié le processeur comme le consommateur majeur d'énergie active, et depuis, plusieurs chercheurs se sont intéressés à l'optimisation de ce composant en utilisant plusieurs techniques [102, 176, 187, 74, 29, 138, 86, 85, 96, 18, 126, 188]. Pour réguler la consommation

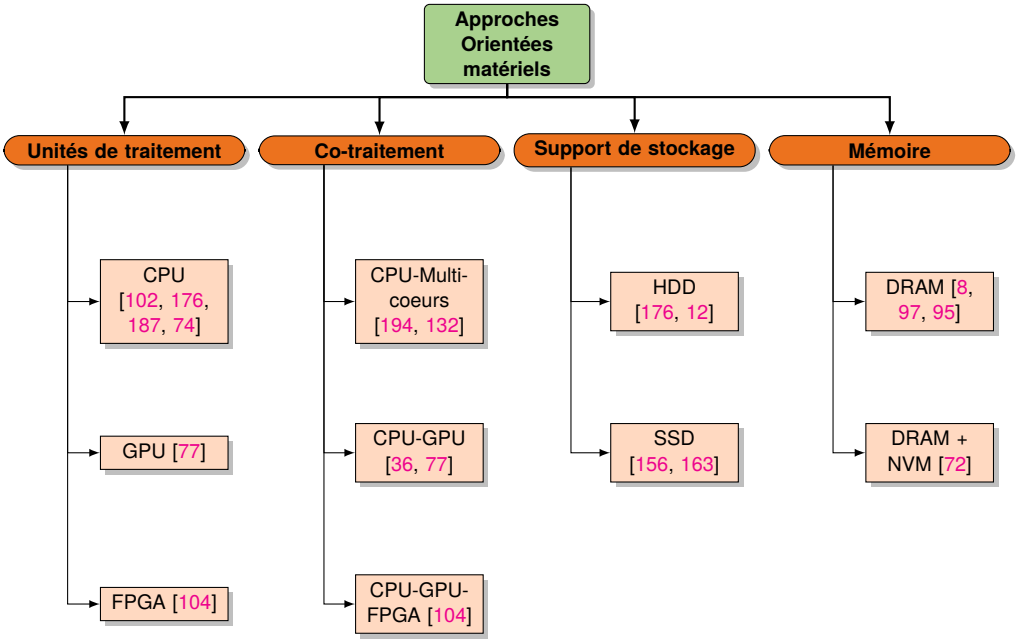


FIGURE 3.8 – Classification des approches d'EE orientées matériels dans les bases de données.

du CPU, les auteurs dans [102], [187] et [176], ont utilisé la technique du DVFS. Les approches proposée réduisent la consommation d'énergie du système avec une certaine dégradation de la performance d'exécution des requêtes à travers l'ajustement de la fréquence du système en fonction des caractéristiques et du débit de la charge de travail. D'autre part, et en tirant parti de la capacité des processeurs modernes, Lang et al. [102] ont proposé le mécanisme PVC (Processor Voltage/Frequency Control). Ce mécanisme vise à exécuter les instructions à une tension et une fréquence de processeur basse afin d'atteindre un équilibre entre la consommation d'énergie et la performance. Afin d'évaluer l'influence de la réduction de la fréquence d'horloge du processeur sur l'efficacité énergétique de l'implémentation algorithmique des opérateurs relationnels (accès séquentiel, la jointure, le tri, et les agrégations), les auteurs dans [126] ont mené des expérimentations dans lesquelles ils ont prouvé que la réduction de la fréquence joue un rôle important lors de l'exécution des requêtes gourmandes en mémoire et permet d'améliorer l'44 d'une manière significative. Pour choisir la fréquence du CPU adéquate pour l'exécution d'une requête Catena et al. [29] ont proposé l'outil PESOS (Predictive Energy Saving Online Scheduling). Ce dernier permet de faire une planification prédictive des économies d'énergie en ligne. En se basant sur les prédictions, qui permettent d'estimer le volume de traitement et le temps de traitement d'une requête avant son exécution, PESOS prend ses décisions d'ajustement en traitant les requêtes dans un délai imparti et grâce aux caractéristiques de planification de l'ordonnanceur. APESOS, qui est une amélioration de PESOS, est proposé dans [138].

Dans un système de base de données intégré, les auteurs dans [86] ont utilisé la technique du DVFS afin de développer une approche pour le traitement d'une transaction en temps réel exécutée simultanément avec d'autres transactions interférentes. Ensuite, ils ont combiné la technique du DVFS et l'adaptation dynamique de la cohérence temporelle des données (QoD) pour améliorer l'efficacité énergétique et pour accélérer le traitement des transactions [85]. Dans [188], les auteurs ont définit un coût énergétique pour chaque micro-opération du CPU (cache niveau L1, Cache niveau L2, Cache niveau L3, mémoire centrale), et en utilisant la technique du DVFS, ils ont développé une méthodologie pour l'évaluation précise du coût énergétique des différentes micro-opérations du CPU. Les expérimentations conduites sur trois

SGBDs (PostgreSQL, SQLite et MySQL) ont montré que le chargement/stockage du cache niveau L1 constitue un goulot d'étranglement énergétique. En se basant sur ces constatations, les auteurs ont proposé une architecture CPU L1 personnalisée permettant d'économiser l'énergie à travers la sélection du niveau DVFS le mieux adapté. Les résultats expérimentaux ont montré que 60% d'économies énergétiques peuvent être réalisées.

Dans un environnement multi-cœurs, Hayamizu et al. [74] ont développé une application qui a conduit a une réduction de 7,6% de la consommation d'énergie. Cette application ajuste d'une manière dynamique la fréquence de fonctionnement des cœurs en se basant sur un délai de réponse, fixé comme un accord de niveau de service (SLA), à travers une fonction de compromis entre l'énergie consommée et le délai de réponse.

Grâce à leur capacité de construire de matériel personnalisé à des coûts de développement relativement faibles, les réseaux de portes programmables sur site (*FPGA* [38]) sont utilisés à la place des processeurs classiques dans le but d'accélérer les applications de données intensives. Woods et al. [181] ont montré que les solutions à base de FPGA permettent de réduire l'énergie consommée tout en améliorant la performance. Cette technique a permis aux auteurs de réduire l'énergie d'un système de 3888 à 216 *Joules* pour certaines requêtes.

Les unités de traitement graphique (GPU) présentent une excellente alternative aux processeurs traditionnels grâce à leur meilleure performance, leur haut débit de traitement, et surtout leur économie d'énergie. Rofouei et al. [148] ont montré que l'utilisation d'un GPU est meilleure en termes de consommation énergétique, par rapport à une solution CPU classique, lorsque l'amélioration de la performance est supérieure à un certain seuil.

Le co-traitement des requêtes est un paradigme qui consiste à tirer profit des progrès matériels et des conceptions optimisées des algorithmes afin de réduire le temps d'exécution et l'énergie consommée des requêtes. Ce paradigme profite des avantages de chacun des dispositifs de traitement utilisés, il est de plus en plus adopté par les SGBDs en général et les SGBDs in-memory (en mémoire centrale) pour le traitement des opérations de jointure [36]. Les auteurs dans [77] ont effectué une étude comparative de la consommation moyenne d'énergie entre une architecture de CPU-GPU discrète et une architecture CPU-GPU couplée. Les résultats de cette comparaison montrent que l'approche discrète est entre 36% et 44% supérieures à ceux de l'architecture couplée. Dans la même tendance, Liu et al. [104] ont construit une architecture hétérogène, composée du *CPU*, *GPU*, et *FPGA* compatible à la technique de DVFS. Le régulateur DVFS est utilisé pour adapter la tension de fonctionnement des processeurs. Les paramètres dépendants du matériel dans le modèle énergétique proposé, peuvent être modifiés pour améliorer l'efficacité énergétique.

3.4.1.2 Les supports de stockage

Dans la section précédente, nous avons montré la grande importance apportée aux unités de traitement, parce qu'ils sont les majeurs consommateurs d'énergie. Cependant, l'amélioration de l'efficacité énergétique des supports de stockage et de la mémoire ont également attiré l'attention des chercheurs en raison de leurs capacités de progression significative au cours des dernières années. Pour minimiser la consommation énergétique des supports de stockage, les approches proposées cherchent à trouver un bon niveau de consolidation de la charge, d'améliorer les techniques de mise en cache et de pré-chargement et d'optimiser les modes d'énergie dans les réseaux de disques [176]. Tu et al. [176] ont proposé une approche qui se base sur le placement dynamique des données dans des fragments selon leur fréquence d'accès par les requêtes. Cette technique adapte dynamiquement le niveau de la fréquence des nœuds de fragments les moins accédés ce qui permet de réduire 50% de l'énergie consommée. De manière similaire, les auteurs dans [12] ont proposé une approche qui offre jusqu'à 60% d'économies d'énergie par le biais

38. Field-Programmable Gate Array

de l'ajustement dynamique de la fréquence en prenant des décisions en temps réel pour la commutation des disques en modes de faible puissance. Le modèle est intégré dans un SGBD et il est utilisé pour obtenir le compromis optimal entre le temps d'exécution et la consommation d'énergie d'une requête. Dans les travaux récents, les disques du type SSD (Solide State disk) sont également utilisés en tant que support de stockage des bases de données pour améliorer l'efficacité énergétique de ces dernières. Les SSD possèdent des propriétés proportionnelles à l'énergie, c'est à dire l'énergie consommée est proportionnelle aux opérations d'E/S effectuées par seconde [163]. Dans [156], les auteurs ont analysé le comportement énergétique des SSD en utilisant des bancs d'essai caractérisé par une forte intensité d'E/S. Les résultats de l'étude expérimentale menée, montrent l'efficacité des SSD par rapport aux disques durs classiques en termes de temps d'exécution et de consommation énergétique.

3.4.1.3 La mémoire

Avec l'émergence des bases de données en mémoire (In-Memory database) et des bases de données orientée colonnes, où le stockage des données se fait directement dans la mémoire, la mémoire est considérée comme un consommateur d'énergie important dans les systèmes informatiques, et la réduction de son énergie consommée est devenue un sujet actif de recherche. De nombreux travaux de recherche récents ont réalisé des études sur l'impact du dispositif de mémoire sur l'efficacité énergétique encouragés par l'étude de Appuswamy et al. [8] dans laquelle les auteurs soulignent que la croissance de l'utilisation des bases de données en mémoire va rendre la mémoire comme le majeur consommateur d'énergie au lieu de CPU. Comme le CPU et le disque dur, la mémoire principale possède elle aussi le pouvoir d'adapter sa puissance et sa fréquence en fonction de son utilisation.

Korkmaz et al. [97] ont proposé une méthode qui permet de réduire 40% de l'énergie consommée. Cette méthode consiste en une allocation de mémoire sur la base des rangs, ce qui permet au SGBD de déplacer les rangs inutiles de la mémoire aux états de faible puissance et réduire ainsi la consommation d'énergie globale. Cette approche n'est efficace que lorsque la taille de la base de données est inférieure à la quantité de la mémoire déjà allouée sur le serveur. Dans [72], les auteurs ont proposé une architecture de mémoire hybride afin de réduire la consommation d'énergie de la mémoire jusqu'à 80%. Cette architecture est composée à la fois de la mémoire dynamique à accès aléatoire (DRAM) et de la mémoire non Volatile (NVM). Une analyse et une collecte des statistiques d'accès à la mémoire, lors de l'exécution des applications, sont effectuées. Ensuite, et pour la prise de décision du placement des données sur une mémoire ou une autre, les auteurs ont utilisé une politique de gestion des données au niveau applicatif, et ce, en fonction de l'état du système.

Pour contrôler d'une manière adaptative la consommation de l'énergie, Kissinger et al. [95] ont intégré leur approche dans le fonctionnement du SGBD. L'outil développé se base sur des modèles de coût sensibles à l'énergie adaptatifs, permet d'optimiser à la fois la performance et l'énergie du SGBD tout en respectant les délais impartis du traitement des requêtes.

3.4.2 Les approches logicielles

Dans le domaine des technologies de l'information, les solutions d'EE développées par les académiques et les industriels se concentrent principalement sur le côté du matériel [164]. En effet, plusieurs chercheurs considèrent que les systèmes d'exploitation et les *firmwares* (les programmes du matériel) gèrent l'énergie et par conséquent, ils économisent l'énergie des processeurs de requêtes. Ceci est discutable, car le matériel est là pour exécuter des solutions logicielles qui incluent des implémentations logiques d'opérations de base de données (par exemple, la jointure, la sélection, le tri), des optimisations physiques (par exemple, les vues matérialisées, les indexes), et la décision d'utiliser leurs optimisations est généralement contrôlée

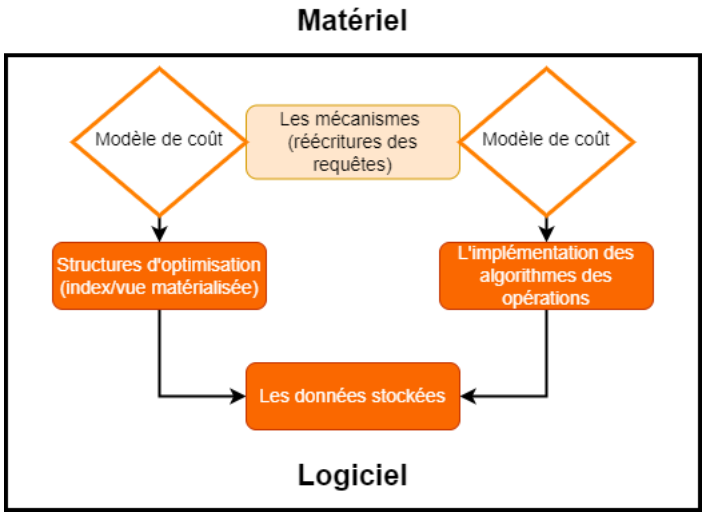


FIGURE 3.9 – Forte interaction entre les solutions matérielles et logicielles

par des modèles de coût mathématiques, ce qui rend la relation entre le matériel et le logiciel fortement liée comme illustré dans la figure 3.9. Dans cette section, nous présentons les approches logicielles permettant de réduire la consommation énergétique dans les bases de données en les divisant en des classes (Figure 3.10).

3.4.2.1 La définition des modèles de coût éco-énergétique

L'objectif principal des bases de données traditionnelles est la performance, ce qui signifie exécuter la requête le plus rapide possible sans prendre en compte le coût de l'énergie. Par conséquent, l'objectif des bases de données vertes a été déplacé vers le traitement des requêtes et l'optimisation des requêtes afin de réduire la consommation d'énergie. Pour réaliser cet objectif, l'optimiseur des requêtes doit être sensible à l'énergie pour tous les plans d'une requête et être capable de sélectionner le meilleur plan d'exécution qui économise de l'énergie pendant le traitement de la requête.

Dans la littérature, il existe une panoplie de modèles de coût énergétique pour modéliser et estimer la consommation énergétique d'une requête, et ces modèles se distinguent chacun par les paramètres utilisés, le niveau de modélisation, les techniques d'apprentissage et le type de la puissance mesurée. La plupart des modèles de coût énergétiques proposés prennent en paramètres l'utilisation du CPU et de la mémoire, et c'est dû généralement à la très grande consommation de ces deux composants. Les résultats trouvés dans [184] montrent que le processeur et la mémoire contribuent le plus à la puissance active (environ 99%).

Xu et al. [184, 185] ont définit un profil de puissance statique pour chaque opération de base de données pendant le traitement d'une requête. Les auteurs proposent un modèle de coût, qui se base sur celui du SGBD PostgreSQL, dans lequel pour chaque opération (le coût de la puissance du processeur pour accéder à un tuple, coût du disque pour la lecture/écriture d'une page) et en utilisant les différentes méthodes d'accès (séquentielle/indexée) et les opérations de jointures, le coût énergétique est calculé afin d'estimer le coût total du plan d'exécution. La technique de régression linéaire est utilisée afin d'identifier les paramètres de ce modèle. La formule de ce dernier est donnée dans l'équation 3.16.

$$\hat{E}_x = W_{cpu} \times N_{tuples}^{m} \times N_{tuples} + \bar{W}_{E/S} \times N_{pages} \tag{3.16}$$

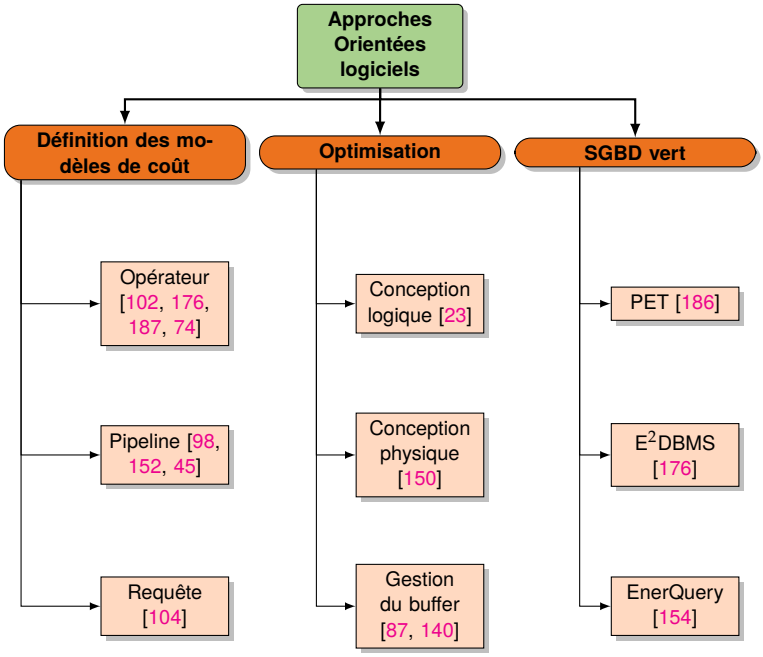


FIGURE 3.10 – Classification des approches d'EE orientées logiciels dans les bases de données.

où W_{cpu} and $\bar{W}_{E/S}$ sont les coefficients énergétiques d'un tuple traité par le CPU (N_{tuples}), et d'une page lue/écrite en mémoire (N_{pages}) respectivement, et m est le coefficient du modèle pour la consommation d'énergie du CPU.

Kunjir et al. [98] se sont intéressés à la modélisation de la puissance maximale des opération d'une requête en utilisant une segmentation en pipeline. Cette segmentation leur a permis de définir un modèle pour identifier les sources de cette consommation maximale et de pouvoir recommander des plans d'exécution avec une faible puissance. Pour chaque pipeline, les auteurs appliquent une fonction mathématique qui donne une estimation de la puissance maximale grâce aux taux et aux tailles des données circulant à travers les opérations de pipelines qui sont données en entrée. Ensuite, les pipelines ayant une consommation maximale sont remplacés, dans la mesure du possible, par d'autres ayant une consommation inférieure. La technique de la régression multiple affine par morceaux est appliquée pour identifier les paramètres du modèle. Par exemple, pour un nœud B (accès séquentiel), le débit est calculé comme suit :

$$Rate_D = \frac{EntreB}{TempsDisque_B} \tag{3.17}$$

où $EntreB$ indique la taille des données extraites du disque par B et $TempsDisque$ est le temps de transfert du disque.

Lang et al. [100] ont proposé un framework pour le traitement éco-énergétique des requêtes. Ce travail étend les plans d'exécution produits par l'optimiseur de requêtes traditionnel en ajoutant une prédiction de la consommation d'énergie pour certains opérateurs d'une base de données (i.e. la sélection, la projection et la jointure). La technique de régression linéaire est également utilisée pour l'identification des paramètres du modèle. Ce dernier, permet d'estimer la puissance moyenne du système pendant l'exécution d'une requête. Trois types paramètres constituent ce modèle :

1. le temps T

2. les caractéristiques de la requête Q : lecture d'un page (R_Q), écriture d'une page (W_Q), nombre d'instructions de processeur (I_Q), et nombre d'accès au mémoire (M_Q)
3. les caractéristiques du système : lecture disque (C_R), écriture disque (C_W), accès mémoire (C_{mm}), CPU (C_{cpu}), reste du système (C_autres)

Le modèle est défini comme suit :

$$P_{moy}Q = C_{cpu} \times \frac{I_Q}{T} + C_R \times \frac{R_Q}{T} + C_W \times \frac{W_Q}{T} + C_{mm} \times \frac{M_Q}{T} + C_{autres} \tag{3.18}$$

L'idée de ce modèle est de calculer la somme de la puissance moyenne du processeur ($C_{cpu} \times \frac{I_Q}{T}$), de la lecture du disque ($C_R \times \frac{R_Q}{T}$), de l'écriture sur disque ($C_W \times \frac{W_Q}{T}$), de la mémoire ($C_{mm} \times \frac{M_Q}{T}$) et du reste du système (C_{autres}) pendant l'exécution de la requête Q.

Roukh et al. [152] ont proposé un modèle de coût permettant d'estimer la consommation d'énergie des requêtes dans une base de données relationnelle centralisée. Comme dans [98], les auteurs optent pour une segmentation en pipeline du plan d'exécution. Pour évaluer les opérations de chaque pipeline, les auteurs utilisent le coût du CPU et du coût d'E/S nécessaires pour chaque opération. Pour identifier les paramètres du modèle, la technique de la régression polynomiale est utilisée. Le modèle est défini de la manière suivante :

$$Puissance(Q) = \frac{\sum_{j=1}^{p} Puissance(PL_j) * Temps(PL_j)}{Temps(Q)} \tag{3.19}$$

où p est le nombre de pipelines de la requête Q, Temps(PL_j) et Temps(Q) représentent respectivement, le temps d'exécution du pipeline j et de la requête Q. Un pipeline implique plusieurs opérations algébriques, ainsi, le coût énergétique d'un pipeline j noté ($puissance(PL_j)$) est calculé en fonction des coûts énergétique de ses opérations comme le montre la formule de l'équation suivante :

$$Puissance(PL_j) = \beta_{cpu} \times \sum_{k=1}^{n_j} CPU\_COST_k + \beta_{io} \times \sum_{k=1}^{n_j} ES\_COST_k \tag{3.20}$$

où n_j représente le nombre d'opérations du pipeline j, β_{cpu} et β_{io} sont les paramètres du modèle et le ES_COST est le nombre prévu d'entrées/sorties nécessaires pour exécuter un opérateur spécifique. Le CPU_COST est le nombre prévu d'instructions CPU que le SGBD a besoin d'effectuer pour exécuter un opérateur spécifique.

Guo et al. [65] ont intégré le coût énergétique de la mémoire centrale dans leur modèle de coût. Dans leur article, les auteurs ont présenté une étude approfondie sur l'impact de la taille de la mémoire centrale sur la consommation d'énergie en analysant les trois structures de mémoires caches du SGBD Oracle (Database Buffer Cache, Dictionary Cache et Library Cache). Pour identifier les paramètres de leur modèle décrit par l'équation 3.21, les auteurs ont utilisé la technique de régression linéaire simple.

$$E = N_{cpu}(W_{cpu} \times T_{cpu}) + K_m(W_m \times T_m) + K_s(W_s \times T_s) + K(W_k \times T_k) \tag{3.21}$$

où W_{cpu} est le coefficient de la puissance de chaque 10000 instructions traitées par le CPU. N_{cpu} est le nombre de chaque 10000 instructions traitées par le CPU, T_{cpu} est le temps du processeur pour traiter les instructions, K_s est le nombre d'opérations effectuant une seule lecture disque, K_m est le nombre d'opérations faisant multiples lectures disque, et K est le nombre de blocs de données accessibles par mémoire.

Dans [44, 45], Dembélé et al. se sont intéressés à l'estimation de l'énergie dans les *SGBD* multi-coeurs. En partant des travaux de [152], les auteurs utilisent une segmentation en pipelines, en divisant les pipelines d'un plan d'exécution d'une requêtes en deux groupes dénotés par : (1)

pipeline séquentiel (PL) et (2) pipeline parallèle (PPL). Un pipeline est dit séquentiel lorsqu'il ne possède pas d'opérations traitées en mode parallèle. La formule pour l'estimation de la puissance du pipeline séquentiel est donnée dans l'équation 3.22 et celle du mode parallèle est donnée dans l'équation 3.23. Pour estimer la puissance d'un pipeline en parallèle, les auteurs ont introduit un paramètre nommé facteur énergétique (fc) qui permet d'exprimer la différence entre la puissance consommée par le CPU lors du traitement d'une requête en mode parallèle par rapport au traitement en mode séquentiel. La formule décrite dans l'équation 3.24 permet de calculer ce paramètre, qui est estimé en fonction du degré de parallélisme, qui est défini par le nombre de nœuds (processeurs ou processus) utilisés pour exécuter un plan.

$$Puissance(PL_i, 1) = W_{CPU} * \sum_{k=1}^{n} C_{CPU_k} \oplus W_{E/S} * \sum_{k=1}^{n} C_{E/S_k} \tag{3.22}$$

$$Puissance(PPL_i, DoP) = (fc_{DoP} + 1) \times W_{CPU} \times \sum_{k=1}^{n} C_{CPU_k} \oplus W_{E/S} \times \sum_{k=1}^{n} C_{E/S_k} \tag{3.23}$$

où DoP est le degré de parallélisme, W_{CPU} et $W_{E/S}$ présentent les unités énergétiques pour une instruction du CPU et une opération de lecture ou d'écriture sur le disque respectivement. C_{CPU_k} est le nombre d'instructions exécutées par le CPU et C_{E/S_k} est le nombre d'accès au disque soit en lecture ou en écriture. n définit le nombre d'opérateurs dans le pipeline i et k est l'indice de sommation, et $\oplus$ exprime la corrélation entre les paramètres.

$$fc_n = \frac{P_n - P_0}{P_0} \tag{3.24}$$

où P_n exprime la puissance moyenne consommée par les n cœurs du CPU pour traiter le plan parallèle de la requête.

Plus formellement, la formule de l'équation 3.25 permet de calculer l'énergie consommée par un plan parallèle d'une requête Q avec n pipelines séquentiels et m pipelines parallèles.

$$Energie(Q, DdP) = \sum_{i=1}^{n_0} Energie(PL_i, 1) + \sum_{i=1}^{m_0} Energie(PPL_i, DdP) \tag{3.25}$$

où DdP définit le degré de parallélisme, et m_0 et n_0 présentent le nombre de pipelines séquentiels et parallèles respectivement, dans le plan de la requête.

Afin d'améliorer la précision de leurs estimations, les auteurs ont proposé un nouveau modèle dans lequel ils prennent en considération la source de chargement des données lors du traitement de la requête.

$$Puissance(PL_i) = W_{CPU} * \sum_{k=1}^{n} C_{CPU_k} \oplus W_{mem} * \sum_{k=1}^{n} C_{mem_k} \oplus W_{E/S} * \sum_{k=1}^{n} C_{E/S_u} \tag{3.26}$$

où W_{CPU}, W_{mem} et $W_{E/S}$ définissent respectivement les unités énergétiques pour une instruction du CPU, une opération de lecture ou bien d'écriture sur disque et une opération de lecture ou d'écriture en mémoire cache. Le C_{CPU_k} est le nombre d'instructions exécutées par le CPU. C_{E/S_k} est le nombre d'accès au disque soit en lecture ou en écriture. C_{mem_u} est le nombre d'accès à la mémoire principale soit en lecture ou en écriture. La variable n définit le nombre d'opérateurs dans le pipeline i et k est l'indice de sommation. $\oplus$ exprime la corrélation entre les paramètres.

Bien que les travaux précédemment présentés considèrent seulement une architecture centralisée de base de données, Rodriguez-Martinez et al. [147] se sont intéressés à la modélisation

Tableau 3.1 – Les modèles de coût énergétique

Approche	Puissance	Paramètres	Niveau de modélisation	Technique de régression
Xu et al. [184]	Moyenne	CPU+Disque	Opérateur	Régression linéaire simple
Kunjir et al. [98]	Maximale	CPU+Disque	Pipeline	Régression linéaire multiple
Rodriguez-Martinez et al. [147]	Maximale	CPU+Disque	Requête	Régression linéaire multiple
Lang et al. [100]	Moyenne	CPU+Disque+RAM	Opérateur	Régression linéaire simple
Xu et al. [185]	Moyenne	CPU+Disque	Opérateur	Régression linéaire simple
Roukh et al. [152]	Moyenne	CPU+Disque	Pipeline	Régression polynomiale multiple
Guo et al. [65]	Moyenne	CPU+Disk+RAM	Requête	Régression linéaire simple
Dembélé et al. [45]	Moyenne	CPU+Disque+RAM	Pipeline	Régression polynomiale

de la puissance maximale au niveau des requêtes dans un environnement cluster. Ils se sont concentrés sur l'estimation de l'énergie des requêtes de sélection simples sur les relations individuelles en extrayant les caractéristiques communes des requêtes (comme le facteur de sélectivité et la cardinalité). La technique de régression linéaire multiple est utilisée pour calculer les paramètres du modèle. La formule de l'équation 3.27 permet d'estimer la consommation énergétique d'une requête de sélection Q sur une table R.

$$E_Q = \beta_0 + \beta_1\ S + \beta_2\ SF_p(R)\ S + \beta_3\ |R|\ SF_p(R) + \beta_4\ |R| < R > SF_p(R) \tag{3.27}$$

où les β_i sont les coefficients de régression et $|R|$, $< R >$, $SF_p(R)$ et S sont respectivement : la cardinalité de la relation R, le nombre de colonnes de R, la sélectivité du prédicat p dans R et le nombre de serveurs utilisés.

Dans le même contexte, les auteurs dans [99] ont proposé un modèle de coût énergétique qui prend en compte les configurations du serveur (les unités de traitement, les disques et la bande passante du réseau) et les paramètres de traitement relatives à la base de données (la sélectivité des prédicats, la taille de la table de hachage). Ce modèle permet de prédire les performances de traitement des données et l'énergie consommée par un opérateur de jointure par hachage. L'erreur maximale rapportée est inférieure à 10%.

La majorité de ces modèles de coût donne la main à l'administrateur d'orienter l'optimiseur de requêtes afin de choisir le plan d'exécution le plus écologique ou bien de trouver un certain compromis entre l'énergie et la performance en utilisant des variables de pondération. Le tableau 3.1 récapitule les différents modèles de coûts proposés en précisant leurs paramètres et leurs techniques d'apprentissage automatique ainsi que leur niveau de modélisation.

L'estimation de l'énergie des plans d'exécution d'une requête et choisir le moins énergivore n'est pas la seule utilité des modèles de coût. Ils sont utilisés dans des différentes techniques ayant le même objectif. Le mécanisme QED (Improved Query Energy-efficiency by Introducing Explicit Delays) proposé par Lang et al. [102], introduit des retards explicites, en exploitant les caractéristiques communes dans la charge de travail. Ce retard permet de grouper les requêtes dans une file d'attente dans la mémoire tampon, ensuite, ces requêtes sont évaluées par des techniques d'optimisation multi-requêtes afin de trouver l'ordre plus efficace de les exécuter. Psaroudakis et al. [140] et Kang et al. [87] se sont intéressés à l'ordonnancement éco-énergétique. Les auteurs dans [140] ont montré que l'exécution en parallèles de certains opérateurs SQL basiques (l'agrégation et les algorithmes d'accès aux données) est quatre fois meilleure en termes d'efficacité énergétique.

Dans la section suivante, nous présenterons l'intégration des modèles de coût dans l'optimiseur de requêtes de quelques $SGBD$s open source afin de développer des $SGBD$s verts.

3.4.2.2 Les SGBD éco-énergétique

Après la définition de leurs modèles de coût énergétique, de nombreux chercheurs ont essayé de les intégrer dans des SGBD afin de proposer des systèmes de gestion de bases de données sensibles à l'énergie.

En se basant sur son modèle de coût proposé dans [184], qui estime la consommation énergétique des opérateurs relationnels, Xu et al. [186] ont proposé le SGBD sensible a l'énergie PET (Power-Energy-Time), avec une interface graphique permettant de définir l'objectif d'optimisation et le degré de concurrence, d'afficher les informations détaillées de la requête en cours d'exécution, telles que l'arborescence du chemin d'exécution et son coût en énergie et en performance, et de visualiser l'utilisation de la puissance de l'exécution de la requête et d'autres statistiques liées à l'énergie.

Un autre SGBD éco-énergétique est proposé dans [176]. E^2DBMS atteint une efficacité énergétique élevée via deux stratégies : modifier les modèles de consommation des ressources, et contrôler les modes d'alimentation du matériel. Les auteurs utilisent une approche basée sur le DVFS pour réduire la consommation énergétique du processeur, dans laquelle ils ajustent dynamiquement le niveau DVFS du CPU en fonction du plan d'exécution choisi pour une requête, ou bien des performances du SGBD. Les stratégies sont mises en œuvre via une série de mécanismes pour cibler les deux principaux consommateurs d'énergie dans un serveur de base de données : le CPU et le disque. Les expérimentations réalisées sur un premier prototype de E^2DBMS et la plate-forme de simulation montrent d'importantes économies d'énergie.

Roukh et al. [154] ont intégré leur modèle de coût proposé dans [152], précédemment détaillé, dans l'optimiseur des requêtes de SGBD PostgreSQL afin de développer le SGBD sensible à l'énergie appelé Ener-Query [39]. Cet outil, permet à l'administrateur d'estimer le coût d'exécution d'une requête en termes d'énergie consommée, puissance moyenne, nombre de cycles de CPU ainsi que le nombre d'opérations E/S. Ener-Query donne également la possibilité à l'administrateur d'orienter l'optimiseur afin de choisir le plan d'exécution selon une préférence (orienté énergie/orienté performance/compromis performance-énergie).

3.4.2.3 La conception logique & la conception physique éco-énergétiques

Les approches et les techniques proposées dans les sections précédentes considèrent uniquement des bases de données déjà déployées ayant un schéma fixe. Néanmoins, avant de déployer une base de données, il est nécessaire de la concevoir et de définir les tables et ses attributs afin d'avoir le schéma qui évolue au cours du temps. Cette étape est appelée la conception logique. Il est possible de détecter l'énergie à ce stade du cycle de vie d'une BD. Malgré l'importance de cette étape, un seul travail s'est intéressé à la gestion de l'énergie de la conception logique. Bouarrar et al. [23] ont étudié la variabilité du schéma logique d'une BD et son impact sur la consommation d'énergie. Dans leur travail, ils ont généré tous les schémas logiques possible à partir d'un schéma initial et pour chacun d'entre eux, ils ont estimé sa consommation d'énergie. Les résultats ont prouvé que l'énergie peut être détectée et améliorée au niveau de la conception logique et que le schéma en étoile est loin d'être le schéma optimal en termes d'énergie et de performance.

La conception physique est la phase cruciale du cycle de vie de la base de données puisqu'elle peut être vue comme un entonnoir des phases précédente [23], puisqu'elle intègre les entrées et les paramètres des premières phases (étapes conceptuelles et logiques). Les structures d'optimisations telles que les indexes, les vues matérialisées, et la fragmentation horizontale/verticale, permettent de réduire considérablement le temps d'exécution. Le bon choix de ces structures permet d'améliorer les performances d'un schéma logique non-optimisé. Comme présenté dans le chapitre précédent (Cf. section 2.5.1), les structures d'optimisation ont été largement étudiées

39. https://forge.lias-lab.fr/projects/ecoprod/wiki/Documentation

dans la littérature, et plusieurs approches ont été présenté. Malgré l'importance vitale de la conception physique, très peu de travaux s'y sont consacrées pour étudier son impact énergétique. Dans [153], les auteurs ont proposé une approche multi-objectifs de sélection des vues matérialisées qui offre le compromis entre la performance et la consommation d'énergie des requêtes. Et dans [110] les auteurs ont comparé l'efficacité énergétique d'un *SGBD* relationnel (MySQL) avec deux autres *SGBD*s NoSQL(Cassandra et MongoDB). Ils ont étudié le comportement énergétique des trois *SGBD*s pour les opérations d'insertion, sélection et suppression ainsi que les indexes. Les résultats obtenus montrent que l'efficacité énergétique peut être améliorée de manière significative pour tous les systèmes de gestion de bases de données étudiés sans dégrader les performances, et que l'efficacité énergétique n'est pas toujours linéaire par rapport au temps d'exécution.

3.4.3 Bilan

En penchant sur la littérature, nous trouvons que les technologies actuelles d'économie d'énergie dans les SGBD se sont généralement focalisées sur le matériel et les solutions logicielles. Du côté du matériel, les travaux existants se sont concentrés sur l'utilisation de nouvelles capacités du matériel moderne, telles que les processeurs modernes, qui traitent à une tension et une fréquence plus faibles, les supports de stockage du type SSD, et la mémoire. Ces travaux proposent des méthodes de conservation de l'énergie en désactivant les composants électroniques pendant les périodes d'inactivité ou bien la modification de leurs performances (fréquence et voltage) pour une meilleure efficacité énergétique. Des approches se basant sur l'utilisation des GPU et le co-traitement sont également proposée. Du côté logiciel, la plupart des travaux se sont concentrés sur le traitement des requêtes à travers la définition des modèles de coût énergétiques pour prédire la consommation de l'énergie des requêtes et choisir le plan d'exécution le plus écologique, ou bien la proposition des techniques d'optimisation basées sur les modèles de coûts pour réduire la consommation tel que des structures d'optimisation sensible à l'énergie ou même modifier les optimiseurs des SGBD pour proposer des SGBD énergétiques. L'intégration de la dimension énergétique dans les bases de données a modifié le rôle de l'optimiseur de requêtes traditionnel, qui était principalement la maximisation des performances de la requête. Le nouvel objectif de l'optimiseur est maintenant de trouver des plans d'exécution qui ont des performances acceptables, mais qui consomment le moins d'énergie possible.

La communauté des bases de données a eu des points de vue controversés à propos la relation entre la performance et l'énergie, de sorte que l'optimisation des requêtes énergétiques ne vaut pas la peine. D'après Tsirogiannis et al. [175] l'efficacité énergétique va main dans la main avec la performance, et ils affirment qu'il y a presque toujours un plan d'exécution qui peut améliorer l'efficacité énergétique en améliorant simplement les performances et par conséquent, le plan plus performant est le plus économe en énergie. Néanmoins, les auteurs dans [184] et [98], ont montré que l'exécution des requêtes le plus rapide possible n'est pas toujours le moyen le plus économe en énergie. Dans le travail de [99], les auteurs se sont demandés si le traitement (optimisation et exécution) de la requête aussi rapidement que possible, est-il toujours le moyen le plus économe en énergie pour exploiter un SGBD. Ils sont parvenus à une réponse négative en raison des caractéristiques opérationnelles typiques du serveur et les caractéristiques de puissance/performance des composants matériels des serveurs modernes. Enfin, Guo et al [65] sont arrivés à la conclusion que la relation entre la consommation d'énergie et la performance du SGBD est controversée.

La majorité des travaux antérieurs ont tenté de réduire l'énergie de la base de données en traitant uniquement le problème de traitement des requêtes. Les chercheurs se sont concentrés sur la définition de modèles de coût énergétiques pour prédire la consommation d'énergie et proposer des plans de requête économes en énergie. Malgré la grande importance des modèles de coût et leur rôle dans la réduction de l'énergie, néanmoins, il existe plusieurs d'autres

composantes dans la base de données qui peuvent contribuer à l'amélioration de l'EE. Nous avons constaté dans cette section que les chercheurs ont négligé l'impact de deux aspects très important du cycle de vie des bases de données, i.e., la conception logique et les structures d'optimisation, où, seulement deux approches ont été proposées.

Dans la suite de notre thèse, nous mettons en exergue l'effet de ces deux aspects sur l'amélioration de l'EE en proposant des approches pour la détection et la réduction de l'énergie lors de la phase de la conception logique et la sélection éco-énergétique des structures d'optimisation dans la phase de conception physique.

3.5 Conclusion

L'énergie est un produit de première nécessité, ce qui rend son économie la responsabilité de tous. Les centres de données sont considérés comme le majeur consommateur d'énergie, ce qui a incité la communauté informatique à réduire sa consommation et a produire des solutions et des approches permettant d'améliorer l'efficacité énergétique. Les vendeurs des appareils informatique utilisent, désormais, la faible consommation des appareils en tant qu'un argument de vente. C'est ainsi que le problème d'optimisation a convergé de l'amélioration de la performance traduite par la réduction du temps d'exécution vers la satisfaction de deux besoins non-fonctionnels, cruciaux et conflictuels à savoir : (i) un traitement rapide (ii) une consommation d'énergie réduite des SIs, pour satisfaire les contraintes écologiques fixées pour sauver notre planète.

Ce chapitre nous a permis de définir les notions de base relatives à l'énergie, de présenter les différentes méthodes pour mesurer/estimer l'énergie et de détailler les approches utilisées par les industriels et les chercheurs pour la réduction de la consommation de l'énergie dans les SI, allant du niveau des composants matériels au niveau logiciel en passant par les plate-formes de déploiement, où nous avons classé les techniques appliquées au niveau du matériel en deux catégories, (1) dispositif de traitement et, (2) la gestion du stockage. Dans ce chapitre, nous avons également établit une revue de l'état de l'art des techniques d'optimisation de l'EE dans les bases de données. Les études précitées dans chaque catégorie ont en effet apporté une contribution importante à la conservation de l'énergie. Ainsi, cette étude nous a permis de repérer quelques axes de recherche qui n'ont pas été très explorés par la communauté des bases de données et qui pourront être prometteurs et permettront de satisfaire la contrainte énergétique.

Notre objectif dans cette thèse est donc de fournir des approches permettant d'améliorer l'efficacité énergétique des bases de données, à un niveau plus fin, sans impacter le temps d'exécution, et en utilisant uniquement des approches logicielles.

Nous proposons d'intégrer l'énergie dans la phase de la conception physique à travers une approche de sélection multi-objectifs d'une structure d'optimisation. Ensuite, nous proposons également de capter l'énergie à un niveau supérieur, en intégrant cet aspect dans la phase de la conception logique et voir son impact sur l'exécution des requêtes. La méthodologie, les étapes et l'implémentation de nos contributions sont détaillées dans la partie suivante de cette thèse aux chapitres 4 et 5.

Troisième partie

Contributions

Chapitre 4

La sélection éco-énergétique des Indexes de Jointure Binaires

4.1 Introduction

La conception physique est l'étape de la plus grande importance pendant le cycle de vie d'une base de données, puisque le bon choix des structures d'optimisations peut dissimuler les problèmes de performance causés par un schéma logique non optimisé comme nous l'avons expliqué dans le premier chapitre. Dans ce chapitre, nous visons à intégrer la dimension énergétique dans cette phase et nous nous sommes plus précisément focalisés sur le problème de sélection d'une structure d'optimisation redondante appelée : Index de jointure binaire (cf. section 2.5.1.1.3).

Pour résoudre ce problème, nous introduisons une approche multi-objectif pour la sélection des IJB dans laquelle nous visons à optimiser le temps d'exécution et l'énergie consommée. Pour ce faire, nous avons commencé par effectuer un audit pour comprendre le comportement énergétique de cette structure d'optimisation. Nous avons également proposé une nouvelle formalisation multi-objectif du problème de sélection des IJB, en tenant en compte deux besoins non-fonctionnels : la performance et la consommation d'énergie.

La suite de ce chapitre est organisée comme suit. Dans la première section, nous présentons l'audit que nous avons réalisé sur les IJB. Dans la section suivante, nous commençons par la définition des concepts fondamentaux sur lesquelles se base notre approche, ensuite nous décrivons notre méthodologie qui se base sur une modélisation par un hypergraphe de la charge de travail, les traverses minimales pour déterminer les attributs candidats, et sur l'opérateur de Skyline pour la sélection multi-objectif, en détaillant chaque étape. Un exemple illustratif est également fourni pour mieux comprendre notre démarche. L'algorithme permettant d'implémenter notre approche est donné par la suite. Jute après, nous présentons et discutons les résultats expérimentaux. Enfin, la dernière section permet de conclure le chapitre en résumant les principaux résultats.

4.2 L'audit des Indexes de Jointure Binaires

L'étude de l'efficacité énergétique des IJB nécessite leur audit afin d'identifier les conditions énergétiques idéales de leur utilisation. Notre intuition est que les attributs indexables ont un impact différent sur l'EE, puisque chacun a sa propre cardinalité et est associé à une table différente qui participe à l'exécution d'une requête. Cet audit doit prendre en compte d'autres classes de requêtes en plus des requêtes idéales pour les IJB (comme COUNT), en utilisant des requêtes plus complexes. En ce qui suit, nous résumons le fruit de nos efforts expérimentaux.

Considérons la requête Q4.3 du benchmark (SSB) (ayant une table de faits *Lineorder*, et 4 tables de dimensions : *Date (D)*, *Customer (C)*, *Part(P)* et *Supplier (S)*) déployés sur le SGBD Oracle12c. Notre choix de cette requête est basé sur la présence de plusieurs jointures (4), de plusieurs sélections (4) et un nombre raisonnable d'attributs indexables.

```
SELECT d_year, s_city, p_brand,
       sum(lo_revenue) - sum(lo_supplycost) AS profit
FROM lineorder
LEFT JOIN dates ON lo_orderdate = d_datekey
LEFT JOIN customer ON lo_custkey = c_custkey
LEFT JOIN supplier ON lo_suppkey = s_suppkey
LEFT JOIN part ON lo_partkey = p_partkey
WHERE c_region = 'AMERICA' AND s_nation = 'UNITED␣STATES'
AND (d_year = 1997 OR d_year = 1998)
AND p_category = 'MFGR#14'||p_mfgr ='MFGR#1'||p_brand ='MFGR#1414'
GROUP BY d_year, s_city, p_brand
ORDER BY d_year, s_city, p_brand;
```

Tableau 4.1 – Les statistiques de Q4.3

	Temps (sec)		Energie (joule)	
	Sans IJB	Avec IJB	Sans IJB	Avec IJB
p_mfgr	75,93	144,83	4,269,851	9,598,225
p_category	75,53	166,97	4,157,719	9,304,893
p_brand	63,25	43,63	3,556,226	2,202,657
	Puissance Moyenne (watt)		Puissance Maximale (watt)	
	Sans IJB	Avec IJB	Sans IJB	Avec IJB
p_mfgr	55,452	58,525	58,905	63,246
p_category	54,706	55,058	67,192	59,799
p_brand	55,566	50,06	63,012	63,891

Comment procédons-nous? La requête *Q*4.3 est composée de quatre attributs indexables (*c_region, s_nation, d_year et p_category*) appartenant à des tables de dimension différentes, avec les cardinalités suivantes 5, 25, 7 et 25 respectivement. Tout d'abord, nous avons réalisé une expérience dans laquelle nous avons exécuté la requête en absence des IJB. Ensuite, nous avons créé quatre IJB différents (chacun correspond à un attribut indexable). Nous ré-exécutons notre requête quatre fois et à chaque fois un IJB est utilisé. Pour chaque exécution, le temps de traitement des requêtes et le coût de consommation d'énergie sont capturés.

Après avoir examiné les plans d'exécution de toutes les différentes requêtes, nous avons remarqué que l'optimiseur de requêtes ignore tous les IJB créés, de sorte que son modèle de coût ne les considère pas comme une structure d'optimisation pertinente. Nous insistons pour voir le comportement des IJB en forçant l'optimiseur à les utiliser à travers les *Hints* [1] proposés par notre SGBD cible.

Les résultats obtenus sont les pires en termes de temps d'exécution et de consommation d'énergie. Sur la base de ces résultats, nous avons conclu que les attributs indexables utilisés ne sont pas adaptés à nos deux besoins non-fonctionnels.

Face à cette situation, nous avons décidé de varier légèrement notre requête Q4.3 en générant deux variantes (à l'aide du signe ||) en substituant son prédicat de sélection défini sur la table *Part* par deux autres (*p_mfgr ='MFGR #1' et p_brand ='MFGR#1414'*). La cardinalité de ces attributs est respectivement de $5 et 1000$. Ensuite, nous avons exécuté les deux requêtes en mode sans indexes et en présence des IJB.

Les statistiques obtenues pour le temps d'exécution, la consommation d'énergie, la puissance moyenne et maximale de la requête sont respectivement affichées dans le tableau.4.1. Pour les attributs de faible cardinalité (*p_mfgr, p_category*), l'optimiseur ignore toujours l'utilisation de l'IJB créé. Cependant, pour l'attribut de cardinalité relativement élevée *p_brand*, l'optimiseur s'appuie sur l'IJB lors de la construction du plan d'exécution.

En effet, l'utilisation de l'IJB crée sur de l'attribut *p_brand* améliore considérablement la performance et l'énergie consommée, par une forte réduction de 31% et 38% respectivement. De l'autre côté, si nous ne forçons pas l'utilisation de ces indexes via le *Hint*, dans ce cas, l'optimiseur utilise le mode sans index et ignore les indexes créés sur *p_mfgr* et *p_category*. En revanche, l'IJB créé sur *p_brand* est utilisé même sans l'intervention du *Hint*. De plus, nous avons remarqué que la puissance maximale est plus nette chaque fois que la requête utilise l'index, ce qui augmente la puissance moyenne lors du traitement de la requête. De plus, le plan d'exécution de la requête change ainsi que l'*ordre de jointure*. Dans un premier temps, l'optimiseur décide d'utiliser l'IJB,

1. Le hint est un commentaire qui indique à l'optimiseur du système de gestion de base de données comment exécuter une requête.

puis effectue les autres opérations de jointure/sélection, par contre, l'opération de projection des autres attributs appartenant à la même table de l'attribut déjà indexé, déclenche de lourds balayages supplémentaires de table.

Analyse. Sur la base des expériences ci-dessus, nous avons réalisé que le choix des attributs indexables joue un rôle crucial dans la satisfaction de nos deux besoins non-fonctionnels. Un autre point concerne les résultats obtenus par l'IJB défini sur l'attribut *p_brand*, est qu'un IJB seul n'est pas suffisant pour exécuter notre requête (avec d'autres opérations telles que la somme, l'agrégation et le tri) car d'autres accès à la table des faits et à la table de dimension elle-même sont nécessaires.

Le cas idéal pour optimiser une telle requête en présence d'un IJB correspond à *quelques instances* de table des faits pointées par le fragment représenté par le prédicat *p_brand= 'MFGR#1414'*. Nous appelons cette quantité FAN-OUT(p_brand= 'MFGR#1414'). Son principe est assez similaire au FAN-OUT utilisé dans l'index B-tree [64].

Plus formellement, le FAN-OUT d'un prédicat donné Pr défini sur la valeur d'un attribut indexable A_j, est donné par l'équation suivante :

$$FAN-OUT(Pr\_A_j) = \frac{||Table\ des\ Faits \ltimes \sigma_{Pr\_A_j}(TD_A)||}{||Table\ des\ Faits||} \tag{4.1}$$

où $||\ ||$, Pr_A_j, TD_A et $\ltimes$ représentent le nombre d'instances, le prédicat sur l'attribut indexable, la table de dimension de l'attribut indexable et l'opération de semi-jointure.

En résumé, les conditions idéales pour les IJB sont les opérations de sélections contenant *les prédicats des attributs ayant une valeur de FAN-OUT ne dépassant pas un seuil faible.* Connaître les informations sur le FAN-OUT de tous les attributs indexables contribuera fortement à définir des stratégies vertes pour sélectionner les IJB.

4.3 G-BJI : une approche verte de sélection des indexes de jointures binaires

Dans cette section, nous présentons notre approche verte de sélection des IJB. Avant de le faire, nous proposons une nouvelle formalisation du $PSIJB$. La formalisation traditionnelle (orientée performance) du problème de sélection des indexes de jointure binaires consiste à choisir une configuration d'indexes qui minimise le temps de réponse global de la requête et satisfait les contraintes fixées, comme nous l'avons défini dans la section 2.5.1.1.3. Dans cette section, nous introduisons une nouvelle formalisation du $PSIJB$.

4.3.1 Une nouvelle formalisation du problème de sélection d'indexes

Dans cette formalisation, nous incluions le besoins non-fonctionnel énergétique lors de la sélection de la configuration finale d'indexes, et par conséquent, le $PSIJB$ devient $PSEIJB$ (problème de sélection éco-énergétique des indexes de jointure binaires) comme illustré dans la figure 4.1. Ainsi, la formalisation du $PSEIJB$ peut être présentée comme suit :

étant donné :

(a) un entrepôt de données relationnel avec une table de faits TF et L tables de dimensions $Dim = \{Td_1, \ldots, Td_L\}$,

(b) une charge de travail avec m requêtes $Q = \{Q_1, Q_2, \ldots, Q_m\}$, et

(c) un ensemble de contraintes telles que le coût de stockage et le coût de maintenance,

(d) un ensemble de besoins non-fonctionnels à optimiser tels que l'énergie et la performance.

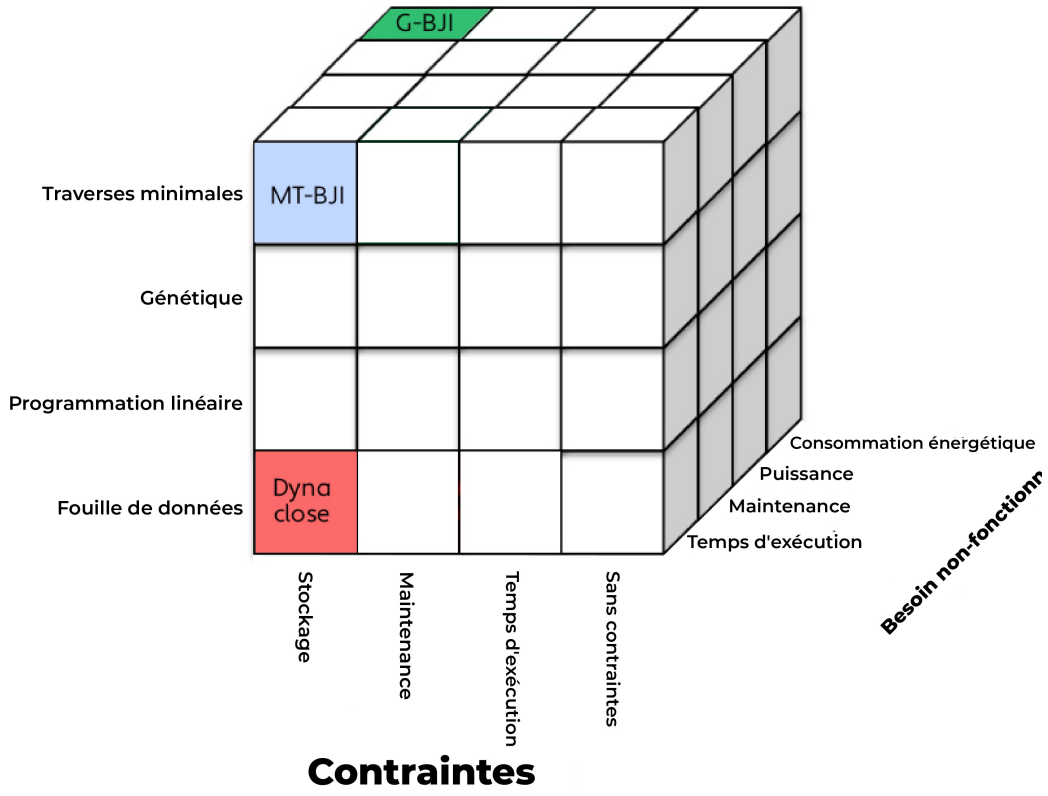


FIGURE 4.1 – Le cube du problème de sélection d'*IJB*

Le *PSEIJB* vise à sélectionner une configuration d'indexes qui permet de *minimiser, à la fois, le temps d'exécution et la consommation d'énergie globale* de la charge de travail, et de satisfaire les contraintes fixées.

4.3.2 Fondements théoriques

Pour simplifier la compréhension de notre approche proposée, nous commençons par présenter deux concepts fondamentaux sur lesquels reposent fortement notre approche : l'hypergraphe qui nous permettra de modéliser la charge de travail et l'opérateur de Skyline qui nous sera utile pour la sélection multi-objectif.

4.3.2.1 La théorie des hypergraphes

La théorie des hypergraphes est une généralisation de la théorie de graphe en définissant la notion de l'hyperarête. Contrairement aux arêtes classiques qui ne peuvent joindre que deux sommets, une hyperarête peut contenir un, deux ou plusieurs sommets [80]. Théoriquement, les hypergraphes permettent de généraliser plusieurs théorèmes de graphes, voire d'en factoriser plusieurs en un seul. Ils sont parfois préférés aux graphes, et ce, dû à leur aptitude à mieux modéliser certains types de contraintes d'un point de vue pratique.

Dans ce qui suit, nous présenterons les définitions essentielles et quelques notions de bases portant sur les hypergraphes et les traverses minimales, qui nous seront nécessaires à l'introduction de la problématique de l'extraction des traverses minimales, en se basant essentiellement sur les définitions proposées par Berge [20].

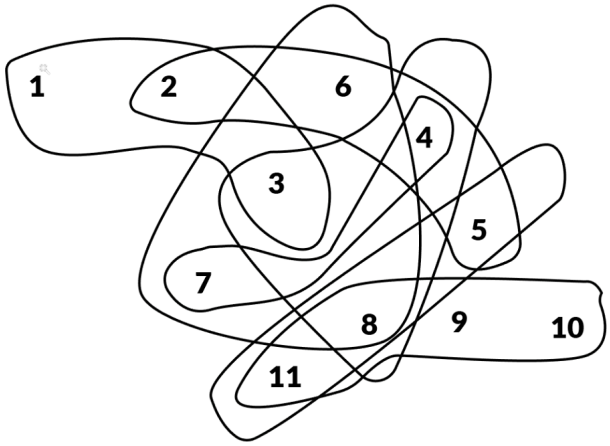


FIGURE 4.2 – Un hypergraphe

4.3.2.1.1 L'hypergraphe Un hypergraphe $H = (S, E)$ est constitué de deux ensembles S et E, et est défini comme suit.

Définition 7 (Hypergraphe) *Soit le couple $H = (S, E)$ avec $S = \{s_1, s2, \ldots, s_n\}$ un ensemble fini et $E = \{e_1, e_2, \ldots, e_m\}$ une famille de parties de E [20]. H constitue un hypergraphe sur S si :*

1. $e_i \neq \emptyset, i \in \{1, ..., m\}$;
2. $\bigcup_{i=1,...,m} e_i = S$

Les éléments s_i de S sont appelés *sommets* de l'hypergraphe et les éléments e_i de E sont appelés *hyperarêtes* de l'hypergraphe. Un hypergraphe est dit d'ordre n si $|S| = n$ où n est le nombre de sommet, et la taille d'un hypergraphe est égale au nombre d'occurrences des sommets dans ses hyperarêtes. L'exemple de la figure 4.2 illustre un hypergraphe $H = (S, E)$ d'ordre 8 et de taille 14 tel que $S = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$ et $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ avec $e_1 = \{1, 2, 3\}$, $e_2 = \{2, 4 , 5, 6\}$, $e_3 = \{4, 7\}$, e_4 $\{3, 4, 8\}$, $e_5 = \{8, 9, 10, 11\}$, $e_6 = \{5, 8, 11\}$ et $e_7 = \{3, 6, 7, 8\}$.

4.3.2.1.2 La traverse minimale Une traverse d'un hypergraphe H est un ensemble de sommets $s \subseteq S$ qui intersecte chaque hyperarête de H au moins une fois et est défini comme suit.

Définition 8 (Traverse minimale) *Soit un hypergraphe $H = (S, E)$. L'ensemble des traverses de H, noté $\gamma(\mathcal{H})$, est égal à : $\gamma(\mathcal{H}) = \{T \subset S \mid T \bigcap s_i \neq \emptyset, \forall i = 1, \ldots, |E|$ [20] . Une traverse T de $\gamma(\mathcal{H})$ est dite minimale s'il n'existe pas une autre traverse S de γ H incluse dans T : $S \in \gamma(\mathcal{H})$ s.t. $S \subset T$.*

Nous noterons $(\mathcal{MH})$, l'ensemble des traverses minimales définies sur H. Dans l'exemple illustratif de la figure 4.2, l'ensemble $(\mathcal{MH})$ des traverses minimales de l'hypergraphe est donné dans le tableau 4.2. À partir d'un hypergraphe $H = (S, E)$, l'ensemble des traverses minimales $(\mathcal{MH})$ permet la construction de l'hypergraphe transversal, qui est un hypergraphe où toutes ses hyperarêtes sont des traverses minimales.

Définition 9 (Nombre de transversalité) *Soit un hypergraphe $H = (S, E)$, le nombre minimum de sommets d'une traverse minimale est appelé le nombre de transversalité de l'hypergraphe H et est désigné par : $\tau(\mathcal{H}) = min\{|T|$, tel que $T \in (\mathcal{MH})\}$ [20].*

Ainsi, dans l'exemple illustratif de la Figure 4.2, le nombre de transversalité de l'hypergraphe H est égal à 3 car la plus petite traverse minimale de $\mathcal{MH}$ est composée de 3 sommets. La détermination d'un nombre de transversalité apparaît dans de nombreux problèmes combinatoires [20].

Tableau 4.2 – Les traverses minimales de H

{3, 4, 8}	{3 ,4, 11}	{2 ,7, 8}
{2, 4, 8}	{1, 4, 8}	{3, 6, 7, 8}
{3, 6, 7, 11}	{3, 5, 7, 8}	{3, 5, 7, 9}
{3, 5, 7, 10}	{3, 5, 7, 11}	{3, 4, 5, 10}
{3, 4, 5, 9}	{2, 3, 7, 11}	{2, 4, 6, 11}
{2, 4, 7, 11}	{1, 6, 7, 8}	{1, 5, 7, 8}
{1, 4, 6, 11}	{1, 4, 7, 11}	{2, 4, 5, 6, 10}
{2, 4, 5, 7, 10}	{2, 4, 5, 6, 9}	{2, 4, 5, 7, 9}
{1, 4, 5, 6, 10}	{1, 4, 5, 7, 10}	{1, 4, 5, 6, 9}
	{1, 4, 5, 7, 9}	

L'intérêt pour l'extraction des traverses minimales ne cesse pas de s'accroître, et ce, en raison de la multitude et de la diversité des domaines d'application où les traverses minimales peuvent être proposées comme une solution. Le large éventail des domaines d'application donne une importance plus grande aux traverses minimales et motive l'intérêt qu'elles suscitent. Plusieurs domaines ont adopté cette solution pour résoudre certains problèmes classiques. Modéliser le problème de la détermination de la dualité FNC/FND par un problème de calcul des traverses minimales d'un hypergraphe présente un exemple du recours aux TMs comme solution. Le domaine des bases de données a eu sa part de l'emploi des traverses minimales pour trouver des solutions à ses différents problèmes, comme l'inférence des dépendances fonctionnelles [106] et la déduction des dépendances d'inclusion [43]. Ces dernières, sont une généralisation de la nation des clés étrangères dans un modèle relationnel. Les traverses minimales, présentent par ailleurs, des solutions aux problèmes de réécriture des requêtes, d'exécution des requêtes et d'actualisation des vues. Elles sont utilisées aussi pour la génération des règles associatives, des motifs fermés ou encore des motifs fréquents dans le domaine de fouille de données [57]. Le large éventail des domaines d'application des TM, est résumé dans la figure 4.3,

Le problème d'extraction des traverses minimales d'un hypergraphe est l'un des problèmes les plus cruciaux en théorie des hypergraphes. C'est un problème algorithmique central et particulièrement difficile et la question de sa complexité exacte reste toujours ouverte. Pour le traiter, plusieurs travaux s'y intéressaient en proposant diverses méthodes [20]. Trouver une traverse minimale d'un hypergraphe est une tâche loin d'être triviale. Calculer l'ensemble de toutes les traverses minimales devient de plus en plus compliqué proportionnellement au nombre de sous-ensembles de sommets à tester. Pour faire sauter les différents verrous scientifiques que posait l'extraction des traverses minimales d'un hypergraphe, plusieurs travaux de recherche ont été proposés. Ces travaux se sont attachés surtout à réduire l'espace de recherche. Néanmoins, le coût du calcul reste substantiellement considérable et les algorithmes existants se sont affrontés à des temps d'exécution élevés et parfois à l'incapacité de traitement lorsque le nombre de transversalité de l'hypergraphe est important.

Le nombre de traverses minimales d'un hypergraphe augmente d'une manière exponentielle en la taille de l'hypergraphe. Ainsi, la question de la mise en place d'un algorithme permettant de résoudre le problème de l'extraction des traverses minimales d'un hypergraphe avec une complexité polynomiale reste néanmoins ouverte. Dans la littérature, plusieurs chercheurs se sont intéressés au problème de l'extraction des traverses minimales en proposant une panoplie d'approches. Ainsi, nous présentons un survol sur ces différentes approches, en mettant en exergue leurs points forts et leurs limites. Le premier à s'être intéressé à ce problème en proposant un algorithme pour le résoudre était *Berge* [20]. Son algorithme, dont le principe est simple, commence par calculer l'ensemble des traverses minimales de la première hyperarête, qui est l'ensemble des sommets qui compose cette dernière. Ensuite, il met à jour cet ensemble des

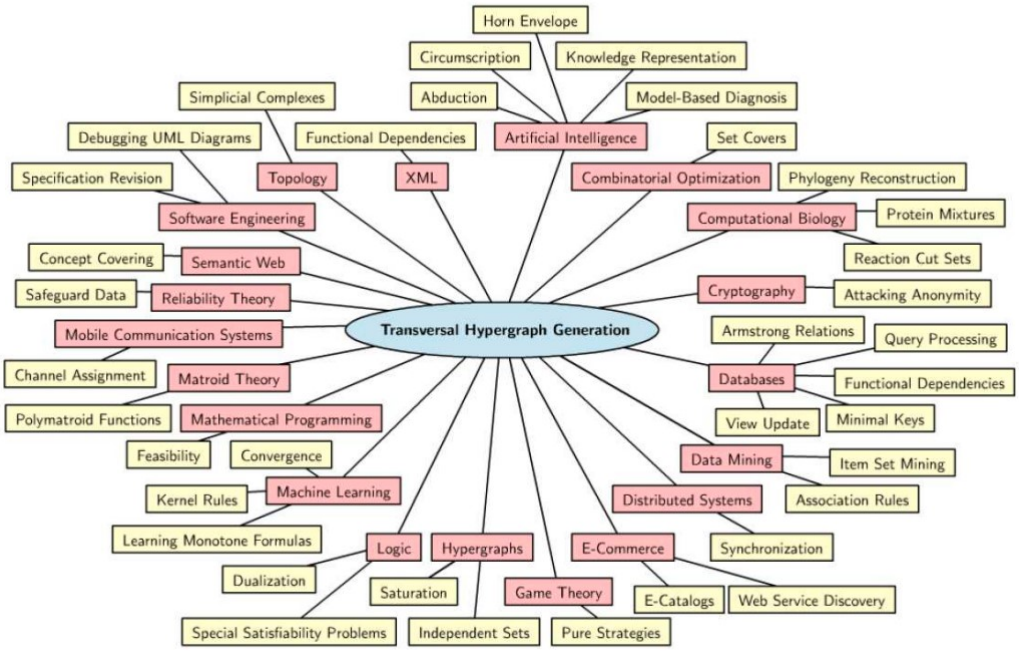


FIGURE 4.3 – Les domaines d'application des traverses minimales [68]

traverses minimales en ajoutant, une à une, les autres hyperarêtes de manière incrémentale. De cette manière, des hypergraphes partiels sont construits au fur et à mesure en ajoutant des hyperarêtes. Cependant, l'algorithme a toujours besoin de stocker les TMs intermédiaires avant de passer à l'étape suivante, dans laquelle il ajoute une nouvelle hyperarête. L'un des inconvénients majeurs de cet algorithme est la consommation démesurée en mémoire, ce qui a incité plusieurs chercheurs à l'améliorer. Parmi les améliorations proposées figurent celles introduites par *Dong et al.* [48], *Bailey et al.* [10] et *Kavvadias et Stavropoulos* [91].

Murakami et Uno proposent les algorithmes SHD, MMCS et RS, qui visent à réduire l'espace de recherche [119]. Ces algorithmes sont conçus pour le traitement des hypergraphes gigantesque ayant un très grand nombre d'hyperarêtes. Ces algorithmes adoptent une stratégie de parcours en profondeur de l'espace de recherche qui est équivalente à celle de l'algorithme proposé dans [91]. La principale différence repose sur l'élimination des itérations redondantes où aucun sommet n'est ajouté à un ensemble de sommets générés auparavant. De plus, *Murakami et Uno* introduisent deux nouveaux concepts, i.e., le test de la transversalité (UNCOV) et les hyperarêtes critiques (CRIT), dans le but d'optimiser les tests sur la minimalité effectués sur l'ensemble des traverses générées.

L'algorithme SHD dont le pseudo-code est donnée dans Algorithme 1, se base sur la notion de la récursivité, il fournit en sortie des traverses minimales en série. Pour tester un ensemble de sommets X, les algorithmes cherchent, de façon itérative, les sous-ensembles de X et effectuent un appel récursif pour chacun tout en mettant à jour les ensembles CRIT et UNCOV. En procédant de cette manière, les auteurs permettent à leur algorithme de balayer l'espace de recherche en profondeur en ne cherchant que les sous-ensembles du candidat courant. La méthode et les étapes pour la recherche des sous-ensembles d'un candidat sont détaillées dans [119].

Algorithme 1 : L'algorithme SHD

Entrées : $H = (\mathcal{X}, E)$: Hypergraphe, X : ensemble de sommets
Sorties : T tel que $T \in \mathcal{M}_H$

1 ***Var. Globale*** : *uncov* (initialisé à E), $Cand$ (initialisé à $\mathcal{X}$), $crit[x]$ initialisé à $\emptyset$ pour chaque x
2 **si** uncov = $\emptyset$ **alors**
3 **retourner** X
4 Choisir une hyperarête e à partir de *uncov* ;
5 $C = Cand \cap e$;
6 $Cand = Cand \setminus C$;
7 **pour chaque** $x \in C$ **faire**
8 UPDATE_CRIT_UNCOV(x, *crit*, *uncov*) ;
9 **si** crit(f, $X \cup x$) $\neq \emptyset$ ***pour chaque*** $f \in X$ **alors**
10 SHD($X \cup x$) ;
11 $Cand = Cand \cup x$;
12 Restaurer les valeurs de *crit* et *uncov* d'avant la ligne 8 ;

4.3.2.2 L'opérateur de Skyline

Dans le cas de choix sans incertitude, le choix devient très facile en connaissant les conséquences de chaque option. Par contre, le choix devient plus difficile lorsque les options disponibles ont des forces et des faiblesses qui se compensent les unes par rapport aux autres. C'est pour cette raison que les recherches sur la préférence sont étendues. Généralement, pour un ensemble de points le problème d'optimisation multi-objectif est composé de trois paramètres. Un ensemble d'attributs (variables de décision) qui caractérisent chaque point, un ensemble de fonctions objectives définies sur les attributs et un ensemble de contraintes. L'objectif d'optimisation est de déterminer l'ensemble de points optimal permettant de minimiser ou de maximiser les fonctions objectives (selon la préférence) tout en respectant les contraintes.

Pour résoudre un problème multi-objectif (PMO), deux étapes peuvent être identifiées : (*i*) la recherche et (*ii*) la prise de décision. La recherche, concerne le processus d'optimisation dans lequel l'ensemble des solutions réalisables est identifié pour trouver les éventuelles solutions. La deuxième étape est la prise de décision qui traite le problème de sélection d'une solution de compromis appropriée à partir de l'ensemble de solutions précédemment identifié.

Pour traiter un PMO plusieurs approches ont été proposées, ces approches peuvent être divisées en deux classes. La classe des méthodes classiques qui regroupent les objectifs en une seule fonction objective paramétrée. Il existe plusieurs approches appartenant à cette classe, parmi lesquels nous pouvons citer : la méthode de pondération, la méthode ϵ-contrainte, et la programmation par but. La deuxième classe comporte les méthodes basées sur les algorithmes évolutionnaires comme les algorithmes génétiques et les approches basées sur la dominance. Ces derniers adoptent des méthodes de recherche inspirées de la nature pour pallier les problèmes complexes ayant un énorme espace de recherche ce qui les rend comme une alternative aux méthodes classiques décrites précédemment. Ces algorithmes sont souvent populaires chez les chercheurs qui les préfèrent pour la résolution des PMO dans divers domaines d'applications grâce à leurs caractéristiques telles que la possibilité de trouver des solutions optimales multiples dans une seule simulation.

Parmi les approches basées sur la dominance, nous trouvons l'opérateur de Skyline. Ce dernier, utilise la dominance de Pareto (Cf. définition 10). Dans un ensemble de points noté P, le Skyline se compose de l'ensemble de points qui ne sont dominés par aucun autre point [21]. Le

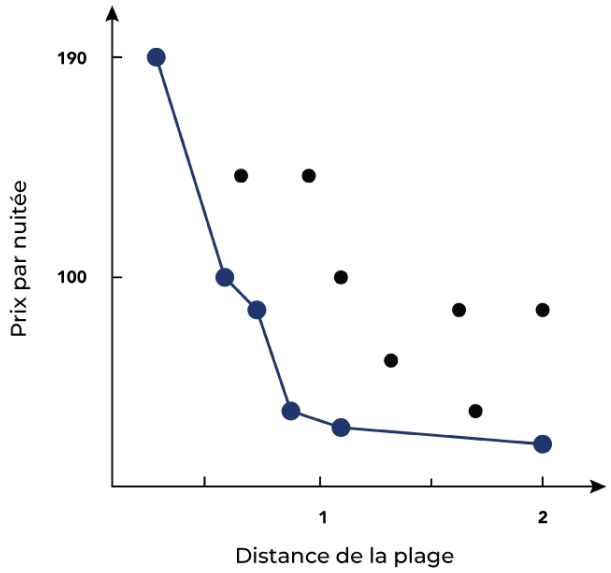


FIGURE 4.4 – Le Skyline des hôtels

Skyline vise à aider à prendre des décisions intelligentes sur des données multidimensionnelles pour des problèmes multi-objectif. Il consiste donc à extraire les points les plus intéressants selon des critères définis par l'utilisateur (préférence de l'utilisateur). L'analyse du Skyline implique plusieurs attributs. La préférence d'un utilisateur sur les valeurs d'un attribut peut être modélisée par un ordre partiel sur l'attribut, qui est définie une relation irréflexive, asymétrique et transitive.

La relation de dominance a une structure simple : soient deux vecteurs V_1 et V_2. On dit que V_1 domine V_2 si V_1 est meilleur ou égal à V_2 dans toutes les dimensions et strictement meilleur que V_2 dans au moins une dimension [21]. La dominance entre les vecteurs dépend de la préférence de l'utilisateur (c'est-à-dire minimum, maximum) [2]. Formellement , la dominance de pareto est définie comme suit :

Définition 10 *Dominance de Pareto : Soit A et B deux points définis dans un ensemble de points noté P avec n attributs. Un point B domine un point A noté* $B \succ A$, *si* $\forall i \in [1, n]\ b_i \leqslant a_i \wedge \exists j, b_j < a_j$.

L'exemple le plus connu d'une requête de Skyline est le problème de réservation d'hôtel où les utilisateurs recherchent les hôtels les moins chers et les plus proches de la plage.

Exemple 4 *Étant donné une liste d'hôtels (tableau 4.3) ayant deux critères (attributs), la distance de la plage et le prix par nuit. Ainsi, nous visons à trouver les hôtels ayant une bonne combinaison des deux critères, un prix bas, et à proximité de la plage.*

Nous illustrons sur la figure 4.4 l'ensemble des hôtels, où chaque point est caractérisé par deux valeurs : le prix et la distance de la plage. Les hôtels dominés par aucun autre selon les critères mentionnés précédemment (prix bas et courte distance) sont représentés en bleu. En effet, ces points représentent l'ensemble de Skyline. L'hôtel $h_{12}(27, 2.00)$ *a le prix le plus bas parmi tous les hôtels. Par conséquent, il n'est dominé par aucun autre hôtel. L'hôtel* $h_6(70, 1.90)$ *est dominé par l'hôtel* $h_3(70, 1.40)$, *puisque* h_3 *est le plus proche de la plage, donc* h_6 *est supprimé du Skyline.*

En raison de leur large utilisation dans les problèmes de prise de décision, l'opérateur de Skyline a été utilisé dans plusieurs domaines d'application, comme les analyses de flux de

2. Nous supposons dans la définition suivante, que la plus petite valeur est la plus préférable.

Tableau 4.3 – Les caractéristiques des hôtels

Hôtel	Prix	Distance (km)
h_1	160	1.00
h_2	43	1.75
h_3	70	1.40
h_4	66	1.70
h_5	55	1.50
h_6	70	1.90
h_7	81	0.80
h_8	90	1.00
h_9	125	0.60
h_{10}	115	1.40
h_{11}	140	0.50
h_{12}	27	2.00
h_{13}	190	0.30

données en temps réel et dans le contexte du traitement parallèle des requêtes sur des systèmes multi-cœurs.

Pour déterminer l'ensemble de Skyline, plusieurs approches ont été proposées dans la littérature. Ces approches peuvent être réparties sur trois grandes familles, les approches basées sur l'algorithme des boucles imbriquées, les approches basées sur la notion diviser pour régner et les approches basées sur le B-Tree.

L'algorithme de boucles imbriquées [21] est un algorithme itératif qui analyse à plusieurs reprises un ensemble d'enregistrements. L'algorithme de boucles imbriquées est un solution naïve pour déterminer l'ensemble de Skyline, où chaque tuple est comparé à tous les autres tuples. L'algorithme diviser pour régner a été proposé par *Preparata et Shamos* [139]. L'algorithme B-tree permet l'utilisation d'un index ordonné pour un Skyline en deux dimensions en balayant tout l'index et en obtenant les tuples dans l'ordre trié et en filtrant les tuples du Skyline. Le tableau 4.4 résume les différences existantes entre les trois algorithmes. Une étude détaillée de l'état de l'art sur les différentes approches et techniques pour implémenter l'opérateur Skyline est présentée par *Kalyvas et Tzouramanis* [83]. L'opérateur Skyline peut être implémenté directement dans le langage SQL en utilisant des instructions SQL courantes, mais cela s'est avéré être très lent.

Tableau 4.4 – Les approches de sélection du Skyline

Algorithme	Pré-traitement	Complexité	Inconvénient
Boucles imbriquées	Le tri	$O(n^2)$	Hors ligne
Diviser pour régner	Des Skylines partiels	$O(n^2)$	Hors ligne
B-tree	Index ordonné	$O(log\ n)$	Manque de l'interaction de l'utilisateur

4.3.3 Les étapes de notre approche

Dans cette section, nous décrivons en détail notre approche proposée, dans laquelle l'idée principale réside dans l'utilisation des traverses minimales d'un hypergraphe. Ce dernier, représente la charge de travail, à travers une matrice requêtes-attributs, où chaque hyperarête représente une requête, et chaque sommet représente un attribut qui a fait l'objet d'un prédicat de sélection ou de projection. Dans cette approche, la réduction de la consommation énergétique

d'un entrepôt de données est l'objectif recherché en choisissant la configuration d'IJB la plus écologique tout en respectant l'objectif initial des différentes structures d'optimisation qui est l'optimisation du temps d'exécution.

Après l'analyse de la charge de travail et l'extraction de tous les attributs indexables, nous construisons un hypergraphe à travers uniquement les attributs indexables ayant des faibles valeurs de $FAN-OUT$. L'utilisation de l'hypergraphe pour la modélisation de la charge de travail est motivé par son aptitude à représenter un ensemble considérable de requêtes et par sa capacité de modéliser les relations existantes entre elles. Dans notre approche, nous commençons par le calcul du nombre de transvers alité, qui est le cardinal de la plus petite traverse minimale. Ensuite, nous extrayons toutes les traverses minimales (TMs) qui ont une taille égale au nombre de transversalité. Ensuite, pour chaque TM nous calculons la métrique Penalty, et nous ne gardons que les TMs minimisant cette dernière. Si à ce niveau, nous avons plus qu'une seule traverse minimale retenue, alors dans ce cas, nous allons calculer le $FAN-OUT$ moyen des attributs de chaque traverse minimale grâce à la métrique AFO et on ne retiendra que la TM qui minimise cette dernière, puisque la principale cause de l'explosion du temps de traitement lors de l'utilisation d'un IJB est le $FAN-OUT$ élevé. Ensuite, pour chaque sommet de la traverse minimale qui représente un attribut indexable, nous évaluons son coût d'E/S et sa consommation d'énergie, et nous utilisons l'opérateur Skyline pour la sélection multi-objectifs. À partir du résultat de l'opérateur de Skyline, nous allons construire la configuration finale d'indexes (CFI). Il convient de mentionner que si la CFI viole la contrainte de stockage, alors les indexes les plus énergivores seront éliminés un par un jusqu'à ce que la CFI satisfait la contrainte de stockage. Le *modus operandi* de notre approche est présenté par la figure.4.5.

Dans ce qui suit, nous décrivons en détail les étapes de notre approche énergétique de sélection d'IJB.

Étape 1 : Analyse de la charge de travail : les requêtes SQL de la charge de travail sont traitées par un parseur automatique pour extraire tous les attributs indexables. Ces attributs sont ceux présents dans la clause Where. Ainsi, les attributs clés sont éliminés.

Étape 2 : Élimination des attributs ayant un FAN-OUT élevé : Après l'extraction des attributs indexables, nous supprimons ceux ayant une valeur de FAN-OUT supérieure ou égale à un seuil fixé expérimentalement.

Etape 3 : Construction de l'hypergraphe : à partir des attributs extraits de l'étape précédente, nous construisons une matrice (requêtes-attributs) appelée *hypergraphe* qui a pour hyperarêtes les requêtes de la charge de travail et pour sommets les attributs indexables.

Etape 4 : Extraction des traverses minimales : une fois l'hypergraphe est construit, nous passons à l'extraction des traverses minimales. Pour cette étape, nous profitons des travaux récents de la littérature consacrés à l'extraction des TMs. Pour ce faire, nous utilisons le meilleur algorithme existant en termes de performance, c'est-à-dire SHD de *Murakami et Uno* [119]. Nous avons configuré l'algorithme SHD pour qu'il ne génère que les plus petites traverses minimales. Pour ce faire, nous avons récupéré le nombre de transversalité de notre hypergraphe via la fonction GetMinTransversality [80] dont le pseudo-code est donné dans l'algorithme 2. Cette dernière prend en entrée un hypergraphe $H = (S, E)$ et renvoie son nombre de transversalité. Cette fonction parcourt les sommets un par un, ensuite, pour chaque sommet s de H, elle supprime les hyperarêtes de E contenant s. Les hyperarêtes restantes sont stockées dans E'. Ensuite, cette fonction fait appel à la fonction hyp_empty() qui est une fonction récursive qui stocke les sommets ayant le plus grand support dans E' et supprime à chaque fois les hyperarêtes auxquelles appartient le sommet traité. Cette fonction récursive s'arrête lorsque E' soit vide. La valeur retournée correspond au nombre d'appels à la fonction hyp_empty nécessaires pour que E' soit vide.

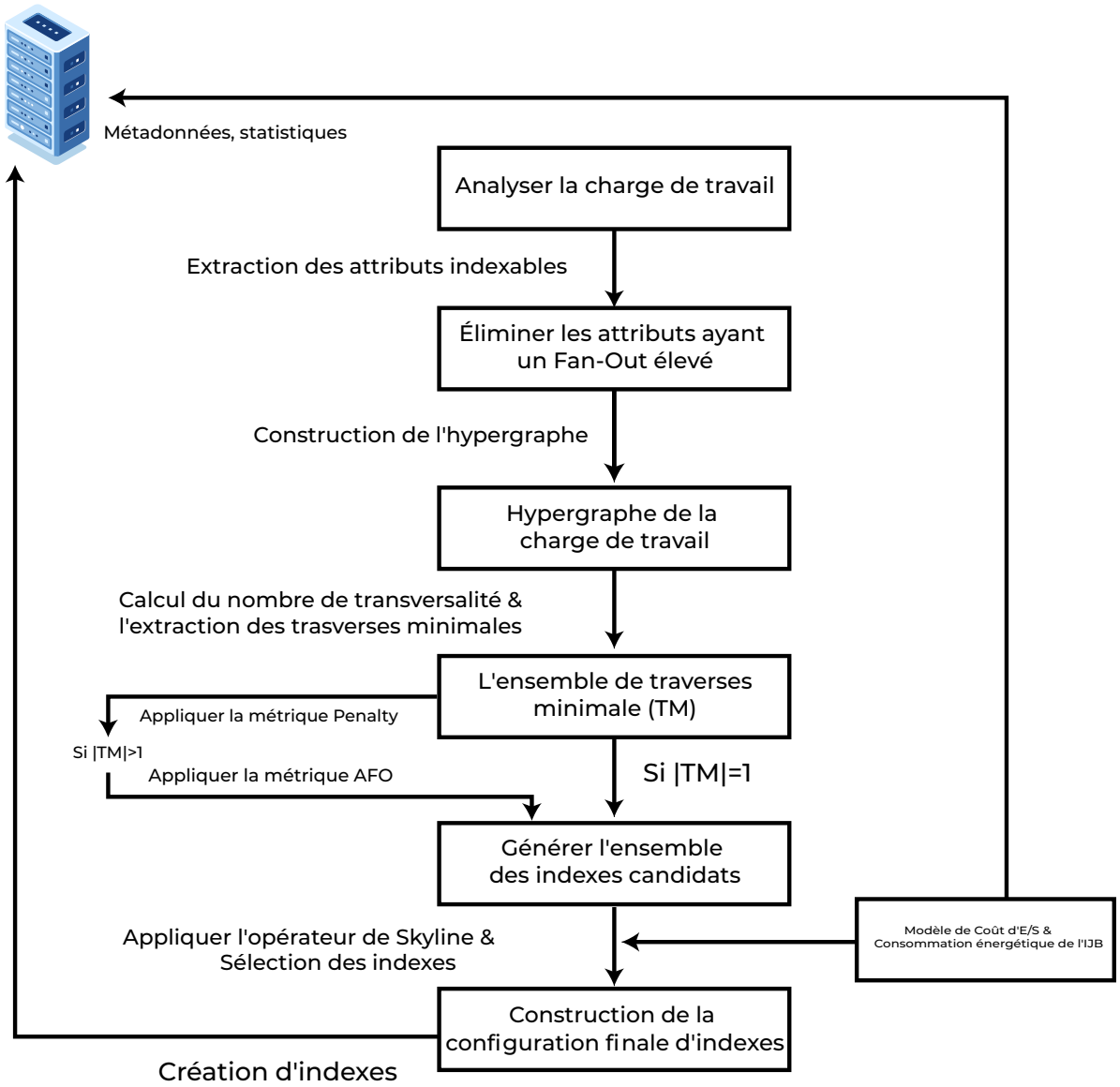


FIGURE 4.5 – GBJI : Une approche multi-objectifs pour la sélection des *IJB*

Algorithme 2 : GetMinTransversality

Entrées : Matrice d'incidence IM_H associée à $H = (S, E)$;
Sorties : T : Une plus petite traverse minimale de H,
k : Nombre de transversalité de H ;

```
1  début
2      k = | E | ;
3      T = ∅ ;
4      pour chaque s ∈ S faire
5          i = 1 ;
6          T_tmp = ∅ ;
7          T_tmp[i] = s ;
8          E' = E \ {e ∈ E | s ∈ S} ;
9          (n, T_tmp) = HYP_EMPTY(E', | E |, i, T_tmp) ;
10         si n < k alors
11             k = n ;
12             T = T_tmp ;
13     retourner (k, T)
```

Etape 5 : Génération de l'ensemble des indexes candidats : après l'extraction de l'ensemble des traverses minimales, nous passons à la génération des indexes candidats. Pour ce faire, nous procédons comme suit :

1. **Application de la métrique Penalty**
   Plutôt que de se fier uniquement à la fréquence d'occurrence comme critère unique pour déterminer les attributs pertinents, nous introduisons une métrique de pénalité. Cette dernière pénalise chaque traverse minimale, qui contient des attributs appartenant à des tables de dimension de petite taille, compte tenu de la taille de ces tables, et de la taille de la table de faits. Notre métrique Penalty est calculée comme suit :

$$\text{Penalty}(tm) = \frac{1}{\sum_{i=1}^{n} sup_i \times \frac{|D_i|}{|F|}}$$

   où n est le nombre d'attributs indexables de la traverse minimale tm, et i est un attribut de tm. sup_i représente le support de i, c'est-à-dire le nombre d'occurrences de l'attribut i dans la matrice de la charge de travail. $|D_i|$ et $|F|$ désignent respectivement la taille de la table de dimension D à laquelle appartient l'attribut i, et la table de faits F en nombre de pages.

Si le calcul de la métrique Penalty donne plusieurs traverses minimales candidates, nous utiliserons une autre métrique pour choisir la meilleure configuration possible.

2. **Application de la Métrique AFO**
   La principale cause de l'augmentation du temps de traitement lors de l'utilisation de IJB sur un attribut est le $FAN-OUT$ de la valeur de ce dernier comme nous l'avons montrer expérimentalement lors de l'audit que nous avons effectué. Dans cette métrique, nous calculons pour chaque TM retenue, la moyenne du $FAN-OUT$ de ses sommets, et nous ne gardons que la TM ayant la valeur la plus faible.

Etape 6 : Sélection des indexes : une fois les différentes métriques appliquées, nous utilisons l'opérateur de Skyline, pour la sélection multi-objectifs. Pour chaque attribut indexable de la traverse minimale sélectionnée, nous évaluons son coût d'E/S et sa consommation

d'énergie. Le modèle de coût d'Aouiche et al. [7] est utilisé pour estimer le coût de traitement des requêtes en termes d'opérations d'E/S, et pour la consommation d'énergie, nous utilisons un wattmètre. Pour l'implémentation de l'opérateur de Skyline, nous avons utilisé le package RPREF [3]. Il s'agit d'un package pour le langage de calcul statistique *R*, permettant le calcul de l'opérateur de Skyline et quelques légères généralisations ("préférences de base de données").

Étape 7 : Construction de la configuration finale : une fois toutes les étapes précédentes soient terminées, les attributs renvoyés par l'opérateur Skyline vont construire la configuration finale des indexes. Si cette dernière viole la contrainte de stockage exigée, dans ce cas, nous éliminons les IJB un par un en se basant sur leur consommation d'énergie, nous commençons par le plus énergivore, jusqu'à que la configuration finale satisfait la contrainte de stockage.

4.3.4 Exemple illustratif

Considérons un schéma en étoile contenant trois tables de dimension, à savoir $Product$ (noté P), $Customer$ (noté C) et $Store$ (noté St) et une table de faits $Sales$ (noté par S) comme le montre la figure 4.6. Les tailles respectives de ces tables en termes de nombre de pages sont de 420, 36, 19 et 9852. Nous avons aussi défini une charge de travail W composée des requêtes les plus fréquemment exécutées. Chaque requête contient un ou plusieurs attributs indexables.

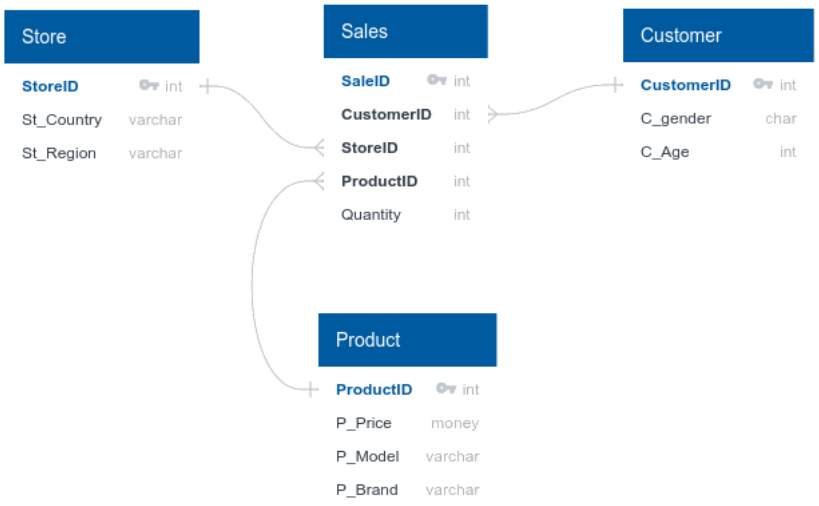


FIGURE 4.6 – Le schéma logique de notre exemple de sélection d'IJB

Pour faciliter la construction de l'hypergraphe, nous donnons aux attributs indexables un identifiant tel que tracé dans le tableau 4.5. Il convient de mentionner que l'hypergraphe (tableau 4.6) est construit uniquement des attributs indexables, pour éviter la génération des traverses minimales contenant des attributs clé. Ensuite, nous supprimons de l'hypergraphe généré les attributs qui ont une valeur élevée de FAN-OUT (tableau.4.7), par exemple, nous avons élagué tous les attributs ayant un FAN-OUT supérieur à 1%.

L'exécution de l'algorithme d'extraction des traverses minimales donne deux TMs, TM_1={a_3, a_6} et TM_2={a_3, a_5}. Ainsi, pour chacune des TMs candidates nous calculons la valeur de PENALTY correspondante. Ci-dessous, nous montrons comment cette dernière est calculée pour TM_1 {3,6}.

3. https://www.p-roocks.de/rpref/index.php?section=download

Tableau 4.5 – Les méta-données des attributs indexables de la charge de travail W

Id	Attribut Indexable	FAN-OUT	Table
A_1	C_Gender	50.00%	Customer
A_2	c_Age	2.50%	Customer
A_3	St_Country	0.49%	Store
A_4	St_region	20.00%	Store
A_5	p_Price	0.66%	Product
A_6	P_Model	0.10%	Product
A_7	P_Brand	2.85%	Product

Tableau 4.6 – L'hypergraphe de la charge de travail

$a_4\ a_3$
a_3
$a_1\ a_3$
a_4
$a_3\ a_6\ a_5$
$a_7\ a_4\ a_6\ a_5$
a_2
$a_1\ a_5\ a_3$
$a_2\ a_5\ a_6$
$a_3\ a_7\ a_6$

Tableau 4.7 – L'hypergraphe après l'élagage de FAN-OUT

a_3
a_3
a_3
$a_3\ a_6\ a_5$
$a_6\ a_5$
$a_5\ a_3$
$a_5\ a_6$
$a_3\ a_6$

$$\begin{aligned}\textsc{Penalty}(TM_1) &= \frac{1}{sup_{a3} \times \frac{|D_{a3}|}{|F|} + sup_{a6} \times \frac{|D_{a6}|}{|F|}} \\ &= \frac{1}{(6 \times \frac{19}{9{,}852}) + (4 \times \frac{420}{9{,}852})} \\ &= \frac{1}{0.181} \\ \textsc{Penalty}(TM_1) &= 5.52\end{aligned}$$

Le calcul de la métrique Penalty de TM_1 et TM_2 donne la même valeur pour les deux traverses minimales. Par conséquent, pour rompre cette égalité, nous utilisons la métrique AFO qui calcule le FAN-OUT moyen des attributs de chaque TM et ne conserve que celle ayant la valeur la plus faible.

$$\begin{aligned}AFO_{TM_1} &= (0.49 + 0.1)/2 = 0.29 \\ AFO_{TM_2} &= (0.66 + 0.1)/2 = 0.38\end{aligned}$$

Dans ce cas, TM_1 est considérée comme une solution potentielle, car son FAN-OUT moyen est inférieur à celui de TM_2. Ainsi, après le calcul des différentes métriques de pénalisation, l'algorithme renvoie sa recommandation, à savoir (a_3, a_6). Nous exécutons le modèle de coût pour évaluer le coût de traitement de la charge de travail en présence de chaque attribut en tant que IJB et nous évaluons l'énergie consommée pour exécuter chaque requête. L'opérateur de Skyline est utilisé par la suite pour choisir quel attribut sera indexé.

4.4 L'algorithme de G-BJI

L'algorithme G-Bji, dont le pseudo-code est donné dans l'algorithme 3, prend en entrée une charge de travail et retourne une configuration d'IJB la plus écologique. Une fois, l'hypergraphe construit à travers les attributs indexables avec une faible valeur de $FAN - OUT$ (ligne 1 & 2) le processus mis en œuvre par notre approche commence par la construction de l'hypergraphe correspondant et le calcul du nombre de transversalité (ligne 3 & 4). Ensuite, il s'enchaîne par l'extraction de toutes les traverses minimales dont la taille est égale au nombre de transversalité en invoquant l'algorithme SHD (ligne 5) dont le pseudo-code est donné dans l'algorithme 1. Une fois toutes les TMs sont extraites, nous calculons pour chacune d'entre elles, la métrique Penalty et nous ne gardons que celle ayant la valeur minimale (ligne 6).

Algorithme 3 : G-Bji

Entrées : Une charge de travail W ;
Sorties : Une configuration finale d'indexes : un ensemble d'attributs indexables ;

1 Att := ExtraireAttributsIndexables(W) ;
2 Att_F := EliminerFanOutElevé(Att) ;
3 H := ConstruireHypergraphe (Att_F) ;
4 t := GetMinTransversality (H) ;
5 MT := Shd(S,E,t) ;
6 MT_{min} := min(penalty(MT)) ;
7 **si** ($|MT_{min}| > 1$) **alors**
8 $\quad R$ = AFO(MT_{min}) ;
9 **sinon**
10 $\quad R = MT_{min}$;
11 **pour chaque** $v \in R$ **faire**
12 $\quad$ EstimerCoûtES(v) ;
13 $\quad$ MesurerConsommationEnrgie (v) ;
14 S = Skyline(R) ;
15 **retourner** S ;

Si une unique traverse minimale minimise la métrique Penalty, alors, elle sera choisie et nous passons à l'étape suivante (ligne 9). Sinon, on calcule pour chaque TM le $FAN - OUT$ moyen de ses attributs, en utilisant la métrique AFO (ligne 7 & 8). Et nous ne gardons que la TM qui minimise la métrique AFO (ligne 7). Ensuite, pour chaque attribut de la traverse minimale retenue, nous estimons son coût d'E/S (ligne 11) et nous mesurons sa consommation d'énergie (ligne 12). Après cela, nous appliquons l'opérateur Skyline pour extraire l'ensemble (S) (ligne 13), qui est composé d'attributs non dominés. Les attributs retournés par l'opérateur de Skyline seront considérés comme la configuration finale de l'index (ligne 14).

4.5 Étude expérimentale

Afin de mesurer l'efficacité et la pertinence de notre proposition, nous avons mené plusieurs expérimentations. Nous avons effectué des tests sur un entrepôt de données implémenté sur le SGBD $Oracle12c$ sur $Ubuntu\ Server$ 18.04 LTS, avec un processeur $Intel\ Core\ i7$-4770, $3,40GHz \times 8$, 16 Gb de mémoire principale, et un disque dur SSD de $256Go$. Nous équipons notre machine par un wattmètre "Yocto-Watt" comme représenté sur la figure 4.7. Le wattmètre permet de mesurer l'énergie consommée de chaque requête.

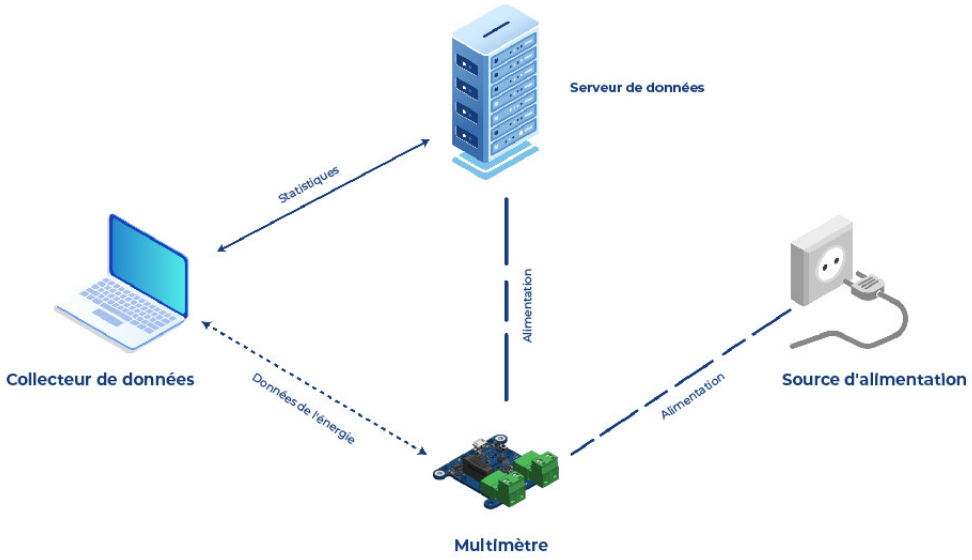


FIGURE 4.7 – L'environnement de travail

Par souci de précision et de fiabilité des mesures, nous avons désactivé les tâches d'arrière-plan inutiles. De plus, le système et la mémoire cache sont vidés après chaque exécution de requête, les requêtes Oracle ci-dessous permettent de le faire.

```
ALTER SYSTEM FLUSH SHARED_POOL ;
ALTER SYSTEM FLUSH BUFFER_CACHE ;
```

Enfin, il convient de mentionner que nous exécutons les requêtes de manière isolée. Notre étude expérimentale a été réalisée sur l'entrepôt de données issu du benchmark SSB, avec un jeu de données de 50 Go. Les requêtes de SSB sont divisées en quatre ensembles, où chaque ensemble contient des requêtes plus complexes que l'ensemble précédent, contrairement aux approches de l'état de l'art qui utilise des benchmarks obsolètes avec des requêtes simples et favorables pour l'utilisation des IJB. Pour prouver l'efficacité de notre approche, nous la comparons à deux approches de la littérature, MT-BJI [62] et DynaClose [17].

Au cours de cette étude expérimentale, nous nous sommes intéressés d'abord à l'évaluation de l'influence de l'IJB sur les économies d'énergie. La figure 4.8 montre un histogramme de l'énergie consommée pour exécuter l'ensmble des requêtes de la charge de travail avant et après la création des IJB fournis par les différentes approches. Pour les requêtes optimisées, l'histogramme montre clairement que notre approche diminue d'une manière drastique l'énergie consommée de la charge de travail par rapport aux requêtes sans indexes, et aux autres approches (DynaClose et MT-BJI) bien que ces dernières soient des approches orientées performance. Cette réduction d'énergie concerne les requêtes ayant des attributs sur lesquels un ou plusieurs IJB sont crées. Dans cette expérimentation, nous avons utilisé uniquement des IJB mono-attribut afin d'augmenter leur utilisation par les différentes requêtes. Le fait de charger uniquement une portion de la table, représentée par l'IJB, et d'éviter de faire une opération de jointure très coûteuse en termes de ressources entre la table de dimension et l'énorme table de faits qui nécessite un temps considérable pour la charger dans la mémoire, nous permet de faire des économies d'énergie et de ressources.

L'utilisation de l'IJB réduit fortement la consommation d'énergie, et notamment pour les requêtes du type *Count (*)*. Pour ce type de requêtes, l'énergie ne dépasse pas 100 (*joule*), et les

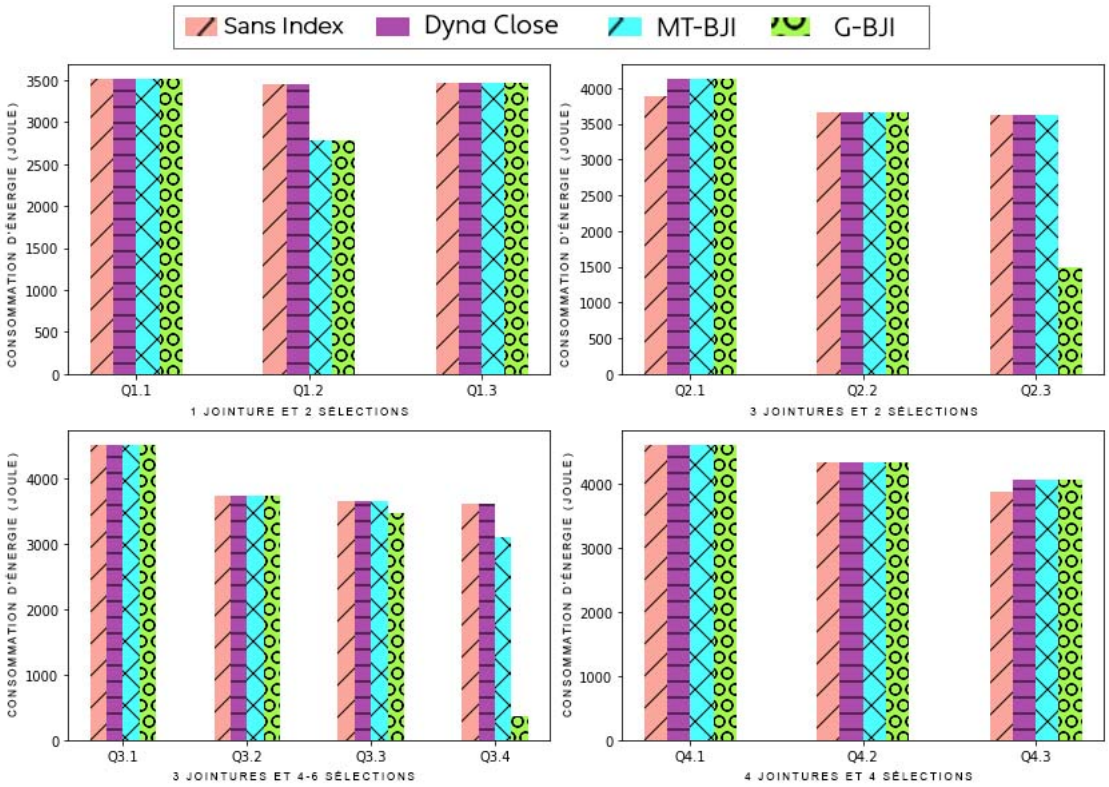


FIGURE 4.8 – La consommation énergétique de la charge de travail

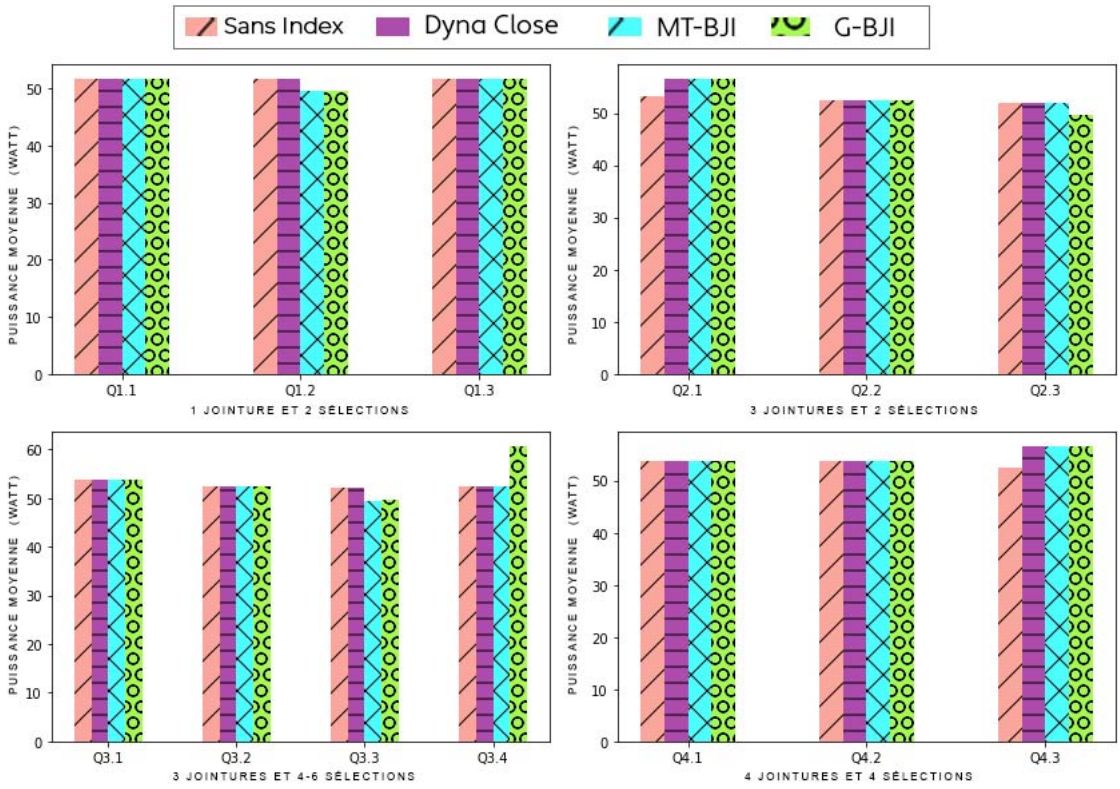


FIGURE 4.9 – La puissance moyenne consommée de la charge de travail

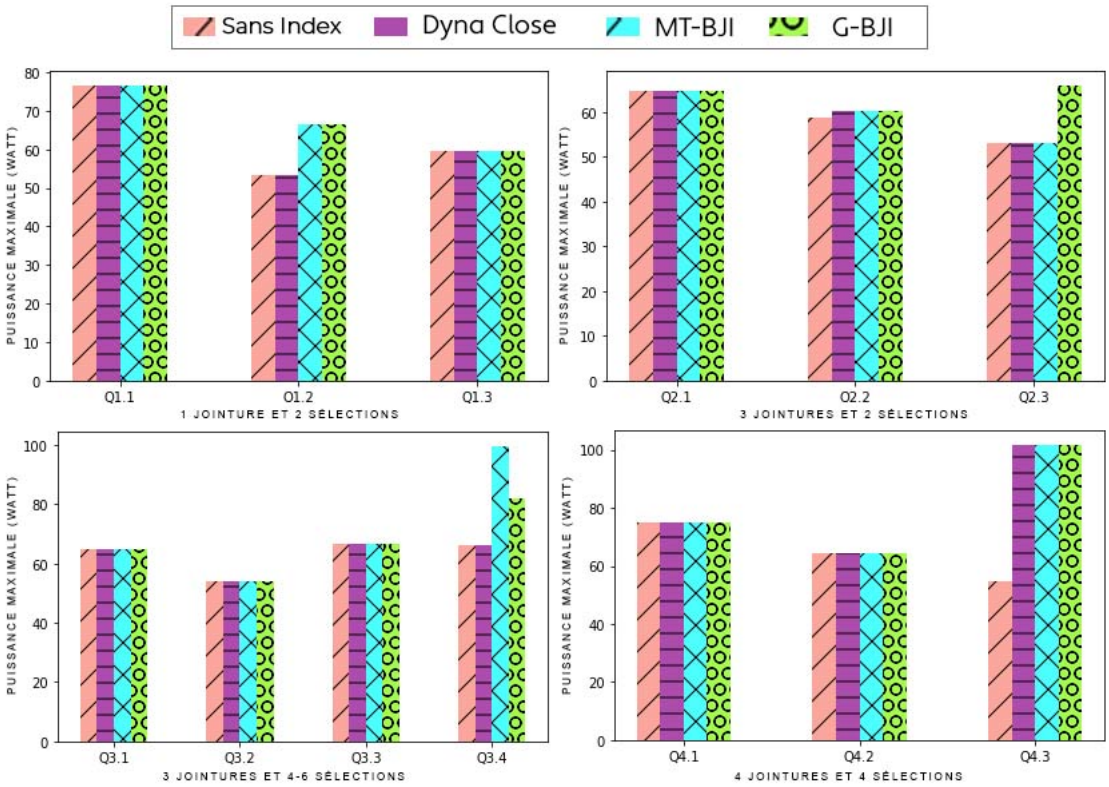


FIGURE 4.10 – La puissance maximale consommée de la charge de travail

résultats sont quasi instantanés puisqu'ils dépassent à peine 1s. D'un autre côté, et contrairement à son type favori de requêtes, l'IJB a montré son efficacité pour les requêtes plus complexes contenant plusieurs jointures et sélections comme celles présentées dans la charge de travail du benchmark SSB.

Contrairement à ce que nous avons vu dans la figure 4.8, nous constatons que la puissance moyenne de toutes les requêtes utilisant les IJB est supérieure à celle des requêtes sans indexes comme le montre la figure 4.9. Pour mieux comprendre ces résultats, nous avons essayé, également, de mesurer la puissance maximale des requêtes. L'histogramme illustré dans la figure 4.10 nous expose la puissance maximale de chaque requête. Le résultat reste inchangé, au contraire, nous remarquons que la puissance maximale des requêtes indexées est largement supérieure à celle des requêtes sans index. Pour dévoiler le secret de cette augmentation de puissance, nous avons décidé d'inspecter le comportement énergétique de ces requêtes en visualisant leur consommation d'énergie en temps réel grâce à l'outil de visualisation de notre wattmètre.

La figure 4.11 montre qu'il y a une crête de puissance au début de l'exécution des requêtes, et lorsque nous regardons le plan d'exécution des requêtes nous constatons que la première étape est le chargement de l'IJB. En effet, lors du chargement de l'IJB dans la mémoire, il y a une augmentation considérable de la puissance, qui est fortement corrélée à la taille de l'IJB. Par conséquent, ces observations sont en accord avec les conclusions de Roukh et al. [152], quand ils ont prouvé que la puissance n'est pas stable pendant le traitement des requêtes.

Bien que l'opérateur de Skyline ait éliminé certains attributs de la traverse minimale sélectionnée, il a été démontré que les attributs restants améliorent les performances de la plupart des requêtes de la charge de travail. En effet, la stratégie d'élagage pilotée par le FAN-OUT nous permet d'engloutir de manière transparente les attributs indexables indésirables et ne laisser que ceux qui nous permettent d'améliorer le traitement des requêtes et la consommation

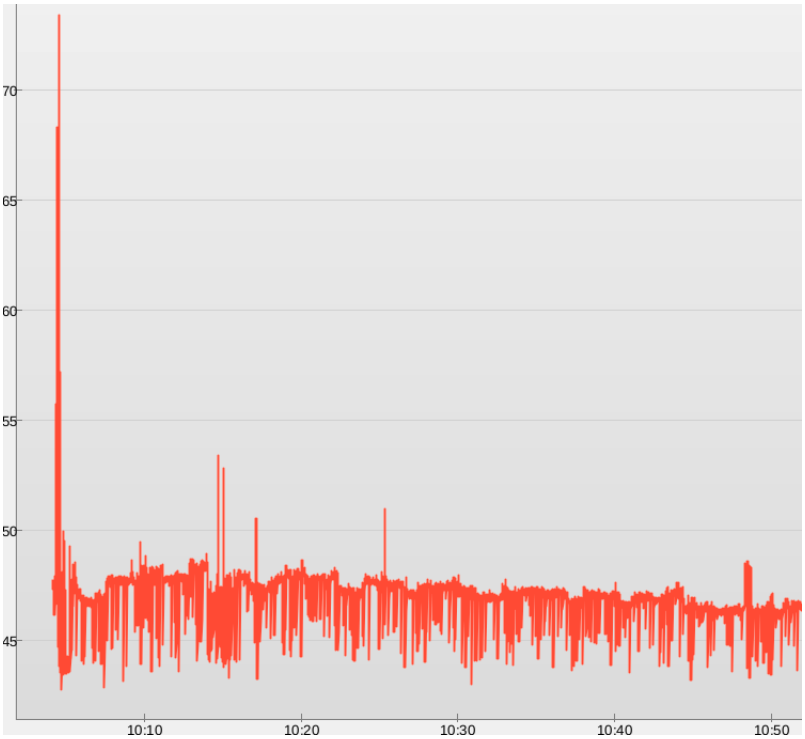


FIGURE 4.11 – L'outil de visualisation de Yocto-Watt

d'énergie. Notre stratégie élimine les indexes qui ont été construits par erreur sur des attributs ayant une valeur de FAN-OUT très élevée. Ce traitement, nous évite le piège de l'indexation des attributs inutiles qui nous conduiraient totalement à une augmentation du temps d'exécution comme nous avons vu dans la section 4.2. Grâce à l'étape d'élagage basée sur le FAN-OUT, les attributs indexés fournis par notre approche améliorent la consommation d'énergie mieux que ceux fournis par l'approche MT-BJI pour la requête $Q3.4$ (cf. Fig.4.8).

Dans la figure 4.12, nous pouvons voir que notre approche réduit non seulement la consommation d'énergie, mais améliore également le temps de traitement de la charge de travail en surpassant ses concurrents, bien qu'ils sont orientés performance. Ainsi, grâce à l'IJB, la consommation d'énergie et le temps de traitement tendent principalement à aller la main dans la main. À l'inverse, dans certaines requêtes, nous avons constaté une augmentation de la consommation d'énergie en même temps qu'une diminution du temps de traitement et vice-versa. Par exemple, la requête $Q4.3$ montre comment, parfois, l'IJB peut réduire le temps de traitement, mais consomme plus d'énergie. Cela concorde avec les expériences de [184] et [98] lorsqu'ils ont démontré que le traitement des requêtes le plus rapidement possible n'est pas toujours le moyen le plus économe en énergie. Par conséquent, nos expérimentations illustrent comment la consommation d'énergie et les performances ne sont pas corrélées.

Pour mieux comprendre ces résultats, nous avons prêté attention au plan d'exécution des différentes requêtes avant et après la création des indexes. Ainsi, nous avons dévoilé que l'ordre de jointure des tables change lors du traitement des requêtes en présence des IJB. Ce changement concerne généralement les tables auxquelles appartiennent les attributs indéxés. Ces tables ont tendances à être les premières à être jointes avec la table des faits, et cela, pour bénéficier de la jointure pré-calculée par l'IJB. Ainsi, ce changement d'ordre de jointure réduit la consommation d'énergie et le temps de traitement. Nous avons également remarqué, dans certaines requêtes, qu'à chaque fois que l'attribut indexé a une valeur élevée de FAN-OUT, forçant son utilisation entraîne une dégradation des performances. Par conséquent, en guise de conclusion, nous

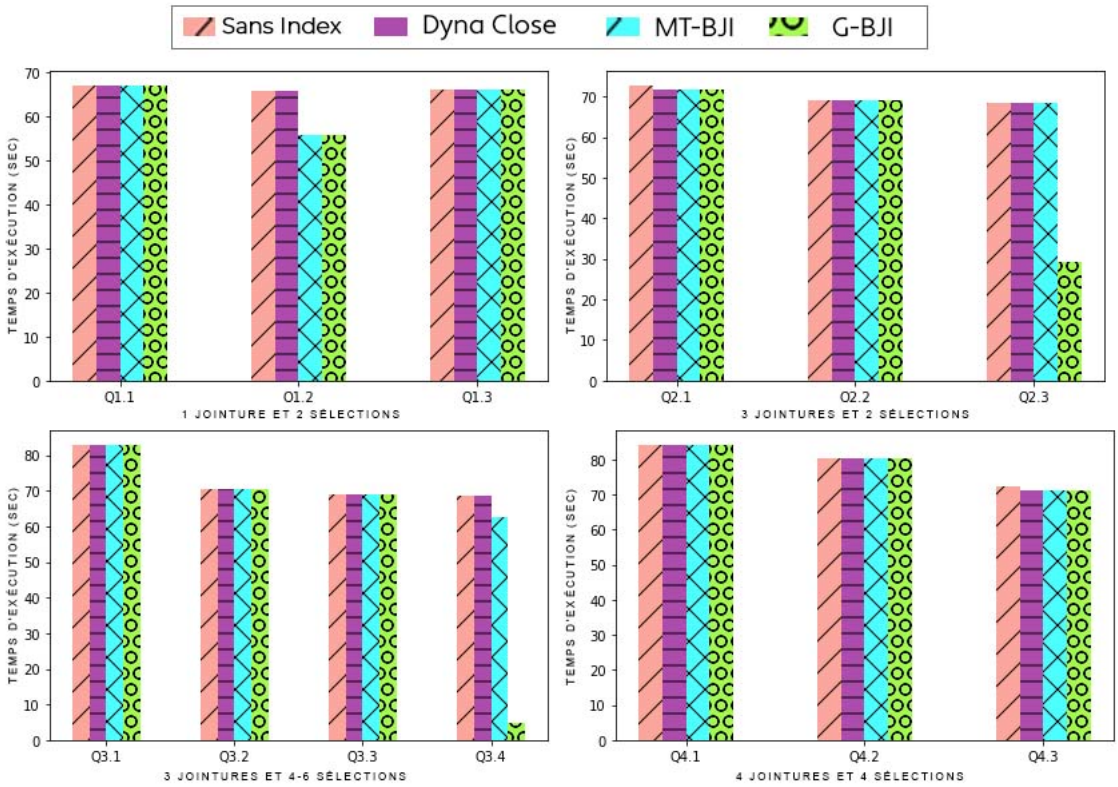


FIGURE 4.12 – Le temps d'exécution de la charge de travail

pouvons conclure que la construction de l'IJB sur des attributs de FAN-OUT de faible valeur, qui dépasse à peine le 1% de la table de faits, présente un avantage réel.

En ce qui concerne le nombre de jointures et de sélections des requêtes de la charge de travail du benchmark SSB, nous soulignons que toutes les 13 requêtes satisfont une particularité très importante, qu'elles contiennent plusieurs conditions dans la clause *Where*. Dans ce contexte, les IJB viennent jouer et apportent une valeur ajoutée déterminante. D'autre part lors de l'exécution d'une requête en présence de l'IJB, les lignes qui permettent de satisfaire certaines conditions de la clause *Where* de la requête sont filtrées avant d'accéder à la table elle-même comme expliqué dans la documentation d'Oracle [4]. Ainsi, l'optimiseur des requêtes favorise l'utilisation de l'IJB, ce qui engendre une modification de l'ordre des jointures dans le plan d'exécution. Ces modifications influencent beaucoup le plan d'exécution et le rendent plus optimisé.

Dans l'ensemble, les résultats de l'étude expérimentale apportent la preuve que notre approche, qui élague l'espace de recherche grâce au FAN-OUT et qui se base sur une modélisation en hypergraphe de la charge de travail et utilise les traverses minimales pour proposer une configuration d'indexes conduite par une sélection multi-objectifs de l'IJB, donne des résultats très encourageants surtout en matière d'économie d'énergie et pour la performance elle surpasse les autres approches de la littérature.

4.6 Conclusion

Dans ce chapitre, nous avons proposé une initiative permettant d'intégrer l'énergie dans la conception physique des entrepôts de données relationnel, en considérant le cas des indexes

4. https://docs.oracle.com/cd/B28359_01/server.111/b28313/indexes.htm

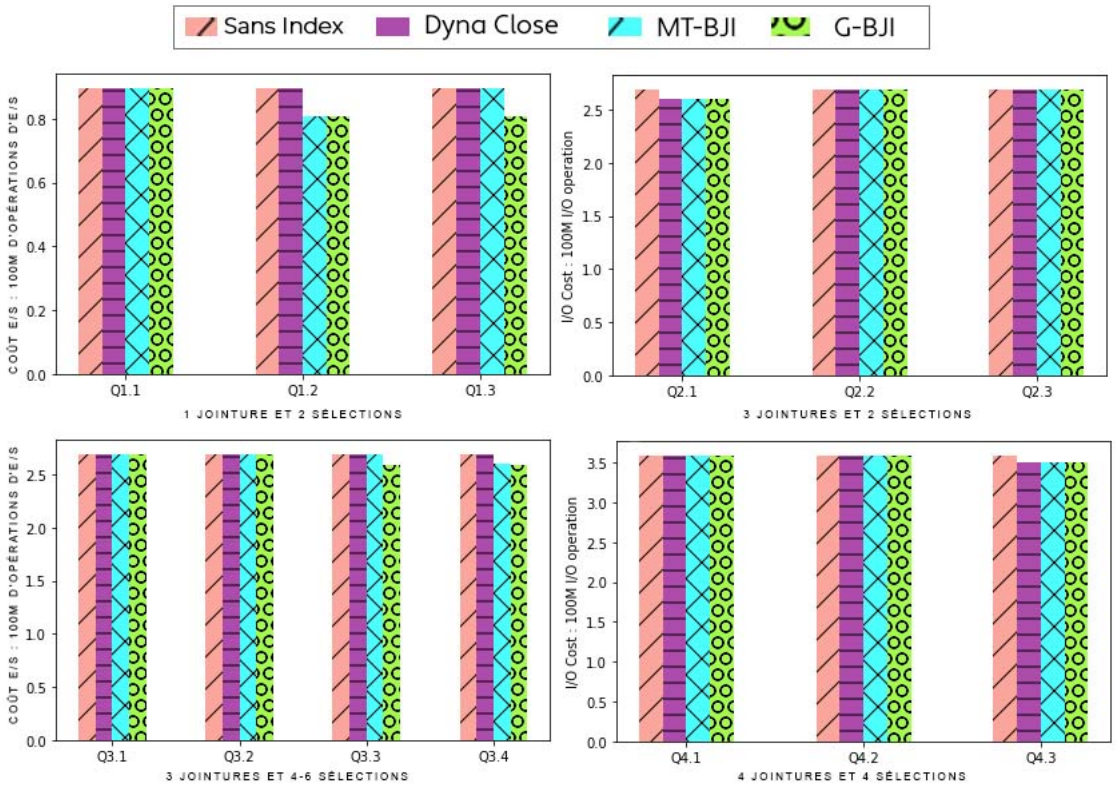


FIGURE 4.13 – Le coût d'exécution de la charge de travail en opérations d'E/S

de jointure binaire, qui sont une structure d'optimisation redondante. Afin de proposer une approche pour la sélection des IJB, nous avons réalisé un audit pour comprendre leur comportement énergétique. Cet audit, nous a permis de détecter un paramètre très important que nous l'avons appelé FAN-OUT. Ensuite, nous avons proposé une nouvelle formalisation du $PSIJB$, dans laquelle nous intégrons l'aspect énergétique. Dans ce chapitre, nous avons introduit un nouvel algorithme sensible à l'énergie pour la sélection de ces indexes, qui se base sur une modélisation par un hypergraphe de la charge de travail et sur le FAN-OUT pour l'élagage des attributs indexables ainsi que l'opérateur de Skyline pour la sélection multi-objectifs.

L'étude expérimentale réalisée valide notre démarche, étant donné que l'approche proposée permet de réduire considérablement la consommation d'énergie globale de la charge de travail, de réduire également le temps d'exécution et nous a permis de caractériser la puissance maximale et la puissance moyenne. D'autre part, cette étude montre le rôle important que joue le FAN-OUT et son impact sur l'optimiseur de requêtes pour choisir les plans d'exécutions des requêtes et ordonner les jointures. En outre, ces expérimentations changent la vision traditionnelle de la charge de travail, où les attributs indexables extraits des requêtes ont la même chance d'être indexés en une vision orientée instances. Dans cette nouvelle vision, les attributs indexables extraits de la charge de travail doivent être analysés en fonction de leur participation à la table de faits. Cette participation a été capturée par le FAN-OUT, qui déterminera la pertinence de ces attributs.

Détecter et réduire l'énergie lors de la phase de conception physique n'est pas la seule solution pour améliorer l'efficacité énergétique d'une base de données. Dans le chapitre suivant, nous allons essayer de capter l'énergie à une étape antérieure dans le cycle de vie de la BD et étudier l'impact de la variabilité du schéma logique sur la consommation de l'énergie.

Chapitre 5

L'impact de la conception logique sur la consommation d'énergie des bases de données

5.1 Introduction

Le concept de l'énergie est omniprésent tout au long du cycle de vie d'une base de données comme nous l'avons montré dans le chapitre 3. Ayant l'objectif d'optimiser ce besoin non-fonctionnel dans notre esprit, nous essayons dans ce chapitre de le traiter à une étape antérieure. La conception logique est une phase très importante durant le cycle de vie d'une base de données en général et d'un entrepôt de données en particulier. À la fin de cette phase, le modèle logique est décrit dans un langage de définition des données (LDD), et il peut être complété, en cas de besoin, lors de la phase de conception physique. Capturer l'énergie à ce niveau, nous permettra d'améliorer considérablement l'efficacité énergétique de la base de données lors de sa mise en production. Un schéma logique peut être modélisé de plusieurs manières différentes, et ce, à cause des attributs hiérarchiques des tables et les formes normales appliquées sur chaque table. Lorsque le nombre de tables et les attributs par table deviennent importants, l'énumération exhaustive de tous les schémas logique possibles devient peu pratique, voire impossible, pour explorer complètement l'espace de recherche qui évolue d'une forme proportionnelle en nombre d'attributs.

Pour résoudre ce problème, nous présentons, dans ce chapitre, une approche de sélection du schéma logique le moins énergivore dans laquelle nous introduisons une technique pour élaguer l'espace de recherche. Nous entamons ce chapitre par la présentation des hypothèses que nous avons considérées et sur lesquelles se fonde notre méthodologie afin de proposer notre approche. Nous présentons ensuite quelques notions de bases et les piliers de notre approche. Juste après, nous donnons la démarche et les différentes étapes permettant de choisir le schéma logique le plus écologique d'une base de données. Un exemple illustratif est également fourni pour mieux comprendre notre démarche. L'algorithme Ls-Energy permettant d'implémenter notre approche est donné par la suite. Enfin, pour montrer l'utilité et l'efficacité de notre approche, nous effectuons une étude expérimentale dans laquelle nous évaluons notre approche à travers un benchmark réel. Nous terminons le chapitre par une conclusion dans laquelle nous résumons les principaux résultats.

5.2 Hypothèses

Dans cette section, nous présentons les hypothèses sur lesquelles nous nous sommes basées pour proposer notre approche. Nous commençons par considérer un exemple qui est utilisé pour mieux expliquer nos hypothèses.

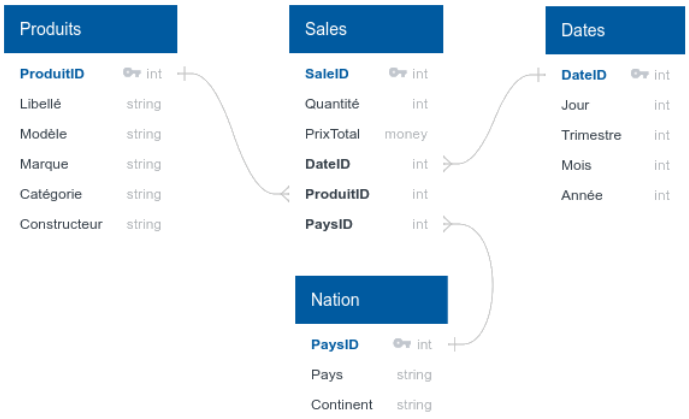


Figure 5.1 – Schéma logique *L*

Soit un schéma logique L composé de 3 tables de dimension. Soit Dim l'ensemble de tables de dimensions $Dim = \{Nation, Dates, Produits\}$, et pour chaque table de dimension D de Dim, il existe un ensemble d'attributs hiérarchique $H_D = \{a_1, a_2, \ldots, a_m\}$. Ainsi, pour chaque table de dimension D ayant m attributs hiérarchiques, D peut être décomposée 2^{m-1} fois. Soit la table de dimension $Dates = \{DateID, Jour, Mois, Trimestre, Année\}$, cette table comporte une hiérarchie de 4 niveaux, il y aura ainsi 8 (2^{4-1}) possibilités de décomposition de la table, comme illustré à travers un diagramme de *Hasse* dans la figure 5.2. Et pour l'ensemble des tables de dimensions Dim, l'univers de décompositions possibles de toutes les tables, est le produit des possibilités de chaque dimension : $\prod_{d=1}^{n} 2^{H_D-1}$ où H_D est le nombre des hiérarchies de la dimension D. Et comme les deux autres tables de dimensions ($Nation$ et $Produits$) comportent chacune 2 et 3 attributs hiérarchiques respectivement, par conséquent, ce schéma logique admet $2^{2-1} \times 2^{3-1} \times 2^{4-1} = 64$ décompositions possibles.

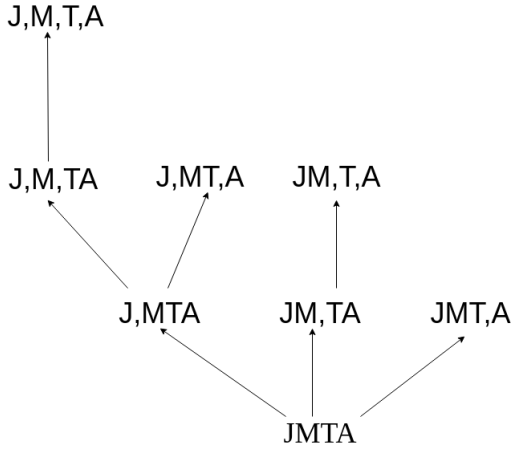


FIGURE 5.2 – Le diagramme de Hasse de notre décomposition

Définition 11 (Relation d'ordre) *: Une relation réflexive, antisymétrique et transitive, s'appelle une relation d'ordre. Un ensemble muni d'une relation d'ordre est un ensemble ordonné.*

On dit qu'une relation binaire R définie sur un ensemble E est :

- *réflexive si* $(\forall x \in E)\ xRx$.
- *symétrique si* $(\forall (x, y) \in E^2)\ (xRy) \Leftrightarrow (yRx)$.
- *transitive si* $(\forall (x, y, z) \in E^3)\ (xRy) \wedge (yRz) \Rightarrow (xRz)$. *Si x précède y et si y précède z alors x précède z.*
- *antisymétrique si* $(\forall (x, y) \in E^2)\ (xRy) \wedge (yRx) \Rightarrow (x = y)$. *Si x et y sont différents, et si x précède y, alors y ne peut pas précéder x (sinon ce n'est plus une hiérarchie...).*

Il est possible de représenter les ensembles ordonnés $(B, \preceq)$ finis et suffisamment petits par un diagramme appelé diagramme de $Hasse$. Les éléments de B sont les sommets du diagramme et certains couples de la relation sont représentés par des arcs (lignes entre les sommets). Si $b \prec c$, alors le sommet b est placé plus bas que le sommet c. Si $b \prec c$ et qu'il n'y a pas d'élément d tel que $b \prec d \prec c$, un arc est tracé entre b et c.

Exemple 5 *Nous supposons qu'il existe une relation d'ordre entre les différentes décompositions possibles d'une table en se basant sur les différents attributs hiérarchiques. Si une décomposition D_1 précède une décomposition D_2, alors la consommation énergétique de D_2 est inférieure à D_1.*

Dans notre exemple, nous considérons l'ensemble des décompositions possibles des attributs hiérarchiques d'une table. Nous considérons la table $Dates$ avec ses attributs hiérarchiques (Jours, Mois, Trimestre et Année), et nous représentons chaque attribut par sa première lettre (i,e, Jour par J, Trimestre

par T etc,). Le diagramme de Hasse *présenté dans la figure 5.2 montre les différentes décompositions possibles de la hiérarchie en partant de la décomposition la plus dé-normalisée* ($JMTA$) *représentée par la table* $Dates$ *comme le montre la figure 5.1 à la décomposition la plus normalisée (J,M,T,A) de la même figure en passant par les différentes décompositions possibles comme par exemple (J,M,TA). Nous explorons le diagramme de Hasse du bas en haut, du niveau de granularité le plus épais au niveau de granularité le plus fin. Nous gardons en tête que plus la table est dé-normalisée plus sa taille augmente plus l'opération de jointure devient gourmande en ressources. Ainsi, il existe une relation d'ordre entre les différentes décompositions en partant de la moins normalisée au plus normalisée.*

Selon Roukh et al. [152] et Bouarrar et al. [23], pour certaines requêtes, la lecture de gros fichiers de données nécessite plus d'opérations d'E/S que de traitement par le CPU (les opérations fortement consommatrices, les opérations d'agrégation, de tri, etc.). Ensuite, l'exécution des requêtes sur des tables de petite taille nécessite parfois plus de puissance que celles exécutées sur des tables de grande taille. Cependant, lorsque les résultats intermédiaires ne peuvent pas être stockés en mémoire, ils seront écrits sur le disque et lus ultérieurement. Cela entraîne une baisse de consommation de ressources de traitement, car la requête passe plus de temps à lire/écrire ses données qu'à traiter ses enregistrements. Néanmoins, les requêtes dominées par les opérations d'E/S consomment moins d'énergie. En revanche, lorsque la taille des données est faible, la lecture des données se termine rapidement et pendant le temps restant, l'exécution de la requête est dominée par le traitement du processeur, qui est gourmand en énergie.

Dans leurs travaux, Bouarrar et al. [23] ont mis en exergue deux faits qui méritent d'être mentionnés :

— faire varier le schéma logique a un impact sur la consommation d'énergie, et mieux encore, cela montre que le schéma en étoile est loin d'être le plus écologique.
— la normalisation des tables de plus petite taille en présence d'opérations gourmandes en CPU représente un inconvénient certain en termes de consommation d'énergie.

Sur la base des résultats des travaux de Roukh et al. [152] et Bouarrar et al. et la modélisation proposée, nous présentons les hypothèses que nous avons considérées pour établir notre approche :

1. Le schéma en étoile n'est pas le schéma logique optimal et il faut le varier, en le considérant comme un point de départ.
2. Donner la priorité à la normalisation des tables les plus grandes pour éviter les jointures supplémentaires des tables de faible taille.
3. Il existe une relation d'ordre entre les différentes décompositions.
4. Si la normalisation d'une table de dimension n'améliore pas la consommation d'énergie actuelle du schéma logique, il n'est pas nécessaire de continuer la normalisation de cette table.

En s'appuyant sur les hypothèses précédemment décrites, nous proposons une nouvelle approche appelée LS-Energy pour générer des schémas logiques. Cette approche nous permettra de choisir le schéma logique le plus écologique. Néanmoins, explorer tout l'espace de recherche composé de tous les schémas logiques possibles est tout simplement irréalisable. C'est pourquoi nous devons élaguer efficacement l'espace de recherche. Pour ce faire, nous nous basons sur la contrainte d'anti-monotonie de la consommation d'énergie.

5.3 LS-Energy : une approche écologique de sélection de schéma logique

Avant de détailler l'approche que nous introduisons et pour faciliter sa compréhension, nous commençons par définir deux concepts fondamentaux sur lesquels reposent fortement

notre approche : la variabilité qui nous permettra de générer les différents schémas logiques et la contrainte d'anti-monotonie qui nous permettra d'élaguer l'espace de recherche.

5.3.1 La variabilité

La plupart des entreprises proposent des familles de systèmes et de solutions logicielles qui sont très similaires. Ces dernières, ne se différencient que par quelques caractéristiques. C'est pour cette raison que la réutilisation a été et reste toujours un domaine important du génie logiciel. Traditionnellement, lors de la conception et le développement des logiciels, chaque produit logiciel se construisait d'une manière isolée, dans un but très précis et avec des fonctionnalités bien déterminées. Il s'est avéré que cette méthodologie est très coûteuse et longue, et elle a donc rapidement évolué vers une production de masse appelée "*lignes de production*". Au départ, des composants de base (éléments constitutifs communs), qui forment le pilier de tous les produits, sont développés, ensuite une personnalisation est appliquée pour réaliser les différents produits finaux.

La variabilité est définie par la capacité d'un système à se configurer, à s'adapter et à se spécialiser en fonction du contexte de son utilisation. Une amélioration de la variabilité d'un système implique une amélioration de la facilité d'effectuer certains types de changements.

Définition 12 (Variabilité) *: La variabilité est la capacité d'un système à être efficacement étendu, adapté ou configuré pour un usage donné dans un contexte particulier [166]. Elle est la description des variations possibles d'un système par des points de variation.*

Définition 13 (Point de variation) *: Un point de variation identifie et localise l'endroit ou se produit la variabilité. Il précise les solutions possibles de résolution de cette variabilité (explicitation des variants), et éventuellement le moment de cette prise de décision [38].*

Cette définition montre qu'un variant est une réalisation possible de la variabilité indiquée par le point de variation. Le choix et la fixation de la variabilité peuvent intervenir à différents moments du cycle de vie du système, de la conception, à la compilation, à l'édition de liens, au déploiement, et jusqu'à l'exécution. La variabilité peut être définie à toutes les étapes du processus de développement, et dans une approche orientée modèles à différents niveaux d'abstraction.

Lors de la conception d'un système, il est toujours envisageable d'anticiper et d'identifier un certain type de variabilité afin de construire un système d'une manière qui permet d'accepter cette variabilité. La diversité implique le besoin de faire adapter les systèmes à un contexte particulier, ce qui est désigné par la variabilité dans le vocabulaire du génie logiciel.

La notion de la variabilité et sa gestion doivent être traitées d'une manière plus large, couvrant non seulement sa définition initiale établie dans le domaine du génie logiciel, mais aussi les différents concepts, outils et techniques qui visent à maîtriser les évolutions et la variété et ainsi à réduire la complexité du processus de conception des systèmes. La gestion de la variabilité inclut les activités suivantes : l'élicitation et l'explicitation de la variabilité des artefacts tout au long du cycle de vie, la gestion des dépendances entre ces différentes variabilités et le support qui permettra l'instanciation de ces variabilités [38].

La gestion de la variabilité d'une application est la capacité d'exprimer les variations possibles que peut avoir cette application, en se basant sur deux étapes : l'identification de la variabilité, et l'implémentions de la variabilité.

La première étape consiste à la détection des caractéristiques en commun et des caractéristiques distinctes entre les différents produits d'une même famille et à définir l'architecture permettant la réutilisation. Cette étape se compose elle-même de deux parties. La modélisation où plusieurs techniques ont été proposées dans la littérature, comme l'Analyse de domaine

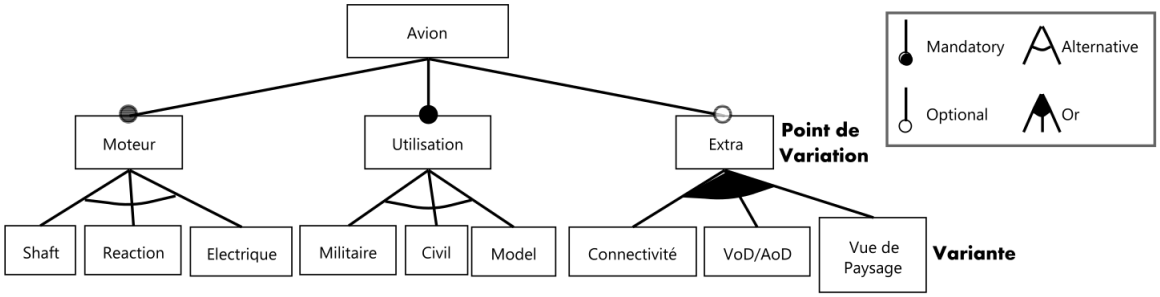


FIGURE 5.3 – Diagramme de fonctionnalités [22]

orientée *features* (fonctionnalité),(FODA) [84] à titre d'exemple. Dans les systèmes modélisés par le concept de fonctionnalités, la variabilité est facilement identifiée.

La figure 5.3 montre un exemple d'un diagramme de fonctionnalités. Ce dernier prend toujours la structure d'un arbre dont les nœuds représentent les fonctionnalités. Dans un tel diagramme, la variabilité est représentée à travers les arcs ainsi que les groupements des fonctionnalités. Il englobe des points de variations obligatoires ou optionnels. Chaque nœud peut avoir des feuilles sous forme de groupes de variantes exclusifs ou alternatifs. Une fois que la modélisation est effectuée et les fonctionnalités sont identifiées, toutes les combinaisons des différentes fonctionnalités ne sont pas forcément valides. Ainsi, certains choix sont incompatibles entre eux. Pour trouver les combinaisons valides d'une manière efficace, l'ajout des contraintes de cohérence de type *impliquer* et *exclure* permet de faciliter cette tâche.

La deuxième étape consiste à définir la manière dont les produits finaux vont être dérivés, et à construire un produit particulier. En effet, il s'agit de choisir, parmi la collection qui existe, la technique la plus appropriée pour réaliser la variabilité. Cela se déroule en deux sous-étapes :

— la configuration : où l'utilisateur commence par la sélection des fonctionnalités concernées par la variabilité définie, qui correspondent à ses exigences. Un choix particulier lors de la dérivation d'un produit peut exclure ou exiger d'autres choix (en se basant sur les contraintes précédemment définies) ;
— la dérivation : chaque fonctionnalité est instanciée et remplacée par son rôle (code, script, test, documentation, modèle, etc.).

En l'appliquant aux $SGBD$s, la variabilité permet d'offrir une solution à mi-chemin entre l'empilement peu performant d'un large éventail de fonctionnalités retrouvé dans les $SGBD$s génériques, et le re-développement parfois très coûteux, en partant de zéro, de grandes parties des $SGBD$s. Également, nous pouvons le retrouver dans les solutions à usage spécifique. Ainsi, la variabilité offre une personnalisation des solutions selon le contexte, en se basant sur la réutilisation et la configurabilité des composants communs et variables définis au préalable et déjà implémentés. La variabilité existe tout au long du cycle du vie d'une base de données, et elle est présente dans toutes les phases.

- Phase conceptuelle : La variabilité concerne les formalismes qui peuvent être utilisés :
  — Merise.
  — UML.
- Phase logique : Dans cette phase, la variabilité peut impliquer quatre points de variation :
  — Le modèle (relationnel/objet/NoSQL).
  — Les règles de mapping.
  — La normalisation ($1FN$, $2FN$, $3FN$, $FNBC$).
  — L'optimisation logique : les techniques utilisées (règles algébriques et/ou les modèles de coût) afin d'obtenir le meilleur plan avant de recourir aux structures d'optimisation physiques.

- Phase physique : La variabilité au sein de cette phase, peut concerner plusieurs niveaux différents comme :
  — Plateforme de déploiement : le traitement des données peut être local ou distribué.
  — Support de stockage : disques durs magnétique (HDD), disques électroniques (SSD), disques en RAID, mémoires centrales, mémoires flash, etc.
  — Le "layout" de stockage désigne la façon dont les données sont physiquement organisées (verticale/horizontale).
  — Structures d'optimisation : la fragmentation (horizontale/verticale), les indexes, les vues matérialisées, l'ordonnancement de requêtes, la gestion de buffer.
  — Moteur de traitement : concerne les composants matériels qui exécutent les différentes opérations élémentaires, comme CPU, GPU.

5.3.2 La contrainte d'anti-monotonie

Dans le cas où un espace de recherche noté L est gigantesque, il n'est généralement pas possible de l'explorer d'une manière exhaustive, ainsi, il faut réussir à n'en parcourir qu'une partie. Par contre, il faut être sûr d'avoir extrait tous les motifs qui satisfont au prédicat p. Cela signifie qu'à chaque fois que l'on décide de ne pas explorer toute une partie de l'ensemble L (élaguer une partie de L), il faut pouvoir prouver qu'aucun motif satisfaisant le prédicat p n'y présent.

Soit $Items = \{x_1, ..., x_1\}$ un ensemble d'éléments distincts, généralement appelés *items*. Un ensemble d'éléments ($un\ itemset$) X est un sous-ensemble non vide d'éléments. Si $|X| = k$, X est appelé un k-itemset. Une transaction est un couple $\langle tID, X\rangle$ où tID est l'identifiant de la transaction et X est le contenu de la transaction (un ensemble d'éléments). Une base de données de transactions BDT est un ensemble de transactions. Un ensemble d'éléments X est contenu dans une transaction $\langle tID, Y\rangle$ si $X \subseteq Y$. Étant donné une base de données de transactions BDT, le sous-ensemble de transactions qui contient un ensemble d'éléments X est nommé $BDT[X]$. Le *support* d'un ensemble d'éléments X, écrit $supp_{BDT}(X)$ est la cardinalité de $BDT[X]$. Étant donné un support minimum défini par l'utilisateur δ, un ensemble d'éléments X est appelé fréquent dans BDT si $supp_{BDT}(X) \geqslant \delta$.

Ceci définit la contrainte de fréquence $C_{freq}[BDT]$: si X est fréquent on écrit $C_{freq}[TDB](X)$ ou simplement $C_{freq}(X)$. Soit $Th(C) = \{X|C(X)\}$ l'ensemble de tous les itemsets X qui satisfont la contrainte C. Le problème d'extraction d'items fréquents nécessite de calculer l'ensemble de tous les itemsets fréquents $Th(C_{freq})$. En général, étant donné une conjonction de contraintes C, le problème d'exploration d'items contraints nécessite de calculer $Th(C)$; le problème d'extraction d'items fréquents contraints nécessite de calculer $Th(C_{freq}) \cap Th(C)$.

Une contrainte C est anti-monotone, ssi, pour chaque ensemble d'éléments E, si E ne satisfait pas C, alors aucun de ses sur-ensembles ne satisfait C.

Définition 14 (Contrainte d'anti-monotonie) *: Une contrainte C est anti-monotone, si pour une cellule de haut niveau c_h et une cellule de bas niveau c_b couverte par c_h, ce qui suit doit être vérifié : c_h ne satisfait pas $C \Rightarrow c_b$ ne satisfait pas C [191].*

La contrainte de fréquence présente un bon exemple dans lequel se manifeste la contrainte d'anti-monotonie. Cette propriété est utilisée par l'algorithme Apriori [1] avec l'heuristique suivante : si un ensemble d'éléments X ne satisfait pas C_{freq} , alors aucun sur-ensemble de X ne peut satisfaire C_{freq}, et donc ils peuvent être élagués. L'algorithme Apriori fonctionne de manière ascendante sur l'ensemble des éléments, et chaque fois qu'il trouve un ensemble d'éléments peu fréquent (inférieur à une valeur précisée en avance), il supprime tous ses super-ensembles.

Notre approche commence par évaluer la consommation énergétique du schéma en étoile et le considère comme la solution optimale. Ensuite, nous trions par ordre décroissant toutes les

tables de dimensions en fonction de leurs tailles. Après cela, pour chaque table de dimension, nous décomposons sa hiérarchie et nous générons son schéma logique, nous réécrivons les requêtes, puis, nous estimons sa consommation d'énergie et nous la comparons à la solution optimale actuelle :

— si la nouvelle valeur est meilleure que la solution optimale, alors, elle devient la solution optimale. Ensuite, nous passons au niveau suivant de la hiérarchie et nous répétons le même traitement jusqu'à ce que nous explorions toute la hiérarchie.

— si la nouvelle valeur est pire que la solution optimale actuelle, on s'abstient de continuer la décomposition de cette hiérarchie et nous passons à la table de dimension suivante, et à ce niveau se manifeste la contrainte d'anti-monotonie.

Dans notre approche, grâce à la contrainte d'anti-monotonie, nous sommes en mesure d'élaguer l'espace de recherche. Nous n'énumérons pas tous les schémas possibles ce qui est loin d'être une solution pratique.

Pour évaluer la consommation énergétique de chaque schéma, nous ajustons le modèle de coût mathématique introduit par Roukh et al. [152]. L'ajustement a été effectué pour rendre le modèle de coût plus générique et pour considérer toutes les variantes des schémas logiques. Puisqu'il a été construit en supposant un entrepôt de données avec un schéma en étoile. L'ajustement concerne principalement la phase d'apprentissage [23]. Grâce à ce modèle de coût, nous pouvons évaluer théoriquement l'énergie consommée de chaque schéma logique. Il convient de mentionner, le processus de génération de chaque schéma logique est basé sur les corrélations des attributs. Les requêtes de la charge de travail sont aussi modifiées et réécrites pour qu'elles soient adaptées au nouveau schéma.

5.3.3 Les étapes de notre approche

Les étapes de notre approche et leurs interactions sont présentées dans la figure.5.5. Dans ce qui suit, nous décrivons en détail les étapes de notre approche énergétique de sélection du schéma logique.

Étape 1 : Initialisation : dans cette étape nous commençons par l'estimation de la consommation énergétique du schéma initiale, qui est le schéma en étoile pris en entrée. Ensuite, nous considérons le schéma en étoile comme la solution optimale et sa consommation d'énergie comme la consommation énergétique optimale. Après, nous trions d'une manière décroissante les tables de dimensions par rapport à leurs tailles de stockage sur le disque.

Étape 2 : Génération des décompositions des tables de dimension : afin de pouvoir générer les différentes décompositions possibles de chaque hiérarchie, nous avons proposé l'algorithme TableDecomposer dont le pseudo-code est donnée dans l'Algorithme 4. Cet algorithme se base sur la construction d'un arbre, dans lequel chaque nœud est composé d'une partie fixe non-décomposable notée "*base*", et d'une autre partie variable "*decomp*". Cette partie contient les attributs hiérarchiques qui sont décomposables. La figure 5.4 montre le résultat de l'exécution de cet algorithme sur la table $Dates$ de l'exemple 5.

Étape 3 : Génération des schémas logiques : une fois que nous avons toutes les décompositions possibles d'une table de dimension, nous passons à la génération du schéma logique correspondant. Pour ce faire, pour chaque décomposition, nous créons les nouvelles tables de sous-dimensions, nous mettons à jour les attributs nécessaires et nous établissons les liens entre les tables de sous-dimensions. Cette étape est automatisée grâce à l'algorithme SchemaGenerator dont le pseudo-code est donnée dans l'algorithme 5.

Étape 4 : Remplissage des tables de sous-dimensions : à partir des tables de dimension d'origine, nous redistribuons les données dans les nouvelles tables de dimensions correspondantes tout en ajoutant les liens de clé primaire-clé étrangère dans ces dernières.

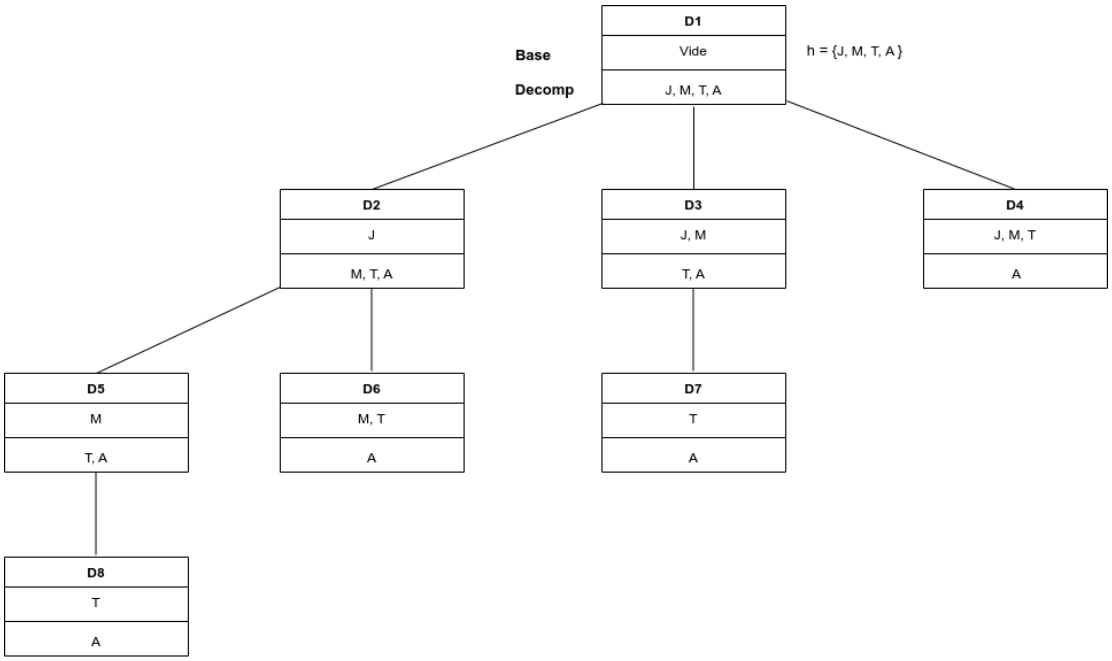


FIGURE 5.4 – Exécution de l'algorithme TABLEDECOMPOSER

Algorithme 4 : TABLEDECOMPOSER

Entrées : Une hiérarchie h qui contient n attributs hiérarchiques : $h = \{a_1, a_2, \ldots, a_n\}$;
Sorties : L'ensemble des différentes décompositions ;

```
1  int nbFils = 0;
2  String base ← "";
3  String decomp ← " a1 a2 ... an ";
4  procedure CreateTree()
5  {
6  si decomp.length() ! = 1 alors
7      nbFils = decomp.length() − 1;
8      pour i := 1 à decomp.length() − 1 faire
9          String f = decomp.substring(0, i);
10         String v = decomp.substring(i);
11         Tree subTree = new Tree(f, v);
12         this.Fils[i − 1] = subTree;
13         this.Fils[i − 1].CreateTree();
14 }
15 procedure Tree(String base, String decomp)
16 {
17 this.base = base;
18 this.decomp = decomp;
19 this.Fils = new Tree[decomp.length()−1];
20 this.nbFils = 0;
21 }
```

Algorithme 5 : SchemaGenerator

Entrées : Un schéma d'entrepôt de données en étoile ;
Sorties : Un ensemble de schémas logiques ;

```
pour chaque table de dimension D faire
    TableDecomposer (Les attributs hiérarchiques de D) ;
    pour chaque décomposition de D faire
        Créer les nouvelles tables de sous-dimensions ;
        Mettre à jour les attributs ;
        Ajouter la décomposition courante à la liste correspondante à la dimension en
         cours ;
Construire le nouveau schéma correspondant à chaque élément appartenant à ce
 produit cartésien ;
```

Étape 5 : Réécriture des requêtes : les requêtes du benchmark utilisé sont réécrites en fonction du schéma logique cible. En effet, les attributs référencés par les requêtes peuvent être déplacés vers les nouvelles tables de sous-dimensions pendant l'éclatement des dimensions d'origine. Pour ce faire, nous avons proposé l'algorithme QueryRewriter dont le pseudo-code est donnée dans l'Algorithme 6. Cet algorithme nous permet d'effectuer la réécriture syntaxique d'une requête SQL, d'un schéma initial à un schéma cible.

Algorithme 6 : QueryRewriter

Entrées : Un schéma logique cible SL ;
Une charge de travail d'origine : Q = $\{q_1\ q_2 ... q_m\}$;
Sorties : Une charge de travail réécrite Q' = $\{q_1^{'}\ q_2^{'} ... q_m^{'}\}$;

```
pour chaque requête q_i de Q faire
    Extraire les attributs de la clause "Select" ;
    Chercher ces attributs dans les tables de SL (donnée en entrée) ;
    Redéfinir les clauses "Select" et "From" de la nouvelle requête q_i' ;
    Mettre à jour la clause "From" de la nouvelle requête (q_i') ;
    Extraire les attributs de la clause "Where" de (q_i) ;
    Chercher ces attributs dans les tables de SL ;
    Redéfinir la clause "Where" de la nouvelle requête (q_i') ;
    Mettre à jour la clause "From" de la nouvelle requête (q_i') ;
    Ajouter les jointures supplémentaires résultantes de l'éclatement des dimensions ;
    Ajouter (q_i') à Q'
```

Étape 6 : Exécution des requêtes et estimation du coût : une fois que le schéma logique soit prêt et peuplé et les requêtes sont réécrites, dans ce cas nous passons à l'étape la plus importante de notre approche. Nous exécutons les requêtes et nous estimons leur consommation énergétique grâce au modèle du coût énergétique proposé dans [152] et détaillé dans la section 3.4.2.1.

Étape 7 : Choix du schéma logique optimal : pour chaque décomposition de chaque table de dimension, nous comparons sa consommation énergétique avec celle du schéma logique optimal, si elle est meilleure, la décomposition actuelle devient le schéma logique optimal et sa consommation énergétique devient la consommation énergétique optimale, sinon, on s'abstient de décomposer encore cette table de dimension et nous passons à la table

suivante jusqu'à la fin de toutes les tables de dimension. C'est à ce niveau où se manifeste la contrainte d'anti-monotonie.

Étape 8 : Le schéma logique final : lorsque toutes les tables de dimension sont traitées, l'algorithme s'arrête et retourne le schéma optimal avec sa consommation énergétique comme la solution finale.

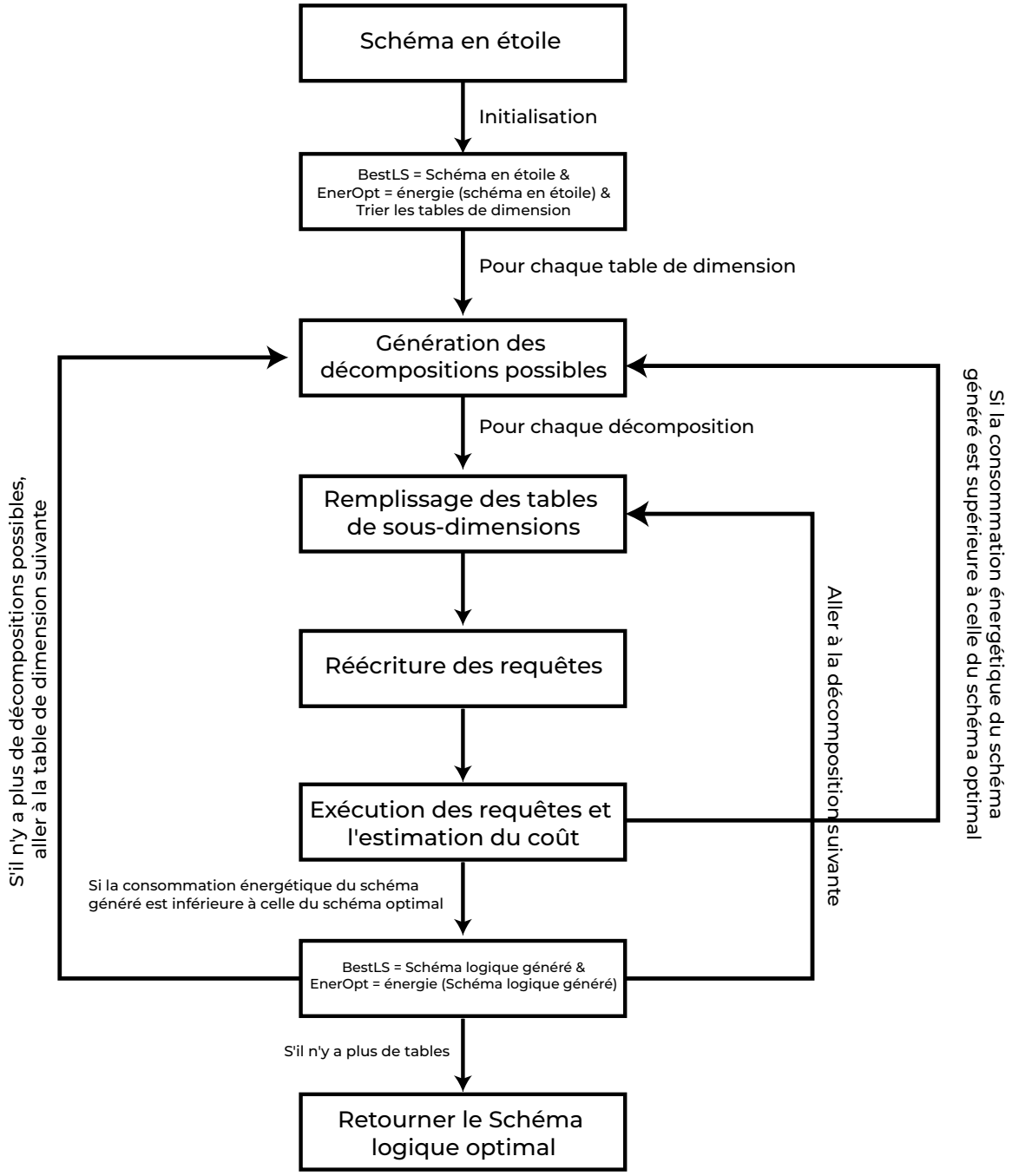


FIGURE 5.5 – Les étapes de l'approche LS-Energy

5.3.4 Exemple illustratif

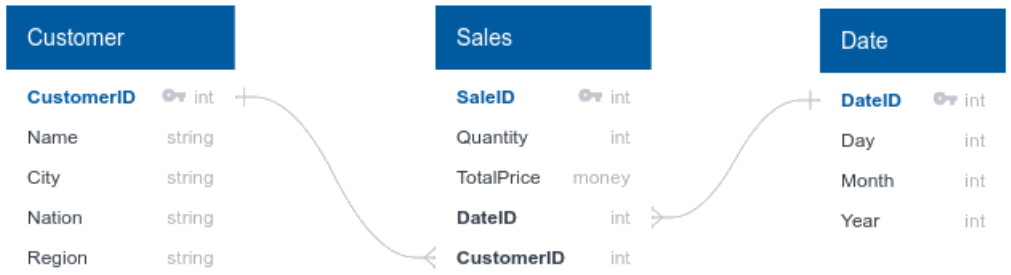


FIGURE 5.6 – Le schéma en étoile

```
SELECT City, Month
FROM Sales S, Customer C, Date D
WHERE S.DateId = D.DateId AND S.CustomerId = C.CustomerId
AND City = 'Vilnius' AND Month = 'December'
```

Considérons le schéma en étoile représenté sur la figure 5.6 et la requête ci-dessus à exécuter. Le schéma de la BD contient deux tables de dimensions *Date* et *Customer*, et une table de faits *Sales*. La table *Customer* contient trois attributs hiérarchiques (*City, Nation* et *Region*). La table *Date* possède également trois attributs hiérarchiques (*Month, Quarter* et *Year*). La taille de la table *Customer* est plus grande que celle de la table *Date*. Nous commençons par évaluer la consommation énergétique du schéma en étoile (figure 5.6).

FIGURE 5.7 – Le schéma *LS*1

Ensuite, nous décomposons la plus grande table en mettant ses attributs hiérarchiques dans une autre table. Nous décomposons la table *Customer* comme le montre la figure.5.7 pour obtenir un nouveau schéma noté *LS*1, nous réécrivons la requête et nous évaluons sa consommation énergétique. Le code *SQL* de la requête réécrite est donné en dessous.

```
SELECT City, Month
FROM Sales S, Customer C, C_City CC, Date D
WHERE S.DateId = D.DateId AND S.CustomerId = C.CustomerId
AND C.CityId = CC.CityId
AND City = 'Vilnius' AND Month = 'December'
```

Comme la consommation énergétique du schéma *LS*1 est inférieure à celle du schéma en étoile, on le considère comme la solution optimale actuelle et nous continuons la décomposition pour avoir un nouveau schéma noté *LS*2 (figure 5.8).

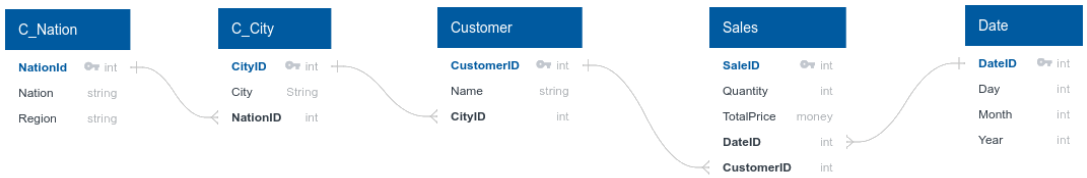


FIGURE 5.8 – Le schéma *LS*2

```
SELECT City, Month
FROM Sales S, Customer C, C_City CC, C_Nation CN, Date D
WHERE S.DateId = D.DateId AND S.CustomerId = C.CustomerId
AND C.CityId = CC.CityId AND CC.NationId = CN.NationId
AND City = 'Vilnius' AND Month = 'December'
```

Nous évaluons sa consommation d'énergie. Cette dernière n'ayant pas été améliorée, on s'abstient de continuer la décomposition de la table *Customer* et on passe à l'autre table de dimension, c'est-à-dire la table *Date*. Ainsi, nous gardons le schéma *LS*1 comme la solution optimale et nous décomposons la table *Date* pour obtenir un nouveau schéma noté *LS*3(figure.5.9). Nous mesurons sa consommation d'énergie et nous la comparons à la solution optimale actuelle. Puisque la consommation d'énergie de *LS*3 n'est pas meilleure que *LS*1, et que la table *Date* est la dernière table de dimension, alors, nous arrivons à la fin du processus de décomposition et nous considérons *LS*1 comme la solution finale, ce qui est le schéma logique le plus écologique.

```
SELECT City, Month
FROM Sales S, Customer C, C_City CC, Date D, D_Day
WHERE S.DateId = D.DateId AND S.CustomerId = C.CustomerId
AND C.CityId = CC.CityId AND D.DayId = DD.DayID
AND City = 'Vilnius' AND Month = 'December'
```

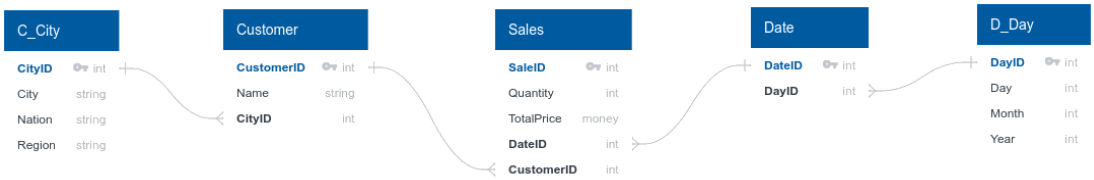


FIGURE 5.9 – Le schéma *LS*3

5.4 L'algorithme de LS-Energy

Notre algorithme, dont le pseudo-code est donné dans Algorithme 7, prend en entrée un schéma en étoile d'un entrepôt de données, composé d'une table de faits TF et un ensemble de table de dimensions $(D_1, D_2, ..., D_n)$, et d'une charge de travail Q composée de m requêtes. La première étape consiste à évaluer sa consommation énergétique $CM_{energ}(SS)$ et attribuer cette valeur à EC_{opt} qui est la consommation énergétique optimale d'un schéma logique (ligne 2). Ensuite, nous trions par ordre décroissant les tables de dimensions par rapport à leurs tailles respectives (ligne 3). Après cela, pour chaque table de dimension, nous générons l'ensemble des décompositions (lignes 4 & 5). Ensuite, pour chaque décomposition, nous générons le schéma

logique correspondant (lignes 6 & 7). À l'étape suivante, pour chaque schéma logique, nous réécrivons les requêtes pour qu'elles conviennent au schéma. Subséquemment, nous les exécutons ces requêtes et nous capturons leur consommation d'énergie (lignes 9 à 11). Ensuite, nous comparons la consommation énergétique de ce schéma (EC_{ls}) avec la consommation énergétique optimale (ligne 12). Si (EC_{ls}) est inférieur à EC_{opt}, alors (EC_{ls}) devient la consommation optimale EC_{opt}, le schéma ls devient le schéma logique optimal et nous passons au niveau hiérarchique suivant (lignes 13 - 15). Si (EC_{ls}) est supérieur à EC_{opt}, alors, on s'abstient de tester les décompositions restantes de cette dimension, et nous passons à la dimension suivante (ligne 17). À la fin, l'algorithme retourne le schéma logique optimal ainsi que sa consommation énergétique (ligne 18 & 19).

Algorithme 7 : LS-ENERGY

Entrées : Un schéma en étoile $SS = \{TF, D_1, D_2, ..., D_n\}$,
une charge de travail $Q = \{q_1, q_2, ..., q_m\}$;
Sorties : BLS : Le schéma logique le plus écologique,
EC_{opt} : la consommation énergétique du BLS ;

```
1  EC_opt = CM_energy(SS);
2  BLS_opt = SS;
3  Dimensions = les tables de dimension de (SS) sont triées en ordre décroissant;
4  pour chaque d ∈ Dimensions faire
5      TableDecomposer (les attributs hiérarchiques de d);
6      pour chaque décomposition dec faire
7          générer le schéma logique correspondant;
8          pour chaque schéma logique ls faire
9              QueryRewriter(ls);
10             éxécuter des requêtes;
11             évaluer la consommation d'énergie EC_ls = CM_energy(ls);
12             si EC_ls < EC_opt alors
13                 EC_opt = EC_ls;
14                 BLS_opt = ls;
15                 passer à la décomposition suivante;
16             sinon
17                 passer à la dimension suivante;
18 retourner BLS_opt : Le schéma logique optimal,
19 EC_opt : La consommation d'énergie optimale;
```

5.5 Étude expérimentale

Afin de montrer la pertinence et l'efficacité de notre approche, nous avons mené plusieurs expérimentations. Ainsi, nous avons effectué des tests sur un entrepôt de données implémenté avec le SGBD $PostgreSQL$ sur une machine de 8 Go de mémoire principale et un disque dur de 1 TB sous Ubuntu Server 14.04 LTS. Notre étude expérimentale a été réalisée sur le l'entrepôt de données issu du benchmark SSB [1] avec un facteur d'échelle de 10 Gb. SSB est composé d'une table de faits *Lineorder* et de 4 tables de dimensions (*Part, Customer, Supplier* et *Date*) comme illustré dans la figure 5.10. Selon la corrélation d'attributs, 256 schémas logique pourraient être générés.

1. https://www.cs.umb.edu/~poneil/StarSchemaB.PDF

FIGURE 5.10 – Le benchmark SSB

Nous avons également désactivé les tâches inutiles d'arrière-plan. De plus, le système et la mémoire cache sont vidés après chaque exécution de requête. Notre étude expérimentale est réalisée sur une machine équipée d'un wattmètre "Watts UP ? Pro ES" comme représenté sur la figure 5.11. Il est à noter que les requêtes sont exécutées de manière isolée. Grâce au modèle de coût, nous avons pu estimer le coût des E/S et le nombre de cycles CPU nécessaire pour exécuter la charge de travail. Nous avons également pu mesurer l'énergie consommée et la puissance moyenne de chaque requête.

Dans un premier temps, nous commençons par estimer le coût du schéma en étoile, puis, nous procédons à la décomposition de la table $Part$ (comme elle est la plus grande table de dimension) à travers ses attributs hiérarchiques (*p_brand*, *p_catgory* & *p_mfgr*). Pour la première décomposition, que nous noterons *P 1st*, nous décomposons la table $Part$ en mettant ses attributs hiérarchiques dans une table séparée et nous mesurons le coût nécessaire pour exécuter la charge de travail. Ensuite, nous comparons sa valeur avec celle du schéma en étoile.

Les résultats montrent que $P1st$ consomme moins d'énergie pour exécuter la charge de travail que le schéma en étoile. Ainsi, la solution optimale devient $P1st$ et nous passons à la deuxième décomposition que nous désignerons par $P2nd$. Nous répétons les même étapes nous comparons $P2nd$ à $P1st$. Les résultats obtenus ont été améliorés et entre-temps, l'énergie consommée a diminué comme le montre l'histogramme illustré dans la figure 5.12. Ainsi, $P2nd$ devient la solution optimale et nous procédons à la dernière décomposition de la table $Part$. Cette dernière devient complètement normalisée.

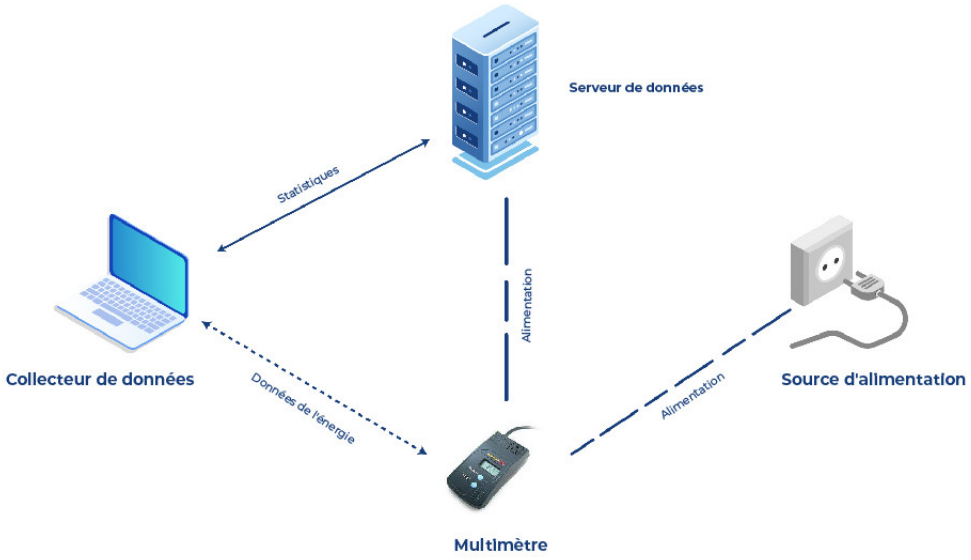


FIGURE 5.11 – L'environnement de travail

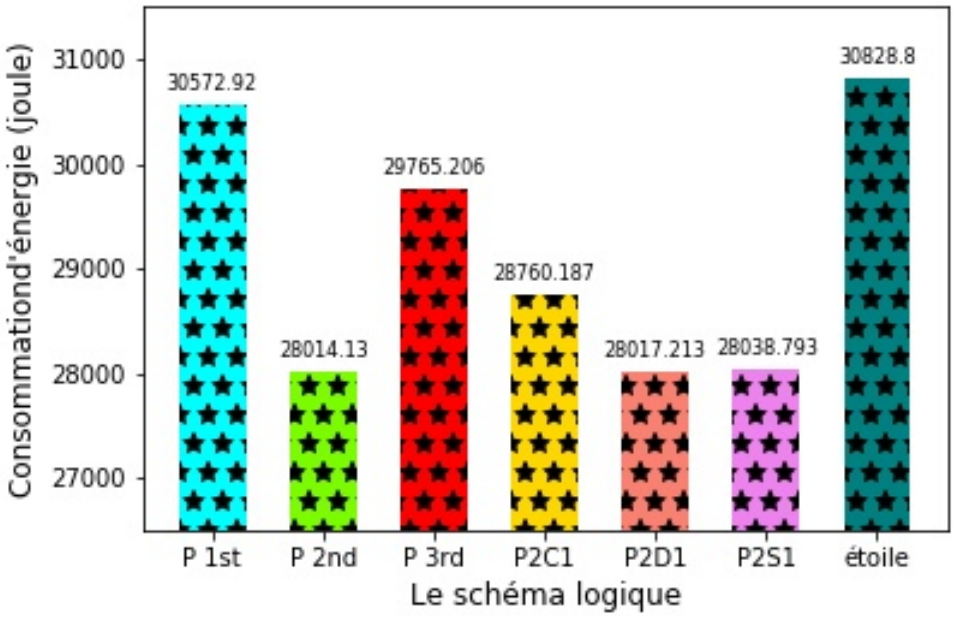


FIGURE 5.12 – La consommation énergétique de chaque schéma

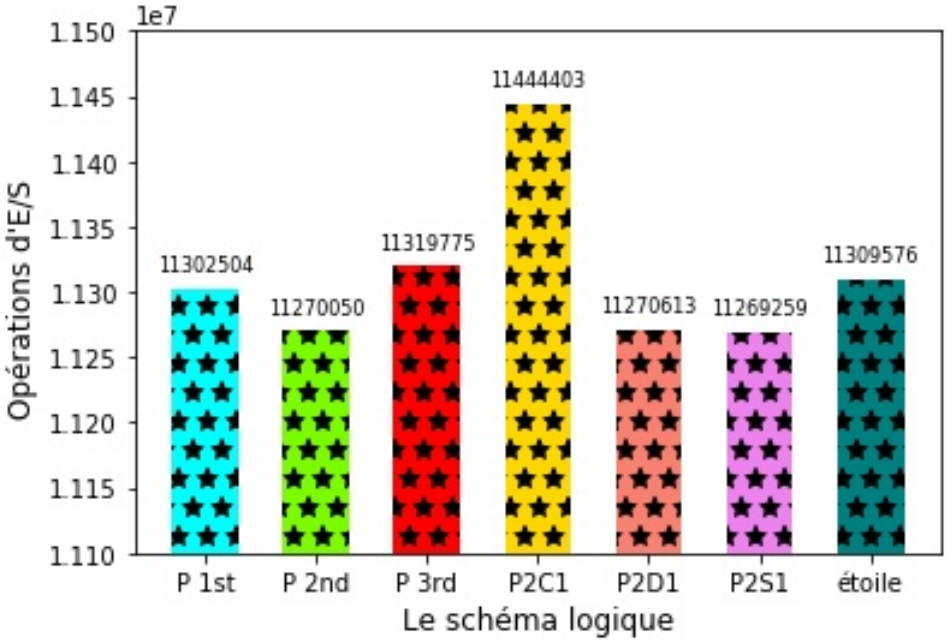


FIGURE 5.13 – Le coût d'exécution en opérations d'E/S

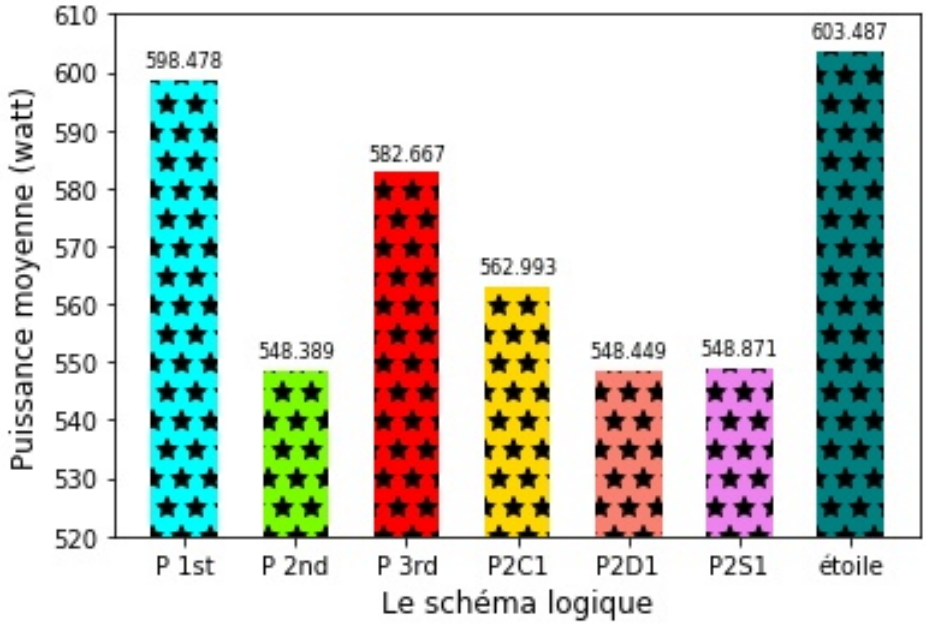


FIGURE 5.14 – La puissance moyenne consommée de chaque schéma

Cette fois, les valeurs ne se sont pas améliorées, par conséquent, $P2nd$ est maintenu comme la solution optimale. Comme la table $Part$ a été entièrement normalisée, nous passons à la table suivante qui est $Customer$. Cette dernière possède 3 attributs hiérarchiques (*c_city, c_nation* & *c_region*). Nous avons décomposé la table $Customer$ en mettant ses attributs hiérarchiques dans une autre table, tout en gardant la seconde décomposition de la table $Part$. Nous avons exécuté la charge de travail du nouveau schéma que nous notons $P2C1$ et nous la comparons avec le schéma optimal actuel, à savoir $P2nd$. Les résultats montrent que les valeurs ne se sont pas améliorées par rapport à la solution optimale mais néanmoins toujours meilleures que celles du schéma en étoile. Par conséquent, nous nous abstenons de générer les autres décompositions de la table $Customer$ et nous passons à la table suivante. La table $Supplier$ possède 3 attributs hiérarchiques (*s_city, s_nation* & *s_region*). Similaire à ce que nous avons fait avec la table $Customer$ nous la décomposons et nous comparons sa consommation d'énergie à celle de $P2nd$. Le schéma généré, noté $P2S1$, n'était pas meilleur que la solution optimale. En conséquence, nous passons à la dernière table, $Date$, qui a 3 attributs hiérarchiques (*d_dayofweek, d_month* & *d_year*). En exécutant les requêtes de la décomposition de la table $Date$ notée $P2D1$, la consommation d'énergie diminue par rapport aux autres décompositions, mais elle n'est pas meilleure que celle de $P2nd$. Ainsi, $P2nd$ est maintenu comme solution finale. Grâce à la contrainte d'anti-monotonie, l'espace de recherche est élagué en diminuant fortement le nombre de schémas "*testés*" de 256 à seulement 7 schémas.

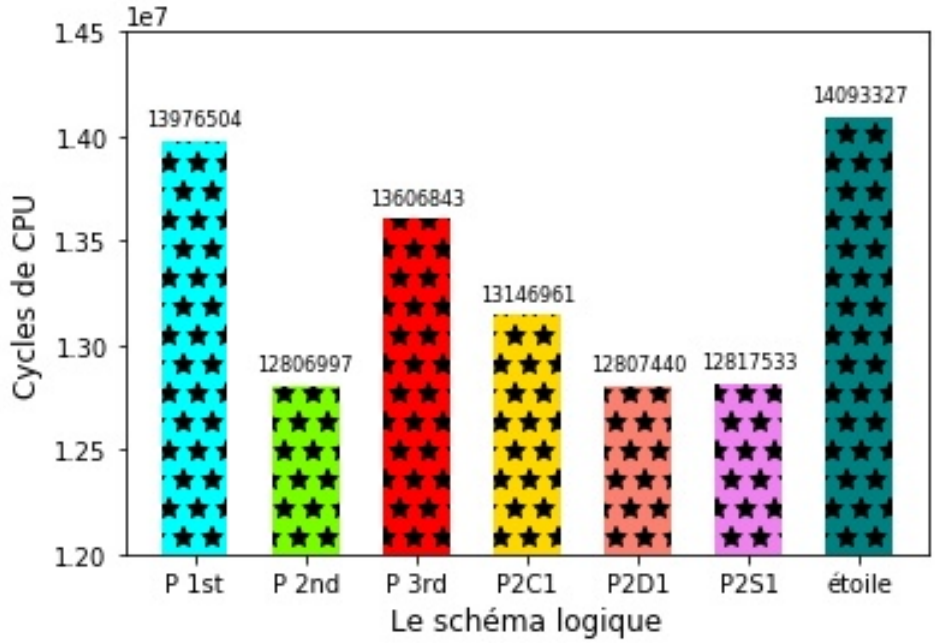


FIGURE 5.15 – Les cycles de CPU nécessaires pour exécuter la charge de travail

Le coût d'exécution en termes des opérations d'E/S, et la puissance moyenne consommée des différents schémas sont donnés respectivement dans les figures 5.13 - 5.14. Il est à noter que la taille du schéma change d'un schéma à l'autre. Selon l'histogramme de la figure 5.17, et comme déjà mentionné dans [23], la normalisation des grandes tables de dimension réduit l'espace de stockage et le schéma en étoile n'est pas le plus économique en termes d'espace de stockage.

Comme nous pouvons le voir dans la figure 5.12, le schéma en étoile est loin d'être le meilleur en termes de consommation d'énergie puisqu'il a besoin de plus de puissance comme représenté sur la figure 5.14 et il a besoin plus de cycles de CPU pour exécuter la charge

de travail comme le montre la figure. 5.15, tout en gardant en tête que le CPU est le majeur composant en consommation d'énergie. Ce phénomène est dû au fait que l'opération jointure des grandes tables nécessite beaucoup de ressources, ce qui engendre une augmentation de la puissance et par conséquent une augmentation de l'énergie totale consommée. Ainsi, notre approche proposée nous a permis d'économiser environ 10% d'énergie.

Néanmoins, le schéma en étoile peut être meilleur que d'autres schémas logiques pour le coût des E/S, comme nous pouvons le voir sur la figure 5.13. Bien que le schéma en étoile consomme plus d'énergie et a besoin de plus de puissance pour exécuter la charge de travail, il reste meilleur que la majorité des schémas logiques généré pour le temps d'exécution comme indiqué dans la figure 5.16.

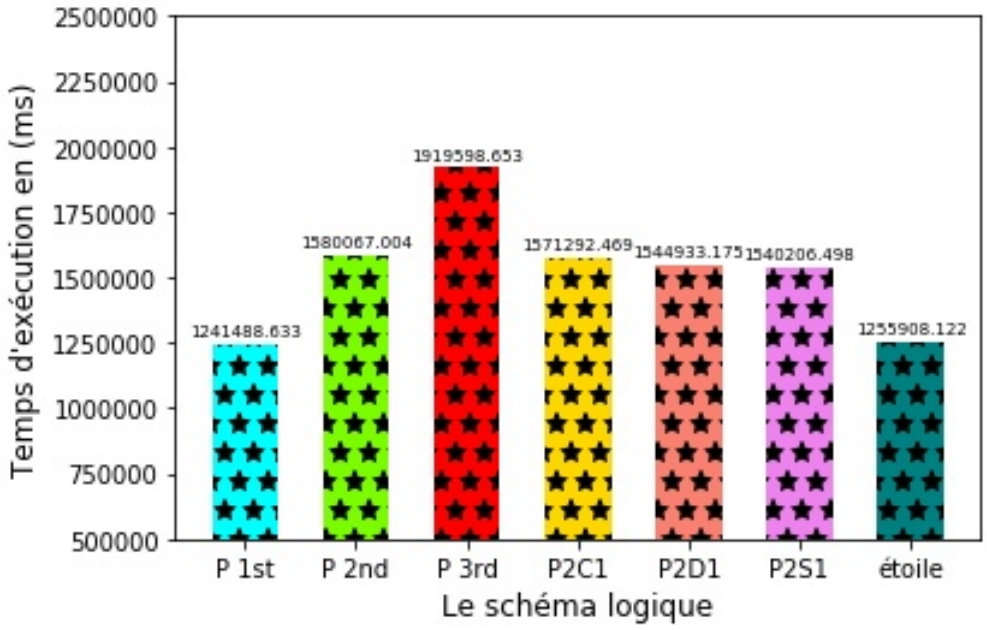


FIGURE 5.16 – Le temps d'exécution de la charge de travail de chaque schéma

Pour mieux comprendre ces résultats, nous avons observé le plan d'exécution des différentes requêtes des différents schémas. Au cours de cette observation, nous nous sommes concentrés sur l'ordre de jointure et les algorithmes de jointure utilisés. Ainsi, nous avons dévoilé que l'ordre de jointure des tables change lors du traitement des requêtes après chaque décomposition, et l'optimiseur de requêtes commence par joindre les petites tables contenant les prédicats de sélection. Ce faisant, génère un gain à la fois en opérations d'E/S et en coût CPU et par conséquent conduit à une baisse de la consommation d'énergie.

D'un autre côté, la modification de l'algorithme de jointure influence également la consommation d'énergie. Nous avons remarqué que l'*optimiseur de requêtes* alterne entre les deux algorithmes, boucles imbriquées et hash join. Cela peut s'expliquer par le fait que la jointure par boucles imbriquées est principalement adéquate pour les jointures avec un petit nombre de lignes. Le hash join est généralement utilisée lorsque le nombre de lignes de la table devient important.

Pour mieux comprendre l'effet des algorithmes de jointure, nous avons mené une étude expérimentale supplémentaire dans laquelle nous avons exécuté la charge de travail de notre benchmark SSB en forçant l'optimiseur de requête à utiliser un seul algorithme de jointure parmi les 3 algorithmes : Hash Join (HJ), Sort-Merge Join (SMJ) et Nested Loop Join NLJ). Les expériences mettent en évidence que la jointure par HJ était la meilleure en termes de temps d'exécution et également en termes de consommation d'énergie par rapport à SMJ et NLJ

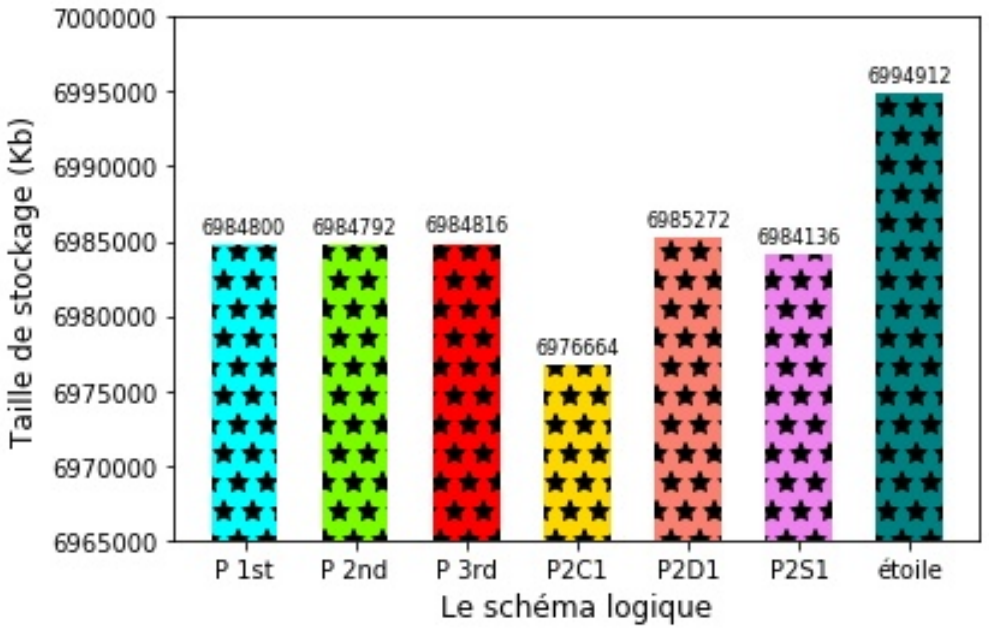


FIGURE 5.17 – La taille de stockage de chaque schéma

respectivement. Les résultats de la consommation d'énergie des algorithmes de jointure sont représentés sur la figure 5.18. Il convient de mentionner que certaines requêtes consomment moins d'énergie lorsqu'elles utilisent la jointure par les boucles imbriquées que par la jointure par hashage ou la jointure par le tri-fusion.

Selon Xu et al. la jointure par NLJ utilise 15 à 17 % de cycles de processeur supplémentaires. Cet algorithme de jointure est sélectionné uniquement lorsque la table est suffisamment petite pour être placée dans le tampon de la base de données [185]. Si la table est de grande taille et si sa table de hachage peut être chargée entièrement dans la mémoire, le coût de l'énergie pour accéder à la table de hachage est relativement faible par rapport au chargement de la table entière. Ainsi, il est plus probable que le choix de HJ soit fait dans les jointures de grandes tables que dans les jointures de tables plus petites.

D'autre part, les auteurs dans [78], ont étudié et comparé la consommation de différentes implémentations algorithmiques des opérateurs de tri et de jointure. Les expérimentations effectuées montrent que pour une table de petite taille, NLJ est le plus efficace, mais la complexité augmente rapidement à mesure que la taille augmente. En revanche, le nombre de cycles de CPU pour le HJ n'augmente que de manière presque linéaire, ce qui le rend cinq fois plus rapide. L'algorithme SMJ a deux facettes : pour les données pré-triées, il s'agit de loin l'algorithme le plus rapide des trois, quelle que soit la taille d'entrée. Par contre, si les données doivent d'abord être triées, les performances sont fortement ralenties par l'algorithme de tri. Par conséquent, HJ semble être l'algorithme de jointure le plus efficace pour les données d'entrée non triées. D'autre part, NLJ et SMJ n'ont presque pas besoin d'espace mémoire supplémentaire, cependant, HJ utilise une quantité considérable de mémoire pour stocker la table de hachage. Par conséquent, nous pouvons dire que les algorithmes de jointure et l'ordre de jointure ont une influence majeure sur la consommation d'énergie.

Comme le $SGBD$ que nous avons utilisé est open source, nous avons pensé à en tirer profit et à étudier l'impact de l'étape de planification sur la consommation d'énergie. La tâche de planification consiste à créer un plan d'exécution optimal. Pour une requête SQL, il existe une multitude de manières pour l'exécuter dont chacune produira le même résultat à la fin de l'exécution de la requête. L'optimiseur de requête estime le coût de l'exécution de chaque plan

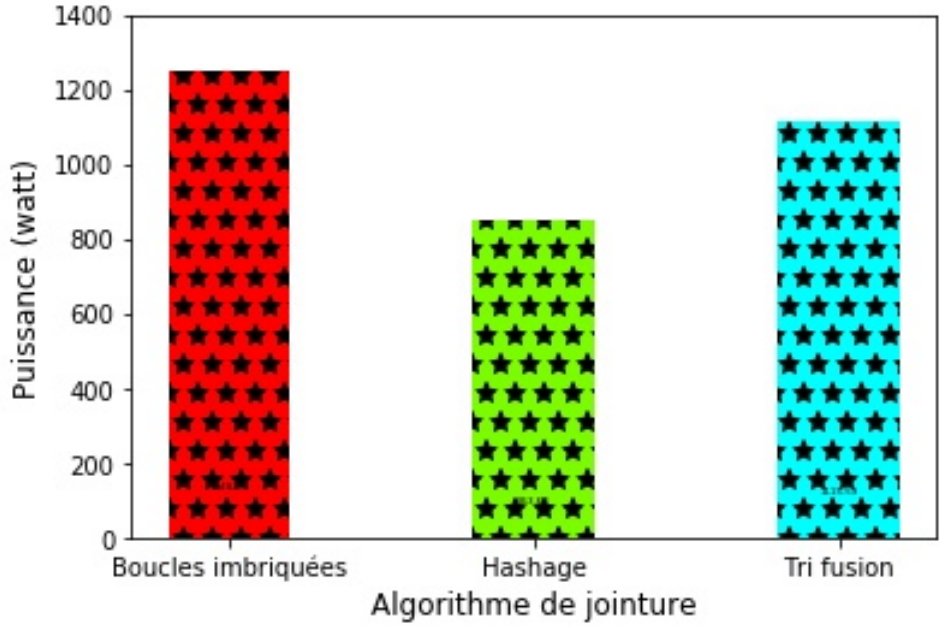


FIGURE 5.18 – La consommation énergétique des algorithmes de jointures

TABLEAU 5.1 – Les stratégies du choix de la séquence de jointures

	Recherche Exhaustive	Algorithme génétique
Temps de Planifications (ms)	11, 07	16, 99
Temps d'exécution (s)	24, 09	29, 17
Opérations d'E/S	2897242	3039436
Puissance moyenne (W)	48.40	47.24
Energie (J)	1210.06	1417.38

en utilisant son modèle de coût et cherche à trouver celui qui minimise le temps d'exécution. Le planificateur commence par la génération des plans d'exécution pour la lecture de chaque table de la requête d'une manière indépendante. La lecture d'une table s'effectue de deux manières, soit par un accès séquentiel soit par un accès indexé (en utilisant les indexes définis), ainsi, un plan d'accès séquentiel est toujours créé. Les autres plans possibles sont déterminés par les indexes disponibles sur chaque table. Dans le cas où une requête impliquerait une ou plusieurs opérations de jointures, ainsi, tous les plans d'exécution possibles pour effectuer ces jointures sont générés. Comme nous l'avons mentionné précédemment, PostgreSQL admet trois stratégies de jointure qui sont : la jointure par hachage, la jointure par boucle imbriquée, et la jointure par fusion. Lorsque une requête contient plus qu'une opération de jointure, le résultat final doit être construit par un arbre de jointures. Le planificateur analyse les différentes séquences de jointure possibles pour choisir la séquence optimale en termes de coût. Pour explorer l'espace de recherche et choisir la meilleure séquence de jointure, PostgreSQL utilise deux algorithmes. Un algorithme basé sur une recherche quasi-exhaustive, et ce dans le cas où le nombre de tables serait inférieur à un certain seuil défini (fixé à 12 relations dans PostgreSQL). Dans le cas où le seuil serait dépassé, un algorithme génétique est utilisé (GEQO). Le paramètre permettant de modifier le seuil de nombre de tables pour l'utilisation de l'algorithme génétique est *geqo_threshold*.

Pour étudier l'influence de ces algorithmes de recherche, nous considérons la requête *Q7* du benchmark TPC-H donnée ci-dessous. Nous avons changé le benchmark pour avoir plus

de tables. Dans notre expérience, nous modifions le planificateur de PostgreSQL pour générer des scénarios différents : (*i*) la recherche d'un plan d'exécution en utilisant l'algorithme de la recherche quasi-exhaustive (*ii*) en utilisant l'algorithme génétique. Pour forcer l'exécution de l'algorithme génétique pour le choix du plan d'exécution, il suffit de modifier la valeur du paramètre *geqo_threshold* et de la mettre à 2. Pour chaque stratégie, nous calculons le temps de planification en utilisant "*Explain Plan*", le temps d'exécution, le coût estimé d'opérations d'E/S, la puissance et la consommation d'énergie lors de l'exécution de la requête. Les résultats de la requête *Q7* sont présentés dans le tableau 5.1.

```
SELECT
  supp_nation,
  cust_nation,
  l_year,
  SUM(volume) AS revenue
FROM
  (
SELECT
        n1.n_name AS supp_nation,
        n2.n_name AS cust_nation,
        YEAR(l_shipdate) AS l_year,
        l_extendedprice * (1 - l_discount) AS volume
FROM
        supplier,
        lineitem,
        orders,
        customer,
        nation n1,
        nation n2
WHERE
        s_suppkey = l_suppkey
        AND o_orderkey = l_orderkey
        AND c_custkey = o_custkey
        AND s_nationkey = n1.n_nationkey
        AND c_nationkey = n2.n_nationkey
        AND (
        (n1.n_name = 'FRANCE' AND n2.n_name = 'GERMANY')
        OR (n1.n_name = 'GERMANY' AND n2.n_name = 'FRANCE')
             )
        AND l_shipdate BETWEEN MDY(1,1,1995) AND MDY(12,31,1996)
    ) AS shipping
GROUP BY
    supp_nation,
    cust_nation,
    l_year
ORDER BY
    supp_nation,
    cust_nation,
    l_year
```

Pour presque la totalité des requêtes, les deux algorithmes arrivent à choisir le même plan d'exécution où le temps d'exécution de ces deux algorithmes afin de trouver le plan d'exécution optimal est de l'ordre de quelques millisecondes, malgré que l'algorithme de la recherche exhaustive s'exécute plus rapidement que celui de l'algorithme génétique. Dans les rares cas où les deux algorithmes ne retournent pas le même plan d'exécution, la différence entre les plans générés sont au niveau de l'algorithme de jointure utilisé et au niveau de l'ordre de jointure.

Pour mieux étudier les différences entre les deux algorithmes, nous avons utilisé un benchmark plus complexes que TPC-H. Le benchmark TPC-DS compte 24 tables et contient des requêtes beaucoup plus complexes que ceux du *SSB* ou bien TPC-H. Néanmoins, même en variant le benchmark, le temps de génération des plans d'exécution reste de l'ordre de millisecondes. Dans cette expérimentation, nous avons constaté, qu'il y a des requêtes où l'algorithme génétique génère un plan d'exécution meilleur que celui de la recherche exhaustive. Et si nous forçons l'utilisation d'un seul algorithme de jointure, les deux approches de planification donnent le même résultat. Nous avons également trouvé pour quelques requêtes, que le coût d'E/S estimé par la recherche quasi-exhaustive est inférieur à celui de l'algorithme génétique, mais le temps d'exécution et l'énergie consommée du plan généré par l'algorithme génétique sont largement meilleurs. La durée d'exécution très réduite des deux algorithmes et leurs résultats très proches nous éloignent de l'idée des plans d'exécution à la carte, dans lesquels, les plans d'exécutions sont préparés à l'avance.

5.6 Conclusion

Dans ce chapitre, nous avons mis en exergue l'importance de la phase de la conception logique à travers l'évaluation de l'impact de sa variabilité sur les économies d'énergie. Nous avons décrit notre méthodologie de sélection éco-énergétique d'un schéma logique. Notre proposition s'articule autour de deux notions : la variablité et la contrainte d'anti-monotonie. Ce chapitre permet en premier lieu de décrire comment gérer la variabilité au niveau de la conception logique, en décrivant comment bénéficier des corrélations existantes entre les différents attributs des tables de dimension afin de générer les différents schémas logiques. Ensuite, nous nous penchons sur la contrainte d'anti-monotonie pour élaguer l'espace de recherche.

À la suite de l'implémentation des algorithmes nécessaires pour la mise en œuvre de notre approche, une étude expérimentale a été menée pour prouver l'efficience de cette dernière. Les résultats obtenus montrent que la variation du schéma logique a un impact direct sur la consommation d'énergie. Les expérimentations révèlent que la technique d'élagage adoptée par notre approche nous a évité la génération inutile de plus que 90% des schémas possibles. En outre, nous avons montré la grande importance des algorithmes de jointure et leur rôle fondamental pour la réduction de l'énergie. De plus, cette étude expérimentale nous incite à rompre avec les pratiques traditionnelles d'amélioration des performances telle que la dénormalisation en montrant que la normalisation des tables de faibles tailles a permis de réduire la consommation énergétique et l'espace de stockage occupé par les tables dé-normalisées.

Quatrième partie

Conclusion et Perspectives

Chapitre 6

Conclusion générale et perspectives

6.1 Conclusion générale

Ce chapitre présente le bilan du travail que nous avons effectué durant cette thèse et propose un ensemble d'ouvertures et de perspectives permettant d'améliorer le travail réalisé.

À l'heure actuelle, la demande d'énergie et l'exploitation excessive des sources d'énergie, qui sont parfois très peu écologiques, ne cessent pas de s'accroître en raison de la nécessité vitale de l'énergie pour notre vie quotidienne. Cependant, la consommation énergétique a des répercussions néfastes sur l'environnement à cause du gaz à effet de serre, tel que le CO_2, qui est lié directement au réchauffement climatique que notre planète terre est en train de subir. Avec l'émergence d'Internet et des objets connectés et l'informatisation d'une grande majorité des services, nos habitudes de consommation ont changé et nous sommes devenues de plus en plus dépendants des systèmes informatiques qui nécessitent une forte consommation d'électricité pour leur fonctionnement permanent quel que soit pour un simple téléphone portable, un ordinateur personnel ou bien un serveur hébergé dans le Cloud. En effet, les consommateurs et les producteurs d'infrastructures de stockage et de traitement des données sont au cœur du débat sur l'informatique verte à cause des quantités colossales de données générées quotidiennement par les utilisateurs, qui doivent être stockées et traitées. En effet, les gestionnaires de nos secteurs de vie sont de plus en plus nombreux à utiliser des approches basées sur les données pour les stratégies de prise de décision qui concernent notre santé, notre planète, notre bien-être, notre éducation, notre alimentation, etc., ce qui a placé les bases de données et les systèmes de gestion de bases de données au cœur des centres de données. Cet emplacement a conduit les bases de données à devenir les majeurs responsables de la surconsommation d'énergie dans les centres de données.

À travers ce travail, nous tentons à tirer la sonnette d'alarme et de mobiliser les concepteurs et les utilisateurs des systèmes informatique en général et la communauté des bases de données en particulier pour qu'ils se prononcent sur l'efficacité énergétique dans les travaux de recherche et les futurs produits. Dans cette thèse, nous nous sommes focalisés sur l'efficacité énergétique dans les bases de données, plus précisément, dans l'intégration de l'énergie dans ces différentes étapes de son cycle de vie, en la considérant en tant que besoin non-fonctionnel additionnel avec la performance, qui reste toujours la marque de fabrique de ce domaine. La grande variété des BD, nous a permis d'identifier plusieurs points d'intégration de l'aspect énergétique. Nous avons abordé ce problème à travers une méthodologie qui se base sur l'étude du comportement énergétique des différentes étapes du cycle de vie de la BD à travers la réalisation des expérimentations. Ces dernières nous ont permis d'identifier les paramètres sensibles à l'énergie et de nous donner des pistes. Ce qui nous a conduit à proposer des approches qui permettent de réduire à la fois l'énergie consommée sans dégrader la performance. Ce travail, nous a permis de couvrir plusieurs branches de la BD, telle que la conception logique, la conception physique, les structures d'optimisation, l'énergie, les entrepôts de données (grâce aux benchmarks utilisés) ; etc. À travers l'étude que nous avons réalisé, nous sommes sûrs que les chercheurs de notre communauté trouveront de grandes opportunités pour intégrer la dimension énergétique dans leur recherche et leur enseignement.

Pour réaliser notre objectif et réduire l'énergie dans le contexte des bases de données, notre travail s'articule autour de 4 points essentiels : (i) l'état de l'art, (ii) le compromis entre l'énergie et la performance (iii) la sélection éco-énergétique des indexes de jointure binaires (iv) la sélection éco-énergétique du schéma logique.

6.1.1 État de l'art

Dans cette thèse, le travail réalisé était guidé, au premier lieu, par une étude des travaux existants afin de maîtriser le domaine, bien se positionner, identifier les points d'intégration

de l'énergie et les paramètres sensibles à l'énergie, et de détecter les pistes prometteuses et très peu explorées dans la littérature. Nous avons divisé cette étude de l'état de l'art en deux parties pour mieux expliquer les deux piliers de notre thèse. La première partie est consacrée à l'étude de la variété des bases de données et leur cycle de vie en mettant en exergue les techniques d'optimisations et plus précisément les structures d'optimisation dans la phase de conception physique. La deuxième partie se focalise sur le traitement de l'énergie dans les systèmes informatiques en général, et dans les bases de données en particulier. Nous avons présenté les notions de base relatives à l'énergie et les différentes méthodes et outils pour l'estimation et la mesure de l'énergie. Nous avons, également, présenté une classification des approches proposées et nous avons proposé un bilan dans lequel nous avons résumé les travaux existants en identifiant les lacunes afin de les améliorer dans notre travail.

6.1.2 Compromis entre l'énergie et la performance

La performance a été, et reste toujours la marque de fabrique des bases de données, ainsi, incorporer la dimension énergétique qui risque d'avoir des répercussions néfastes et dégrader les performances présente un vrai défit. Pour éviter ce risque, il est primordial de trouver un compromis entre l'amélioration de l'efficacité énergétique et la dégradation de la performance. Ce compromis, doit basculer le problème d'optimisation classique, mono-objectif, dans lequel la solution est généralement unique, en un problème d'optimisation multi-objectif, où il existe un ensemble de solutions candidates. L'étude de l'état de l'art a montré que la plupart des approches optent pour des solutions mono-objectif lors du traitement des problèmes d'optimisation malgré la présence de différentes techniques permettant la résolution multi-objectif tel que la méthode de pondération, les algorithmes évolutionnaires comme les algorithmes génétiques et les approches basées sur la dominance. Ainsi, nous tirons profit de ces techniques afin de proposer des approches multi-objectif permettant de réduire l'énergie sans trop dégrader la performance.

6.1.3 La sélection éco-énergétique des Indexes de jointure Binaires

Pour notre première contribution, nous avons démontré notre sensibilité à l'énergie en considérant un problème traditionnel dans les bases de données/entrepôts de données qui traite la sélection des indexes. Nous avons proposé une approche permettant d'intégrer la dimension énergétique dans la phase de conception physique du cycle de vie des bases de données, en considérant le cas des indexes de jointure binaires qui présentent une structure d'optimisation redondante. Notre choix de ce type d'index est motivé pour la capacité des IJBs d'optimiser, à la fois, la jointure et la sélection. Au premier lieu, nous avons proposé une nouvelle formalisation du problème de sélection des IJBs en intégrant la dimension énergétique. Notre méthodologie est basée sur un audit réalisé sur ce type d'index à travers une série d'expérimentations, afin d'identifier les paramètres à prendre en considération, les critères d'élagage et les conditions favorables pour l'utilisation de cet index. Ces expérimentations nous ont permis de se rendre compte de l'importance du facteur de Fan-Out lors de la décision de choisir l'utilisation d'un index ou pas. Dans notre approche, nous avons considéré l'énergie comme un besoin non-fonctionnel additionnel avec la performance. Pour trouver le meilleur compromis entre les différents BnFs, nous avons utilisé l'opérateur de Skyline pour une sélection multi-objectif, et la valeur de Fan-Out comme un critère d'élagage. Nous avons également modélisé la charge de travail à travers un hypergraphe. Les résultats de l'étude expérimentale menée, montre l'efficacité et la pertinence de notre approche, grâce à la réduction de l'énergie consommée lors de l'exécution la charge de travail du benchmark utilisé tout en diminuant le temps d'exécution. Les résultats ont montré aussi que l'énergie et la performance ne sont pas toujours corrélées, et que l'utilisation des structures d'optimisation redondantes améliorent l'efficacité énergétique, mais peuvent engendrer une augmentation de la puissance maximale.

6.1.4 La sélection éco-énergétique du schéma logique

L'étude de l'état de l'art nous a permis de découvrir un manque patent des travaux consacrés à la phase de la conception logique qui est souvent ignorée lors de la résolution des problèmes d'optimisation. Pour notre deuxième contribution, nous avons intégré l'énergie à un niveau supérieur, plus précisément, dans l'étape de conception logique à travers une approche de génération de schémas logique à partir d'un schéma initial en étoile en se basant sur les attributs hiérarchiques des tables. Cette approche utilise un modèle de coût énergétique qui permet d'estimer l'énergie consommée par les requêtes sur un schéma logique. Pour élaguer l'espace de recherche, qui peut s'exploser en fonction des attributs hiérarchiques des tables, nous avons utilisé la contrainte d'anti-monotonie afin de réduire le nombre de schémas générés. L'étude expérimentale réalisée a prouvé l'efficacité de notre stratégie d'élagage et a montré l'impact de la variabilité du schéma logique sur la réduction de l'énergie de la base de données. L'interprétation des résultat obtenus a montré aussi que la dé-normalisation des tables peut avoir des effets contraires et augmenter l'énergie et le temps d'exécution, et que le choix de l'algorithme de jointure peut influencer sur la consommation de l'énergie.

6.2 Perspectives

Les travaux initiés dans cette thèse laissent envisager de nombreuses perspectives. Dans cette section, nous esquissons quelques perspectives qui nous paraissent être les plus importantes, et qui pourront faire objet de futurs travaux.

6.2.1 Étudier l'énergie lors du design conceptuel

Le travail réalisé au cours de cette thèse a consisté principalement à la considération d'un schéma d'une base de données initialement conçu, ensuite, nous avons varié le schéma logique et nous avons montré que cette variabilité nous a permis d'améliorer la consommation l'énergie de la base de données. De même pour la phase physique, le bon choix de la structure d'optimisation et le choix des attributs adéquats a réduit considérablement l'énergie consommée. Ainsi, nous proposons d'améliorer l'efficacité énergétique à un niveau supérieur à celui de la conception logique, à une étape antérieure lors du cycle de vie de la base de données. En effet, nous visons à étudier l'énergie dès le design conceptuel, en modélisant l'énergie a ce stade ce qui permet de faciliter l'évolution de la BD, d'anticiper et d'éviter les pratiques qui risquent d'augmenter l'énergie lors du déploiement de la BD.

6.2.2 Définir un modèle de coût pour les IJB

Malgré les résultats satisfaisants que nous avons obtenu, à travers notre approche énergétique de sélection des indexes de jointure binaires grâce à l'efficacité de la phase d'élagage qui se base sur le Fan-Out, hormis, cette heuristique était fondée principalement sur l'intuition et l'expertise humaine. Cependant, un modèle de coût énergétique est un outil primordial permettant de prédire et d'estimer la consommation énergétique des requêtes en présence des indexes avant leur création. Afin de développer un modèle de coût précis et pertinent, il est essentiel d'identifier d'autres paramètres sensibles à l'énergie lors de l'exécution d'une requête en présence de l'IJB. Ainsi, et en se basant sur les paramètres que nous avons pu identifier dans cette thèse, nous proposons d'effectuer plus d'expérimentations et d'auditer encore les IJB en prenant en considération le matériel utilisé afin de proposer un modèle de coût énergétique capable de prédire l'énergie consommée par les requêtes en présence des IJB.

6.2.3 Étudier d'autres structures d'optimisation

Dans le premier chapitre, nous avons présenté les techniques d'optimisation et une variété de structures d'optimisation permettant d'améliorer la performance. Ces SO, sont fiables lorsqu'elles sont utilisées convenablement et d'une manière adéquate. Dans cette thèse, nous avons étudié le cas des IJB, qui ne sont implémentés que sur quelques $SGBD$, contrairement à la fragmentation horizontale, par exemple, qui est proposée par la majorité écrasante des SGBD et elle joue un rôle très important dans les bases de données NoSQL. Ainsi, nous proposons d'étudier le comportement énergétique de la FH et d'identifier les paramètres sensibles à l'énergie de cette structure d'optimisation, en pratiquant la même stratégie que nous avons utilisé avec les IJB, à travers un audit qui nous permettra de mieux comprendre et d'avoir une idée sur les paramètres à prendre en considération lors du développement d'une approche énergétique de sélection d'un schéma de fragmentation. Nous visons, également, à proposer une approche éco-énergétique d'une sélection multiple des structures d'optimisation dans laquelle nous combinons la sélection de plusieurs SO (IJB et FH par exemple), afin d'améliorer l'efficacité énergétique et la performance de la base de données.

6.2.4 La sélection conjointe des structures d'optimisation et du schéma logique

Les approches que nous avons présentées dans la partie contribution ont été réalisées d'une manière isolée, bien qu'elles sont fortement liées. Ainsi, nous proposons d'introduire une approche énergétique d'une sélection conjointe du schéma logique et d'une structure d'optimisation, de telle sorte, nous choisissons le meilleur schéma logique au premier lieu et ensuite nous essayons de l'améliorer encore à travers le choix des structures d'optimisation correspondantes au schéma choisi. Ainsi, nous proposons un outil comme un advisor qui, à partir d'un schéma de base de données initial, permet de générer le schéma optimal et propose les structures d'optimisation adéquates pour avoir la base de données moins énergivore sans dégrader la performance.

6.2.5 Étudier l'efficacité énergétique des bases de données NoSQL

Le succès des bases de données relationnelles depuis leur apparition et la dominance des SGBD relationnels sur le marché, ont encouragé les chercheurs à se focaliser généralement sur ce modèle de données lorsqu'ils abordent l'aspect énergétique. Sauf que, depuis un peu plus d'une décennie, et avec l'avènement de l'internet et surtout avec la quantité colossale et la variété du type et de la structure des données générées, les SGBD NoSQL ont commencé à connaître leur succès auprès des utilisateurs soutenus par les grandes entreprises technologiques qui ont développé leurs propres SGBD comme Apache Hbase qui est conçue pour la base de données BigTable par Google, DynamoDB par Amazon, Kylin par eBay etc. Ainsi, il nous semble très prometteur d'étudier le comportement énergétique de ces familles de SGBD, en raison de leur popularité qui ne cesse pas de s'accroître, la quantité de données utilisées, et le fait que la majorité de ces SGBD sont open source ce qui facilite l'intégration de l'énergie. Nous proposons de projeter notre approche de sélection des structures d'optimisation sur un SGBD NoSQL comme MongoDB par exemple, afin d'identifier les paramètres sensibles à l'énergie dans un tel SGBD et pouvoir améliorer son efficacité énergétique.

Annexe A

Annexe A

Nous présentons la charge de travail utilisée pour l'expérimentation de notre stratégie de sélection des IJBs sur le benchmark SSB. Nous utilisons les 13 requêtes fournies par le banc de tests, elles divisées en 4 catégories et chaque catégorie est composée de requêtes complexes qui comportent plusieurs opérations de sélection et de jointure en augmentant la complexité des requêtes de catégorie à autre.

```
-- Q1.1
SELECT sum(v_revenue) as revenue
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
WHERE d_year = 1993
AND lo_discount between 1 AND 3
AND lo_quantity < 25;

-- Q1.2
SELECT sum(v_revenue) as revenue
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
WHERE d_yearmonthnum = 199401
AND lo_discount between 4 AND 6
AND lo_quantity between 26 AND 35;

-- Q1.3
SELECT sum(v_revenue) as revenue
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
WHERE d_weeknuminyear = 6 AND d_year = 1994
AND lo_discount between 5 AND 7
AND lo_quantity between 26 AND 35;

-- Q2.1
SELECT sum(lo_revenue) as lo_revenue, d_year, p_brand
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN part on lo_partkey = p_partkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
WHERE p_category = 'MFGR#12' AND s_region = 'AMERICA'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;

-- Q2.2
SELECT sum(lo_revenue) as lo_revenue, d_year, p_brand
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN part on lo_partkey = p_partkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
WHERE p_brand between 'MFGR#2221' AND 'MFGR#2228' AND s_region = 'ASIA'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;
```

```
-- Q2.3
SELECT sum(lo_revenue) as lo_revenue, d_year, p_brand
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN part on lo_partkey = p_partkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
WHERE p_brand = 'MFGR#2239' AND s_region = 'EUROPE'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;


-- Q3.1
SELECT c_nation, s_nation, d_year, sum(lo_revenue) as lo_revenue
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN customer on lo_custkey = c_custkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
WHERE c_region = 'ASIA' AND s_region = 'ASIA'AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_nation, s_nation, d_year
ORDER BY d_year asc, lo_revenue desc;

-- Q3.2
SELECT c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN customer on lo_custkey = c_custkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
WHERE c_nation = 'UNITED␣STATES' AND s_nation = 'UNITED␣STATES'
AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, lo_revenue desc;

-- Q3.3
SELECT c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN customer on lo_custkey = c_custkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
WHERE (c_city='UNITED␣KI1' or c_city='UNITED␣KI5')
AND (s_city='UNITED␣KI1' or s_city='UNITED␣KI5')
AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, lo_revenue desc;

-- Q3.4
SELECT c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN customer on lo_custkey = c_custkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
WHERE (c_city='UNITED␣KI1' or c_city='UNITED␣KI5')
AND (s_city='UNITED␣KI1' or s_city='UNITED␣KI5') AND d_yearmonth = 'Dec1997'
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, lo_revenue desc;

-- Q4.1
SELECT d_year, c_nation, sum(lo_revenue) - sum(lo_supplycost) as profit
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN customer on lo_custkey = c_custkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
LEFT JOIN part on lo_partkey = p_partkey
WHERE c_region = 'AMERICA' AND s_region = 'AMERICA'
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, c_nation
ORDER BY d_year, c_nation;

-- Q4.2
SELECT d_year, s_nation, p_category, sum(lo_revenue) - sum(lo_supplycost) as profit
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN customer on lo_custkey = c_custkey
```

```
LEFT JOIN supplier on lo_suppkey = s_suppkey
LEFT JOIN part on lo_partkey = p_partkey
WHERE c_region = 'AMERICA'AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998)
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category
ORDER BY d_year, s_nation, p_category;



-- Q4.3
SELECT d_year, s_city, p_brand, sum(lo_revenue) - sum(lo_supplycost) as profit
FROM p_lineorder
LEFT JOIN dates on lo_orderdate = d_datekey
LEFT JOIN customer on lo_custkey = c_custkey
LEFT JOIN supplier on lo_suppkey = s_suppkey
LEFT JOIN part on lo_partkey = p_partkey
WHERE c_region = 'AMERICA'AND s_nation = 'UNITED␣STATES'
AND (d_year = 1997 or d_year = 1998)
AND p_category = 'MFGR#14'
GROUP BY d_year, s_city, p_brand
ORDER BY d_year, s_city, p_brand;
```

Annexe B

Annexe B

Nous présentons la charge de travail du benchmark *SSB* utilisée durant l'étude expérimentale de notre stratégie de sélection du schéma logique le plus écologique. Elles sont 30 requêtes de plusieurs types ont été considérées : requêtes ayant plusieurs attributs appartenant à des tables de dimension dans la clause "SELECT", des requêtes de type *count(*)*, avec et sans agrégation, des requêtes utilisant les fonctions d'agrégation comme *Sum*, *Min*, *Max*, etc.

```
-- Q1
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, dates
WHERE lo_orderdate = d_datekey AND d_year = 1993
AND lo_discount >= 1 AND lo_discount <= 3 AND lo_quantity < 25

-- Q2
SELECT count(*)
FROM lineorder, dates
WHERE lo_orderdate = d_datekey AND d_year = 1993
AND lo_discount >= 1 AND lo_discount <= 3 AND lo_quantity < 25

-- Q3
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, dates
WHERE lo_orderdate = d_datekey AND d_year = 1993

-- Q4
SELECT count(*)
FROM lineorder, dates
WHERE lo_orderdate = d_datekey AND d_year = 1993


-- Q5
SELECT sum(lo_revenue), d_year
FROM lineorder, dates, part, supplier
WHERE lo_orderdate = d_datekey AND lo_partkey = p_partkey
AND lo_suppkey = s_suppkey AND p_brAND = 'MFGR#2221' AND s_region = 'ASIA'
GROUP BY d_year ORDER BY d_year


-- Q6
SELECT sum(lo_revenue)
FROM lineorder, part
WHERE lo_partkey = p_partkey AND p_brAND = 'MFGR#2221'


-- Q7
SELECT avg(lo_revenue), d_year
FROM lineorder, dates, part, supplier
WHERE lo_orderdate = d_datekey AND lo_partkey = p_partkey
AND lo_suppkey = s_suppkey AND p_brAND = 'MFGR#2221' AND s_region = 'ASIA'
GROUP BY d_year ORDER BY d_year
```

```
-- Q8
SELECT sum(lo_revenue), d_year
FROM lineorder, dates, part, supplier
WHERE lo_orderdate = d_datekey AND lo_partkey = p_partkey
AND lo_suppkey = s_suppkey AND p_brAND = 'MFGR#2221' AND s_region = 'EUROPE'
GROUP BY d_year, p_brAND ORDER BY d_year, p_brAND


-- Q9
SELECT sum(lo_revenue)
FROM lineorder, part, supplier
WHERE lo_partkey = p_partkey AND lo_suppkey = s_suppkey
AND p_brAND = 'MFGR#2221' AND s_region = 'EUROPE'

-- Q10
SELECT count(*), d_year
FROM lineorder, dates, part, supplier
WHERE lo_orderdate = d_datekey AND lo_partkey = p_partkey
AND lo_suppkey = s_suppkey AND p_brAND = 'MFGR#2221' AND s_region = 'EUROPE'
GROUP BY d_year ORDER BY d_year

-- Q11
SELECT c_nation, s_nation, d_year, sum(lo_revenue) as revenue
FROM lineorder,customer,  supplier, dates
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_orderdate = d_datekey
AND c_region = 'ASIA' AND s_region = 'ASIA' AND d_year >= 1992  AND d_year <= 1997
GROUP BY c_nation, s_nation, d_year ORDER BY d_year asc, revenue desc

-- Q12
SELECT s_nation, sum(lo_revenue) as revenue
FROM lineorder, supplier
WHERE lo_suppkey = s_suppkey AND s_region = 'ASIA'
GROUP BY s_nation ORDER BY revenue desc


-- Q13
SELECT s_nation, count(*) as revenue
FROM lineorder, supplier
WHERE lo_suppkey = s_suppkey AND s_region = 'ASIA'
GROUP BY s_nation ORDER BY revenue desc

-- Q14
SELECT s_nation, d_year, sum(lo_revenue) as revenue
FROM lineorder, supplier, dates
WHERE lo_suppkey = s_suppkey AND lo_orderdate = d_datekey AND s_region = 'ASIA'
AND d_year >= 1992 AND d_year <= 1997
GROUP BY s_nation, d_year ORDER BY d_year asc, revenue desc


-- Q15
SELECT c_nation, s_nation, d_year, avg(lo_revenue) as avg_revenue
FROM lineorder,customer,  supplier, dates
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_orderdate = d_datekey
AND c_region = 'ASIA' AND s_region = 'ASIA' AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_nation, s_nation, d_year ORDER BY d_year asc, avg_revenue desc

-- Q16
SELECT c_nation, s_nation, d_year, count(*)
FROM lineorder,customer,  supplier, dates
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_orderdate = d_datekey
AND c_region = 'ASIA' AND s_region = 'ASIA' AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_nation, s_nation, d_year ORDER BY d_year asc


-- Q17
SELECT c_city, s_city, d_year, sum(lo_revenue) as revenue
FROM lineorder,customer,  supplier, dates
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_orderdate = d_datekey
```

```
AND c_nation = 'UNITED␣STATES' AND s_nation = 'UNITED␣STATES'
AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_city, s_city, d_year ORDER BY d_year asc, revenue desc

-- Q18
SELECT s_city, sum(lo_revenue) as revenue
FROM lineorder, supplier
WHERE lo_suppkey = s_suppkey AND s_nation = 'UNITED␣STATES'
GROUP BY s_city ORDER BY revenue desc


-- Q19
SELECT s_city, avg(lo_revenue) as avg_revenue
FROM lineorder, supplier WHERE
lo_suppkey = s_suppkey AND s_nation = 'UNITED␣STATES'
GROUP BY s_city ORDER BY avg_revenue desc

-- Q20
SELECT c_city, s_city, count(*)
FROM lineorder,customer,  supplier
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey
AND c_nation = 'UNITED␣STATES' AND s_nation = 'UNITED␣STATES'
GROUP BY c_city, s_city

-- Q21
SELECT c_city, s_city, d_year, avg(lo_revenue) as avg_revenue
FROM  lineorder,customer, supplier, dates
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_orderdate = d_datekey
AND c_nation = 'UNITED␣STATES' AND s_nation = 'UNITED␣STATES'
AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_city, s_city, d_year ORDER BY d_year asc, avg_revenue desc

-- Q22
SELECT c_city, s_city, d_year, count(*)
FROM lineorder,customer,  supplier, dates
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_orderdate = d_datekey
AND c_nation = 'UNITED␣STATES' AND s_nation = 'UNITED␣STATES'
AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_city, s_city, d_year ORDER BY d_year asc

-- Q23
SELECT d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit
FROM lineorder,dates, customer, supplier, part
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_partkey = p_partkey
AND lo_orderdate = d_datekey AND c_region = 'AMERICA' AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998) AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category ORDER BY d_year, s_nation, p_category

-- Q24
SELECT d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit
FROM lineorder, dates, supplier, part
WHERE lo_suppkey = s_suppkey AND lo_partkey = p_partkey AND lo_orderdate = d_datekey
AND s_region = 'AMERICA'  AND (d_year = 1997 or d_year = 1998)
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category ORDER BY d_year, s_nation, p_category

-- Q25
SELECT d_year, s_nation, p_category, avg(lo_revenue - lo_supplycost) as avg_profit
FROM lineorder, dates, customer, supplier, part
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_partkey = p_partkey
AND lo_orderdate = d_datekey AND c_region = 'AMERICA' AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998) AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category ORDER BY d_year, s_nation, p_category

-- Q26
SELECT d_year, s_nation, p_category, count(*)
FROM lineorder, dates, customer, supplier, part
WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_partkey = p_partkey
AND lo_orderdate = d_datekey AND c_region = 'AMERICA' AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998)  AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category ORDER BY d_year, s_nation, p_category
```

```
-- Q27
SELECT d_year, s_nation, count(*)
FROM  lineorder,dates, supplier
WHERE lo_suppkey = s_suppkey AND lo_orderdate = d_datekey AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998)
GROUP BY d_year, s_nation ORDER BY d_year, s_nation

-- Q28
SELECT s_nation, count(*) FROM lineorder,supplier
WHERE lo_suppkey = s_suppkey AND s_region = 'AMERICA'
GROUP BY s_nation ORDER BY s_nation

-- Q29
SELECT d_year, s_nation, sum(lo_revenue) as revenue
FROM lineorder,dates, supplier
WHERE lo_suppkey = s_suppkey AND lo_orderdate = d_datekey AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998)
GROUP BY d_year, s_nation ORDER BY d_year, s_nation

-- Q30
SELECT sum(lo_revenue), p_brAND
FROM lineorder, part, supplier
WHERE lo_partkey = p_partkey AND lo_suppkey = s_suppkey AND p_brAND = 'MFGR#2221'
AND s_region = 'ASIA'
GROUP BY p_brAND ORDER BY p_brAND
```

Bibliographie

[1] Rakesh Agrawal et Ramakrishnan Srikant. « Fast Algorithms for Mining Association Rules in Large Databases ». In : *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994, 487–499.

[2] Sanjay Agrawal, Vivek Narasayya et Beverly Yang. « Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design ». In : *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. SIGMOD '04. New York, NY, USA : Association for Computing Machinery, 2004, 359–370.

[3] Rafi Ahmed, Randall G. Bello, Andrew Witkowski et Praveen Kumar. « Automated Generation of Materialized Views in Oracle ». In : *VLDB Endow.* 13.12 (2020), p. 3046-3058.

[4] Ahmed M. Amin et Zeshan Chishti. « Rank-aware cache replacement and write buffering to improve DRAM energy efficiency ». In : *Proceedings of the 2010 International Symposium on Low Power Electronics and Design, 2010, Austin, Texas, USA, August 18-20, 2010*. ACM, 2010, p. 383-388.

[5] Vlasia Anagnostopoulou, Susmit Biswas, Heba Saadeldeen, Alan Savage, Ricardo Bianchini, Tao Yang, Diana Franklin et Frederic T. Chong. « Barely Alive Memory Servers : Keeping Data Active in a Low-Power State ». In : *J. Emerg. Technol. Comput. Syst.* 8.4 (2012). issn : 1550-4832.

[6] G. Antoshenkov. « Byte-aligned Bitmap Compression ». In : *DCC*. 1995, p. 476.

[7] K. Aouiche, J. Darmont, O. Boussaid et F. Bentayeb. « Automatic Selection of Bitmap Join Indexes in Data Warehouses ». In : *DaWaK*. 2005, p. 64-73.

[8] Raja Appuswamy, Matthaios Olma et Anastasia Ailamaki. « Scaling the Memory Power Wall With DRAM-Aware Data Management ». In : *Proceedings of the 11th International Workshop on Data Management on New Hardware*. ACM. 2015, p. 3.

[9] Stéphane Azefack, Kamel Aouiche et Jérôme Darmont. « Dynamic index selection in data warehouses ». In : *CoRR* abs/0809.1965 (2008). eprint : 0809.1965.

[10] James Bailey, Thomas Manoukian et Kotagiri Ramamohanarao. « A Fast Algorithm for Computing Hypergraph Transversals and its Application in Mining Emerging Patterns ». In : *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*. IEEE Computer Society, 2003, p. 485-488.

[11] Joffroy Beauquier et Béatrice Bérard. *Systèmes d'exploitation : concepts et algorithmes*. Paris : édiscience international, 1990.

[12] Peyman Behzadnia, Wei Yuan, Bo Zeng, Yi-Cheng Tu et Xiaorui Wang. « Dynamic Power-Aware Disk Storage Management in Database Servers ». In : *DEXA*. Springer International Publishing, 2016, p. 315-325.

[13] Ladjel Bellatreche et Kamel Boukhalfa. « An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse ». In : *Data Warehousing and Knowledge Discovery*. Sous la dir. d'A. Min Tjoa et Juan Trujillo. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, p. 115-125.

[14] Ladjel Bellatreche et Kamel Boukhalfa. « Yet Another Algorithms for Selecting Bitmap Join Indexes ». In : *Data Warehousing and Knowledge Discovery, 12th International Conference, DAWAK 2010, Bilbao, Spain, August/September 2010. Proceedings*. 2010, p. 105-116.

[15] Ladjel Bellatreche, Kamel Boukhalfa, Pascal Richard et Komla Yamavo Woameno. « Referential Horizontal Partitioning Selection Problem in Data Warehouses : Hardness Study and Selection Algorithms ». In : *IJDWM* 5.4 (2009), p. 1-23.

[16] Ladjel Bellatreche, Kamalakar Karlapalem et Ana Simonet. « Horizontal class partitioning in object-oriented databases ». In : *Database and Expert Systems Applications*. Sous la dir. d'Abdelkader Hameurlain et A. Min Tjoa. Berlin, Heidelberg : Springer Berlin Heidelberg, 1997, p. 58-67.

[17] Ladjel Bellatreche, Rokia Missaoui, Hamid Necir et Habiba Drias. « A Data Mining Approach for selecting Bitmap Join Indices ». In : *J. Comput. Sci. Eng.* 1.2 (2007), p. 177-194.

[18] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, Albert Zomaya et al. « A taxonomy and survey of energy-efficient data centers and cloud computing systems ». In : *Advances in Computers* 82.2 (2011), p. 47-111.

[19] Luca Benini et Giovanni de Micheli. « System-level power optimization : techniques and tools ». In : *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 5.2 (2000), p. 115-192.

[20] Claude Berge. *Hypergraphs : Combinatorics of Finite Sets*. North-Holland, 1989.

[21] Stephan Borzsonyi, Donald Kossmann et Konrad Stocker. « The Skyline operator ». In : *ICDE*. 2001, p. 421-430.

[22] Selma Bouarar. « Vers une conception logique et physique des bases de données avancées dirigée par la variabilité ». Thèse de doct. 2016.

[23] Selma Bouarar, Ladjel Bellatreche et Amine Roukh. « Eco-Data Warehouse Design Through Logical Variability ». In : *International Conference on Current Trends in Theory and Practice of Informatics*. Springer. 2017, p. 436-449.

[24] Rima Bouchakri, Boukhalfa Kamel et Ladjel Bellatreche. « Algorithmes de Sélection des Index de Jointure Binaires Mono et Multi Attributs ». In : *Ingénierie des Systèmes d'Information* 16 (déc. 2011), p. 91-116.

[25] Aurélien Bourdon, Adel Noureddine, Romain Rouvoy et Lionel Seinturier. « PowerAPI : A Software Library to Monitor the Energy Consumed at the Process-Level ». In : *ERCIM News*. Special Theme : Smart Energy Systems 92 (jan. 2013). Sous la dir. d'ERCIM, p. 43-44.

[26] David Brooks, Vivek Tiwari et Margaret Martonosi. « Wattch : A framework for architectural-level power analysis and optimizations ». In : *ACM SIGARCH Computer Architecture News* 28.2 (2000), p. 83-94.

[27] Dinh-Mao Bui, Yongik Yoon, Eui-Nam Huh, Sungik Jun et Sungyoung Lee. « Energy efficiency for cloud computing system based on predictive optimization ». In : *J. Parallel Distributed Comput.* 102 (2017), p. 103-114.

[28] Coral Calero et Mario Piattini. « Introduction to Green in Software Engineering ». In : *Green in Software Engineering*. Sous la dir. de Coral Calero et Mario Piattini. Cham : Springer International Publishing, 2015, p. 3-27.

[29] Matteo Catena et Nicola Tonellotto. « Energy-Efficient Query Processing in Web Search Engines ». In : *IEEE Transactions on Knowledge and Data Engineering* PP (mars 2017), p. 1-1.

[30] Stefano Ceri, Mauro Negri et Giuseppe Pelagatti. « Horizontal Data Partitioning in Database Design ». In : *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data, Orlando, Florida, USA, June 2-4, 1982*. Sous la dir. de Mario Schkolnick. ACM Press, 1982, p. 128-136.

[31] Surajit Chaudhuri, M Datar et Vivek Narasayya. « Index selection for databases : a hardness study and a principled heuristic solution ». In : *IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 11* (2004), 1313–1323.

[32] Surajit Chaudhuri et Vivek R. Narasayya. « Self-Tuning Database Systems : A Decade of Progress ». In : *VLDB*. 2007, p. 3-14.

[33] Nitin Singh Chauhan et Ashutosh Saxena. « A green software development life cycle for cloud computing ». In : *It Professional* 15.1 (2013), p. 28-34.

[34] Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang et Natarajan Gautam. « Managing Server Energy and Operational Costs in Hosting Centers ». In : SIGMETRICS '05. New York, NY, USA : Association for Computing Machinery, 2005, 303–314. isbn : 1595930221.

[35] Zhen Chen, Yuhao Wen, Junwei Cao, Wenxun Zheng, Jiahui Chang, Yinjun Wu, Ge Ma, Mour Hakm et Guodong Peng. « A Survey of Bitmap Index Compression Algorithms for Big Data ». In : *Tsinghua Science and Technology* 20 (fév. 2015), p. 11-100.

[36] Xuntao Cheng, Bingsheng He et Chiew Tong Lau. « Energy-efficient query processing on embedded CPU-GPU architectures ». In : *Proceedings of the 11th International Workshop on Data Management on New Hardware*. ACM. 2015, p. 10.

[37] United Nations Commission. « Report of the World Commission on Environment and Development : Our Common Future ». In : *Technical report* (1987).

[38] Stephen Creff. « Une modélisation de la variabilité multidimensionnelle pour une évolution incrémentale des lignes de produits ». Thèse. Université Rennes 1, déc. 2013.

[39] Leandro Cupertino, Georges Da Costa, Ariel Oleksiak, Wojciech Pia, Jean-Marc Pierson, Jaume Salom, Laura Siso, Patricia Stolf, Hongyang Sun, Thomas Zilio et al. « Energy-efficient, thermal-aware modeling and simulation of data centers : the CoolEmAll approach and evaluation results ». In : *Ad Hoc Networks* 25 (2015), p. 535-553.

[40] Carlo Curino, Yang Zhang, Evan P. C. Jones et Samuel Madden. « Schism : a Workload-Driven Approach to Database Replication and Partitioning ». In : *Proc. VLDB Endow.* 3.1 (2010), p. 48-57.

[41] Alfredo Cuzzocrea, Jérôme Darmont et Hadj Mahboubi. « Fragmenting very large XML data warehouses via K-means clustering algorithm ». In : *IJBIDM* 4.3/4 (2009), p. 301-328.

[42] Miyuru Dayarathna, Yonggang Wen et Rui Fan. « Data Center Energy Consumption Modeling : A Survey ». In : *IEEE Communications Surveys & Tutorials* 18.1 (2016), p. 732-794.

[43] Fabien De Marchi et Jean-Marc Petit. « Zigzag : A new algorithm for mining large inclusion dependencies in databases ». In : déc. 2003, p. 27 -34.

[44] Simon Pierre Dembele, Ladjel Bellatreche, Carlos Ordonez et Amine Roukh. « Think Big, Start Small : A Good Initiative to Design Green Query Optimizers ». In : *Cluster Computing* 23 (sept. 2020).

[45] Simon Pierre Dembele, Amine Roukh et Ladjel Bellatreche. « Vers des Optimiseurs Verts de Requêtes en Mode Parallèle ». In : *Business Intelligence & Big Data, 14ème Edition de la conference EDA, Tanger, Maroc, 4-6 octobar 2018*. 2018, p. 179-194.

[46] Cambridge Dictionary. « Sustainable definition ». In : ().

[47] Thanh Do, Suhib Rawshdeh et Weisong Shi. « ptop : A process-level power profiling tool ». In : *in Proceedings of the 2nd Workshop on Power Aware Computing and Systems (HotPower'09*. 2009.

[48] Guozhu Dong et Jinyan Li. « Mining border descriptions of emerging patterns from dataset pairs ». In : *Knowledge and Information Systems* 8 (2004), p. 178-202.

[49] Khosrow Ebrahimi, Gerard F. Jones et Amy S. Fleischer. « A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities ». In : *Renewable and Sustainable Energy Reviews* 31 (2014), p. 622 -638. issn : 1364-0321.

[50] Ramez Elmasri et Shamkant B. Navathe. *Fundamentals of Database Systems*. 6th. Pearson, 2015.

[51] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros et Rajkumar Buyya. « Virtual machine consolidation in cloud data centers using ACO metaheuristic ». In : *European conference on parallel processing*. Springer. 2014, p. 306-317.

[52] Jason Flinn et M. Satyanarayanan. « Managing Battery Lifetime with Energy-Aware Adaptation ». In : 22.2 (2004), 137–179. issn : 0734-2071.

[53] Jeremy Fowers, Greg Brown, Patrick Cooke et Greg Stitt. « A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-window Applications ». In : *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. FPGA '12. Monterey, California, USA : ACM, 2012, p. 47-56. isbn : 978-1-4503-1155-7.

[54] Amina Gacem et Boukhalfa Kamel. « Immune Algorithm for Bitmap Join Indexes ». In : *ICONIP*. 2012, p. 560-567.

[55] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Yang Hou et Liang Liu. « A multi-objective ant colony system algorithm for virtual machine placement in cloud computing ». In : *Journal of computer and system sciences* 79.8 (2013), p. 1230-1242.

[56] Hector Garcia-Molina, Jeffrey D. Ullman et Jennifer Widom. *Database systems - the complete book (2. ed.)* Pearson Education, 2009. isbn : 978-0-13-187325-4.

[57] Gemma Garriga. *Formal Methods for Mining Structured Objects*. T. 475. Jan. 2013. isbn : 978-3-642-36680-2.

[58] Chang Ge, Zhili Sun et Ning Wang. « A survey of power-saving techniques on data centers and content delivery networks ». In : *IEEE Communications Surveys & Tutorials* 15.3 (2013), p. 1334-1354.

[59] Rong Ge, Xizhou Feng et Kirk W. Cameron. « Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters ». In : (2005), p. 34.

[60] Issam Ghabri, Ladjel Bellatreche et Sadok Ben Yahia. « Energy Efficiency vs. Performance of Analytical Queries : The case of Bitmap Join Indexes ». In : *2021 IEEE International Conference on Big Data (Big Data)*. 2021, p. 3066-3074.

[61] Issam Ghabri, Ladjel Bellatreche et Sadok Ben Yahia. « Selection of a Green Logical Data Warehouse Schema by Anti-monotonicity Constraint ». In : *SOFSEM*. 2020, p. 350-361.

[62] Issam Ghabry, Sadok Ben Yahia et Nidhal Jelassi. « Selection of Bitmap Join Index : Approach Based on Minimal Transversals ». In : *DaWaK*. 2018, p. 302-316.

[63] Goetz Graefe. « Database servers tailored to improve energy efficiency ». In : *EDBT*. ACM. 2008, p. 24-28.

[64] Goetz Graefe et Harumi A. Kuno. « Modern B-tree techniques ». In : *ICDE*. 2011, p. 1370-1373.

[65] Binglei Guo, Jiong Yu, Bin Liao, Dexian Yang et Liang Lu. « A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing ». In : *Journal of Network and Computer Applications* 84 (2017), p. 118-130.

[66] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane Irwin, Narayanan Vijaykrishnan et Mahmut Kandemir. « Using complete machine simulation for software power estimation : The softwatt approach ». In : *Proceedings Eighth International Symposium on High Performance Computer Architecture*. IEEE. 2002, p. 141-150.

[67] Nicolas Gutowski, Olivier Camp et Eric Chauveau. « Measuring the Energy Consumption of Massive Data Insertions : An Energy Consumption Assessment of the PL/SQL FOR LOOP and FORALL Methods ». In : *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2017, p. 450-457.

[68] Matthias Hagen. « Algorithmic and Computational Complexity Issues of MONET ». Thèse de doct. Institut für Informatik, Friedrich-SchillerUniversität Jena, 2008.

[69] Abdelkader Hameurlain et Franck Morvan. « Evolution of Query Optimization Methods ». In : *T. Large-Scale Data- and Knowledge-Centered Systems* 1 (jan. 2009), p. 211-242.

[70] Ali Hammadi et Lotfi Mhamdi. « A survey on architectures and energy efficiency in Data Center Networks ». In : *Computer Communications* 40 (2014), p. 1 -21. issn : 0140-3664.

[71] Stavros Harizopoulos, Mehul Shah, Justin Meza et Parthasarathy Ranganathan. « Energy efficiency : The new holy grail of data management systems research ». In : *arXiv preprint arXiv :0909.1784* (2009).

[72] Ahmad Hassan, Hans Vandierendonck et Dimitrios S Nikolopoulos. « Energy-Efficient In-Memory Data Stores on Hybrid Memory Hierarchies ». In : *Proceedings of the 11th International Workshop on Data Management on New Hardware*. ACM. 2015, p. 1.

[73] Hasan Hassan, Minesh Patel, Jeremie Kim, Abdullah Giray Yaglikçi, Nandita Vijaykumar, Nika Ghiasi, Saugata Ghose et Onur Mutlu. « CROW : A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability ». In : *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*. 2019, p. 129-142.

[74] Yuto Hayamizu, Kazuo Goda, Miyuki Nakano et Masaru Kitsuregawa. « Application-Aware Power Saving for Online Transaction Processing Using Dynamic Voltage and Frequency Scaling in a Multicore Environment ». In : *Architecture of Computing Systems - ARCS 2011*. Sous la dir. de Mladen Berekovic, William Fornaciari, Uwe Brinkschulte et Cristina Silvano. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 50-61. isbn : 978-3-642-19137-4.

[75] Inki Hong, Darko Kirovski, Gang Qu, Miodrag Potkonjak et Mani B Srivastava. « Power optimization of variable-voltage core-based systems ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18.12 (1999), p. 1702-1714.

[76] Chung-Hsing Hsu et Ulrich Kremer. « The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction ». In : *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*. PLDI '03. San Diego, California, USA : Association for Computing Machinery, 2003, 38–48. isbn : 1581136625.

[77] Ali Hurson et Hamid Azad. *Energy Efficiency in Data Centers and Clouds*. Academic Press, 2016.

[78] Hagen Höpfner et Christian Bunse. « Towards an Energy Aware DBMS - Energy Consumptions of Sorting and Join Algorithms. » In : jan. 2009, p. 69-73.

[79] Matthias Jarke et Jürgen Koch. « Query Optimization in Database Systems ». In : *ACM Comput. Surv.* 16.2 (1984), p. 111-152.

[80] Nidhal Jelassi, Christine Largeron et Sadok Ben Yahia. « Efficient unveiling of multi-members in a social network ». In : *Journal of Systems and Software* 94 (août 2014), 30–38.

[81] Theodore Johnson. « Performance Measurements of Compressed Bitmap Indices ». In : *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*. 1999, p. 278-289.

[82] Hamidreza Kadkhodaei et Fariborz Mahmoudi. « A combination method for join ordering problem in relational databases using genetic algorithm and ant colony ». In : nov. 2011, p. 312-317.

[83] Christos Kalyvas et Theodoros Tzouramanis. « A Survey of Skyline Query Processing ». In : (avr. 2017).

[84] Kyo Kang, Sholom Cohen, James Hess, William Novak et A. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Rapp. tech. CMU/SEI-90-TR-021. Pittsburgh, PA : Software Engineering Institute, Carnegie Mellon University, 1990.

[85] Woochul Kang et Jaeyong Chung. « Energy-efficient Response Time Management for Embedded Databases ». In : *Real-Time Syst.* 53.2 (mars 2017), p. 228-253. issn : 0922-6443.

[86] Woochul Kang et Jaeyong Chung. « QoS management for embedded databases in multicore-based embedded systems ». In : *Mobile Information Systems* 2015 (2015).

[87] Woochul Kang, Sang H Son et John A Stankovic. « Power-Aware Data Buffer Cache Management in Real-Time Embedded Databases ». In : *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA'08. 14th IEEE International Conference on*. IEEE. 2008, p. 35-44.

[88] Aman Kansal et Feng Zhao. « Fine-Grained Energy Profiling for Power-Aware Application Design ». In : *SIGMETRICS Perform. Eval. Rev.* 36.2 (2008), 26–31. issn : 0163-5999.

[89] Krishna Kant. « Data center evolution : A tutorial on state of the art, issues, and challenges ». In : *Computer Networks* 53.17 (2009), p. 2939-2965.

[90] Tarandeep Kaur et Inderveer Chana. « Energy Efficiency Techniques in Cloud Computing : A Survey and Taxonomy ». In : *ACM Comput. Surv.* 48 (oct. 2015).

[91] Dimitris Kavvadias et Elias Stavropoulos. « An Efficient Algorithm for the Transversal Hypergraph Generation ». In : *J. Graph Algorithms Appl.* 9 (jan. 2005), p. 239-264.

[92] Mohamed Kechar et Safia Nait Bahloul. « Performance optimisation of the decision-support queries by the horizontal fragmentation of the data warehouse ». In : *Int. J. Bus. Inf. Syst.* 26.4 (2017), p. 506-537.

[93] Atefeh Khosravi, Saurabh Kumar Garg et Rajkumar Buyya. « Energy and carbon-efficient placement of virtual machines in distributed cloud data centers ». In : *European Conference on Parallel Processing*. Springer. 2013, p. 317-328.

[94] Sangchul Kim, Junhee Lee, Srinivasa Rao Satti et Bongki Moon. « SBH : Super byte-aligned hybrid bitmap compression ». In : *Information Systems* 62 (2016), p. 155 -168. issn : 0306-4379.

[95] Thomas Kissinger, Dirk Habich et Wolfgang Lehner. « Adaptive energy-control for in-memory database systems ». In : *Proceedings of the 2018 International Conference on Management of Data*. ACM. 2018, p. 351-364.

[96] Mustafa Korkmaz, Martin Karsten, Kenneth Salem et Semih Salihoglu. « Workload-Aware CPU Performance Scaling for Transactional Database Systems ». In : *Proceedings of the 2018 International Conference on Management of Data*. ACM. 2018, p. 291-306.

[97] Mustafa Korkmaz, A.Ye. Karyakin, Martin Karsten et Kenneth Salem. « Towards Dynamic Green-Sizing for Database Servers ». In : *ADMS@VLDB*. 2015.

[98] Mayuresh Kunjir, Puneet K. Birwa et Jayant R. Haritsa. « Peak power plays in database engines ». In : *EDBT*. ACM. 2012, p. 444-455.

[99] Willis Lang, Stavros Harizopoulos, Jignesh M Patel, Mehul A Shah et Dimitris Tsirogiannis. « Towards energy-efficient database cluster design ». In : *Proceedings of the VLDB Endowment* 5.11 (2012), p. 1684-1695.

[100] Willis Lang, Ramakrishnan Kandhan et Jignesh M Patel. « Rethinking Query Processing for Energy Efficiency : Slowing Down to Win the Race. » In : *IEEE Data Eng. Bull.* 34.1 (2011), p. 12-23.

[101] Willis Lang et Jignesh M. Patel. « Energy Management for MapReduce Clusters ». In : *Proc. VLDB Endow.* 3.1–2 (2010), 129–139. issn : 2150-8097.

[102] Willis Lang et Jignesh Patel. « Towards eco-friendly database management systems ». In : *arXiv preprint arXiv :0909.1767* (2009).

[103] Etienne Le Sueur et Gernot Heiser. « Dynamic Voltage and Frequency Scaling : The Laws of Diminishing Returns ». In : *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*. HotPower'10. Vancouver, BC, Canada : USENIX Association, 2010, p. 1-8.

[104] Xing Liu, Panwen Liu, Lun Hu, Chengming Zou et Zhangyu Cheng. « Energy-aware task scheduling with time constraint for heterogeneous cloud datacenters ». In : *Concurrency and Computation : Practice and Experience* 32.18 (2020), e5437.

[105] Yanan Liu, Xiaoxia Wei, Jinyu Xiao, Zhijie Liu, Yang Xu et Yun Tian. « Energy consumption and emission mitigation prediction based on data center traffic and PUE for global data centers ». In : *Global Energy Interconnection* 3.3 (2020), p. 272-282.

[106] Stéphane Lopes, Jean-Marc Petit et Lotfi Lakhal. « Efficient Discovery of Functional Dependencies and Armstrong Relations ». In : *Extending Database Technology (EDBT)*. Sous la dir. de Carlo Zaniolo, Peter C. Lockemann, Marc H. Scholl et Torsten Grust. T. 1777. Lecture Notes in Computer Science. Konstanz, Germany : Springer, mars 2000, p. 350-364.

[107] Yung-Hsiang Lu et Giovanni De Micheli. « Comparing system level power management policies ». In : *IEEE Design & test of Computers* 18.2 (2001), p. 10-19.

[108] Craig MacDonald, Nicola Tonellotto et Iadh Ounis. « Learning to predict response times for online query scheduling ». In : *SIGIR '12*. 2012.

[109] Kamesh Madduri et Kesheng Wu. « Efficient Joins with Compressed Bitmap Indexes ». In : *ACM CIKM*. 2009, p. 1017-1026.

[110] Divya Mahajan, Cody Blakeney et Ziliang Zong. « Improving the energy efficiency of relational and NoSQL databases via query optimizations ». In : *Sustainable Computing : Informatics and Systems* 22 (mars 2019).

[111] Hadj Mahboubi et Jérôme Darmont. « Data mining-based fragmentation of XML data warehouses ». In : *DOLAP 2008, ACM 11th International Workshop on Data Warehousing and OLAP, Napa Valley, California, USA, October 30, 2008, Proceedings*. Sous la dir. d'Il-Yeol Song et Alberto Abelló. ACM, 2008, p. 9-16.

[112] Nitesh Maheshwari, Radheshyam Nanduri et Vasudeva Varma. « Dynamic Energy Efficient Data Placement and Cluster Reconfiguration Algorithm for MapReduce Framework ». In : *Future Gener. Comput. Syst.* 28.1 (jan. 2012), p. 119-127. issn : 0167-739X.

[113] Imene Mami et Zohra Bellahsene. « A survey of view selection methods ». In : *SIGMOD Record* 41.1 (2012), p. 20-29.

[114] Irene Manotas, Lori Pollock et James Clause. « SEEDS : a software engineer's energy-optimization decision support framework ». In : *Proceedings of the 36th International Conference on Software Engineering*. ACM. 2014, p. 503-514.

[115] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith et Jonathan Koomey. « Recalibrating global data center energy-use estimates ». In : *Science* 367.6481 (2020), p. 984-986.

[116] Dustin McIntire, Thanos Stathopoulos et William Kaiser. « etop-Sensor Network Application Energy Profiling on the LEAP2 Platform ». In : avr. 2007, p. 576-577.

[117] David Meisner, Brian Gold et Thomas Wenisch. « PowerNap : Eliminating Server Idle Power ». In : t. 44. Mars 2009, p. 205-216.

[118] Ryszard S Michalski, Jaime G Carbonell et Tom M Mitchell. *Machine learning : An artificial intelligence approach*. Springer Science & Business Media, 2013.

[119] Keisuke Murakami et Takeaki Uno. « Efficient algorithms for dualizing large-scale hypergraphs ». In : *Discrete Applied Mathematics* 170 (2014), p. 83 -94.

[120] San Murugesan. « Harnessing Green IT : Principles and Practices ». In : *IT Professional* 10 (fév. 2008), p. 24 -33.

[121] Ripal Nathuji et Karsten Schwan. « VirtualPower : Coordinated power management in virtualized enterprise systems ». In : t. 41. Jan. 2007, p. 265-278.

[122] Hamid Necir. « A data mining approach for efficient selection bitmap join index ». In : *IJDMMM* 2.3 (2010), p. 238-251.

[123] Hamid Necir et Habiba Drias. « A distributed maximal frequent itemset mining with multi agents system on bitmap join indexes selection ». In : *IJITM* 14.2/3 (2015), p. 201-214.

[124] Rolf Neugebauer et Derek McAuley. « Energy is just another resource : energy accounting and energy pricing in the Nemesis OS ». In : *Proceedings Eighth Workshop on Hot Topics in Operating Systems*. 2001, p. 67-72.

[125] Amin Y. Noaman et Ken Barker. « A Horizontal Fragmentation Algorithm for the Fact Relation in a Distributed Data Warehouse ». In : *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, November 2-6, 1999*. ACM, 1999, p. 154-161.

[126] Stefan Noll, Henning Funke et Jens Teubner. « Energy Efficiency in Main-Memory Databases ». In : *Datenbank-Spektrum* 17.3 (2017), p. 223-232.

[127] Adel Noureddine, Aurelien Bourdon, Romain Rouvoy et Lionel Seinturier. « Runtime Monitoring of Software Energy Hotspots ». In : (sept. 2012).

[128] Adel Noureddine, Romain Rouvoy et Lionel Seinturier. « A Review of Energy Measurement Approaches ». In : 47.3 (2013), 42–49. issn : 0163-5980.

[129] Patrick E. O'Neil et Goetz Graefe. « Multi-Table Joins Through Bitmapped Join Indices ». In : *SIGMOD Record* 24.3 (1995), p. 8-11.

[130] Anne-Cecile Orgerie, Marcos Dias de Assuncao et Laurent Lefevre. « A survey on techniques for improving the energy efficiency of large-scale distributed systems ». In : *ACM Computing Surveys (CSUR)* 46.4 (2014), p. 47.

[131] M. Tamer Özsu et Patrick Valduriez, éd. *Principles of Distributed Database Systems*. 3e éd. New York : Springer, 2011. isbn : 978-1-4419-8833-1.

[132] Santiago Pagani, Anuj Pathania, Muhammad Shafique, Jian-Jia Chen et Jorg Henkel. « Energy Efficiency for Clustered Heterogeneous Multicores ». In : *IEEE Transactions on Parallel and Distributed Systems* 28.5 (2017), p. 1315-1330. issn : 1045-9219.

[133] Venkatesh Pallipadi et Alexey Starikovskiy. « The ondemand governor ». In : *Proceedings of the Linux Symposium*. T. 2. sn. 2006, p. 215-230.

[134] Youngwoo Park, Dong-Jae Shin, Sung Kyu Park et Kyu Ho Park. « Power-aware memory management for hybrid main memory ». In : *Next Generation Information Technology (ICNIT), 2011 The 2nd International Conference on*. IEEE. 2011, p. 82-85.

[135] Nicolas Pasquier, Yves Bastide, Rafik Taouil et Lotfi Lakhal. « Discovering Frequent Closed Itemsets for Association Rules ». In : *ICDT*. 1999, p. 398-416.

[136] Thomas Phan et Wen-Syan Li. « Dynamic Materialization of Query Views for Data Warehouse Workloads ». In : *2008 IEEE 24th International Conference on Data Engineering*. 2008, p. 436-445.

[137] Johan Pouwelse, Koen Langendoen et Henk Sips. « Dynamic Voltage Scaling on a Low-Power Microprocessor ». In : MobiCom '01. Rome, Italy : Association for Computing Machinery, 2001, 251–259. isbn : 1581134223.

[138] Mr BBK Prasad, N Nimisha, T Tejaswi, B Bhavani et K Prudhviraju. « Enhanced Energy Query Processing in Web Searching ». In : *Journal of Information and Computational Science* 10.3 (2020).

[139] Franco P. Preparata et Michael I. Shamos. *Computational Geometry : An Introduction*. Berlin, Heidelberg : Springer-Verlag, 1985. isbn : 0387961313.

[140] Iraklis Psaroudakis, Thomas Kissinger, Danica Porobic, Thomas Ilsche, Erietta Liarou, Pınar Tözün, Anastasia Ailamaki et Wolfgang Lehner. « Dynamic fine-grained scheduling for energy-efficient main-memory queries ». In : *Proceedings of the Tenth International Workshop on Data Management on New Hardware*. ACM. 2014, p. 1.

[141] Nguyen Quang-Hung, Nam Thoai et Nguyen Thanh Son. « Enery Efficient Allocation of Virtual Machines in High Performance Computing Cloud ». In : *CoRR* abs/1310.7801 (2013).

[142] Sherief Reda et Abdullah N Nowroz. « Power modeling and characterization of computing devices : a survey ». In : *Foundations and Trends in Electronic Design Automation* 6.2 (2012), p. 121-216.

[143] Suzanne Marion Rivoire. *Models and metrics for energy-efficient computer systems*. Stanford University, 2008.

[144] Suzanne Marion Rivoire. « Models and metrics for energy-efficient computer systems ». Thèse de doct. Stanford University, 2008.

[145] Suzanne Rivoire, Parthasarathy Ranganathan et Christos Kozyrakis. « A Comparison of High-Level Full-System Power Models. » In : *HotPower* 8 (2008), p. 3-3.

[146] Suzanne Rivoire, Mehul A Shah, Parthasarathy Ranganathan et Christos Kozyrakis. « JouleSort : a balanced energy-efficiency benchmark ». In : *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM. 2007, p. 365-376.

[147] Manuel Rodriguez-Martinez, Harold Valdivia, Jaime Seguel et Melvin Greer. « Estimating power/energy consumption in database servers ». In : *Procedia Computer Science* 6 (2011), p. 112-117.

[148] Mahsan Rofouei, Thanos Stathopoulos, Sebi Ryffel, William Kaiser et Majid Sarrafzadeh. « Energy-aware high performance computing with graphic processing units ». In : *Workshop on power aware computing and system*. 2008.

[149] Claude Ronneau. *Énergie, pollution de l'air et développement durable*. T. 1. Presses univ. de Louvain, 2004.

[150] A. Roukh, L. Bellatreche, A. Boukorca et S. Bouarar. « Eco-DMW : Eco-Design Methodology for Data Warehouses ». In : *DOLAP*. Melbourne, Australia, 2015, p. 1-10. isbn : 978-1-4503-3785-4.

[151] Amine Roukh. « Prise en compte de l'énergie dans la phase d'exploitation des bases de données volumineuses ». Thèse de doct. University Abdelhamid Ibn Badis Mostaganem, mai 2017.

[152] Amine Roukh et Ladjel Bellatreche. « Eco-processing of OLAP complex queries ». In : *Dawak*. Springer, 2015, p. 229-242.

[153] Amine Roukh, Ladjel Bellatreche, Ahcène Boukorca et Selma Bouarar. « Eco-Physic : Eco-Physical Design Initiative for Very Large Databases ». In : *Information Systems* 68 (2017), p. 44-62. issn : 0306-4379.

[154] Amine Roukh, Ladjel Bellatreche et Carlos Ordonez. « EnerQuery : Energy-Aware Query Processing ». In : *ACM CIKM*. 2016, p. 2465-2468.

[155] Bagher Salami, Hamid Noori et M. Naghibzadeh. « Fairness-Aware Energy Efficient Scheduling on Heterogeneous Multi-Core Processors ». In : *IEEE Transactions on Computers* (2020), p. 1-1.

[156] Daniel Schall, Volker Hudlet et Theo Härder. « Enhancing energy efficiency of database applications using SSDs ». In : *Proceedings of the Third C* Conference on Computer Science and Software Engineering*. ACM. 2010, p. 1-9.

[157] Chiyoung Seo, Sam Malek et Nenad Medvidovic. « An Energy Consumption Framework for Distributed Java-Based Systems ». In : ASE '07. Atlanta, Georgia, USA : Association for Computing Machinery, 2007, 421–424. isbn : 9781595938824.

[158] Hazim Shafi, Patrick J Bohrer, James Phelan, Cosmin A Rusu et James L Peterson. « Design and validation of a performance and power simulator for PowerPC systems ». In : *IBM Journal of Research and Development* 47.5.6 (2003), p. 641-651.

[159] K Shalini et K Naga Prasanthi. « Green Computing ». In : *Journal of Telematics and Informatics* 1.1 (2013), p. 1-13.

[160] Arman Shehabi, Sarah Josephine Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Nathaniel Horner, Inês Lima Azevedo et William Lintner. *United States Data Center Energy Usage Report*. Report. Energy Technology Area, 2016.

[161] Dong Keun Shin et Arnold Charles Meltzer. « A New Join Algorithm ». In : *SIGMOD Rec.* 23.4 (1994), 13–20. issn : 0163-5808.

[162] Youngsoo Shin et Kiyoung Choi. « Power conscious fixed priority scheduling for hard real-time systems ». In : *Design Automation Conference, 1999. Proceedings. 36th*. IEEE. 1999, p. 134-139.

[163] Junaid Shuja, Kashif Bilal, Sajjad Madani, Mazliza Othman, R. Ranjan, Pavan Balaji et Samee Khan. « Survey of Techniques and Architectures for Designing Energy-Efficient Data Centers ». In : *IEEE Systems Journal* 10.2 (2016), p. 507-519. issn : 1932-8184.

[164] Clauirton de Siebra, Paulo Costa, Fabio Q. B. da Silva, Andre Luís de Medeiros Santos et Angélica A. Mascaro. « The hardware and software aspects of energy consumption in the mobile development platform ». In : *Int. J. Pervasive Comput. Commun.* 9.2 (2013), p. 139-162.

[165] Bob Steigerwald et Abhishek Agrawal. « Developing green software ». In : *Intel White Paper* (2011).

[166] Mikael Svahnberg, Jilles van Gurp, Gurp And et Jan Bosch. « A taxonomy of variability realization techniques ». In : *Software-Practice & experience* 35.8 (2005), p. 705-754. issn : 0038-0644.

[167] Juha Taina. « Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors ». In : *CEPIS UPGRADE* 2011.4 (oct. 2011), p. 22-27. issn : 1684-5285.

[168] Constantin Timm, Andrej Gelenberg, Peter Marwedel et Frank Weichert. « Reducing the Energy Consumption of Embedded Systems by Integrating General Purpose GPUs ». In : (juin 2010).

[169] Nidhi Tiwari, Umesh Bellur, Santonu Sarkar et Maria Indrawan. « Optimizing MapReduce for energy efficiency ». In : *Software : Practice and Experience* 48.9 (), p. 1660-1687.

[170] Vivek Tiwari, Sharad Malik, Andrew Wolfe et Mike Lee. « Instruction Level Power Analysis and Optimization of Software ». In : t. 13. Sept. 1996. isbn : 978-1-4612-8633-2.

[171] Lyazid Toumi, Abdelouahab Moussaoui et Ahmet Ugur. « A Linear Programming Approach for Bitmap Join Indexes Selection in Data Warehouses ». In : *ANT*. 2015, p. 169-177.

[172] Lyazid Toumi, Abdelouahab Moussaoui et Ahmet Ugur. « EMeD-Part : An Efficient Methodology for Horizontal Partitioning in Data Warehouses ». In : *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*. IPAC '15. New York, NY, USA : Association for Computing Machinery, 2015.

[173] Lyazid Toumi, Ahmet Ugur et Yamina Azzi. « GPU-Based PSO For Bitmap Join Indexes Selection Problem In Data Warehouses ». In : *2019 International Conference on Advanced Electrical Engineering (ICAEE)*. 2019, p. 1-6.

[174] Anne E. Trefethen et Jeyarajan Thiyagalingam. « Energy-aware software : Challenges, opportunities and strategies ». In : *J. Comput. Sci.* 4 (2013), p. 444-449.

[175] Dimitris Tsirogiannis, Stavros Harizopoulos et Mehul A Shah. « Analyzing the energy efficiency of a database server ». In : *sigmod*. 2010, p. 231-242.

[176] Yi-Cheng Tu, Xiaorui Wang, Bo Zeng et Zichen Xu. « A system for energy-efficient data management ». In : *ACM SIGMOD Record* 43.1 (2014), p. 21-26.

[177] Vasanth Venkatachalam et Michael Franz. « Power reduction techniques for microprocessor systems ». In : *ACM Computing Surveys (CSUR)* 37.3 (2005), p. 195-237.

[178] Jun Wang, Ling Feng, Wenwei Xue et Zhanjiang Song. « A survey on energy-efficient data management ». In : *ACM SIGMOD Record* 40.2 (2011), p. 17-23.

[179] Yewan Wang. « Evaluating and modeling the energy impacts of data centers, in terms of hardware/software architecture and associated environment ». Thèse de doct. Ecole nationale supérieure Mines-Télécom Atlantique, 2020.

[180] Mark Weiser, Brent Welch, Alan Demers et Scott Shenker. « Scheduling for reduced CPU energy ». In : *Mobile Computing*. Springer, 1994, p. 449-471.

[181] Louis Woods, Zsolt István et Gustavo Alonso. « Ibex : an intelligent storage engine with support for advanced SQL offloading ». In : *Proceedings of the VLDB Endowment* 7.11 (2014), p. 963-974.

[182] Kesheng Wu, Ekow J. Otoo et Arie Shoshani. « Optimizing Bitmap Indices with Efficient Compression ». In : *ACM Trans. Database Syst.* 31.1 (2006), p. 1-38. issn : 0362-5915.

[183] Changjiu Xian, Yung-Hsiang Lu et Zhiyuan Li. « A programming environment with runtime energy characterization for energy-aware applications ». In : *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*. IEEE. 2007, p. 141-146.

[184] Zichen Xu, Yi-Cheng Tu et Xiaorui Wang. « Exploring power-performance tradeoffs in database systems ». In : *ICDE*. 2010, p. 485-496.

[185] Zichen Xu, Yi-Cheng Tu et Xiaorui Wang. « Online Energy Estimation of Relational Operations in Database Systems ». In : *IEEE Transactions on Computers* 64.11 (2015), p. 3223-3236.

[186] Zichen Xu, Yi-Cheng Tu et Xiaorui Wang. *Power Modeling in Database Mansgement Systems*. Rapp. tech. CSE/12-094. University of South Florida, Department of Computer Science et Engineering, 2012.

[187] Zichen Xu, Xiaorui Wang et Yi cheng Tu. « Power-Aware Throughput Control for Database Management Systems. » In : *ICAC*. 2013, p. 315-324.

[188] Chen Yang, Yongjie Du, Zhihui Du et Xiaofeng Meng. « Micro Analysis to Enable Energy-Efficient Database Systems ». In : *EDBT*. 2020.

[189] Alan G Yoder. « Energy efficient storage technologies for data centers ». In : *WEED 2010-Workshop on Energy-Efficient Design*. 2010.

[190] Erik Young, Paul Cao et Mike Nikolaiev. « First TPC-Energy Benchmark : Lessons Learned in Practice ». In : *Performance Evaluation, Measurement and Characterization of Complex Systems*. Sous la dir. de Raghunath Nambiar et Meikel Poess. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 136-152.

[191] P. S. Yu, J. Han et C. Faloutsos. *Link Mining : Models, Algorithms, and Applications*. 1st. Springer Publishing Company, Incorporated, 2010.

[192] Heng Zeng, Carla S Ellis, Alvin R Lebeck et Amin Vahdat. « ECOSystem : Managing energy as a first class operating system resource ». In : *ACM SIGPLAN Notices*. T. 37. 10. ACM. 2002, p. 123-132.

[193] Yu Zhang, Yansong Zhang, Mingchuan Su, Fangzhou Wang et Hong Chen. « HG-Bitmap Join Index : A Hybrid GPU/CPU Bitmap Join Index Mechanism for OLAP ». In : *WISE Workshops*. 2014, p. 23-36.

[194] Yi Zhou, Shubbhi Taneja, Xiao Qin, Wei-Shinn Ku et Jifu Zhang. « EDOM : Improving energy efficiency of database operations on multicore servers ». In : *Future Generation Computer Systems* (2017).

ملخص

في الوقت الحاضر، يتفق الجميع على أن المورد الأكثر قيمة في العالم لم يعد النفط بل أصبح البيانات. ولكن، مثل النفط، تعد البيانات مصدرًا للتلوث، ناتجًا بشكل أساسي عن معالجة كميات كبيرة من البيانات بواسطة الخوادم ومراكز البيانات. وقد دفع هذا، العديد من المنظمات والبلدان إلى تكريس جهود واستثمارات كبيرة لتقليل استهلاك الطاقة. لذلك فإن موفري حلول تخزين البيانات ومعالجتها هم في قلب النقاش حول تكنولوجيا المعلومات الخضراء. لذلك فإن موفري حلول تخزين البيانات ومعالجتها هم في قلب النقاش حول تكنولوجيا المعلومات الخضراء. حيث يجب أن تلبي هذه الحلول في نفس الوقت متطلبين غير وظيفيين متعارضين و هما، (١) الأداء و (٢) تقليل الطاقة. ترتبط هذه المتطلبات إرتباطًا وثيقًا بمعالجة الطلبات. على عكس الأداء، الذي تمت دراسته على نطاق واسع من قبل الأوساط الأكاديمية والصناعية، لا يحظى الحد من الطاقة بنفس الاهتمام. يركز العمل الحالي الذي يتعامل مع فعالية الطاقة لقواعد البيانات بشكل أساسي على التحسينات المنطقية وتعريفات نماذج التكلفة. ومع ذلك، لا يمكن لأحد أن ينكر أن تلبية أول متطلب غير وظيفي يتطلب بالضرورة تحسينات فعلية من خلال هياكل التحسين مثل الفهارس وأن مخطط قاعدة البيانات له تأثير مباشر على تحسينها. في هذه الأطروحة، نهدف إلى تقليل استهلاك الطاقة لقواعد البيانات من وجهة نظر البرمجيات من خلال دمج الطاقة في المراحل المختلفة من دورة حياتها. لتعزيز رؤيتنا للعالم الأخضر والدفاع عنها، نقدم أولاً نهجًا متعدد الأهداف لتحديد نوع من الفهارس يسمى فهرس الصلة الثنائي. في هذا النهج نقترح تعريف جديد لمشكلة اختيار هذا الفهرس من خلال دمج بُعد الطاقة. بعد ذلك، حاولنا التقاط الطاقة في وقت مبكر من دورة حياة قاعدة البيانات ودراسة تأثير تباين المخطط المنطقي على استهلاك الطاقة لقواعد البيانات. في كل من مساهمات هذه الرسالة، تم تطوير الخوارزميات من أجل تنفيذ الأساليب المقترحة الخوارزميات من أجل تنفيذ الأساليب المقترحة وإجراء تجارب مكثفة على معيار باستخدام الأجهزة لقياس استهلاك الطاقة.

الكلمات المفتاحية تصميم قواعد البيانات، إدارة قواعد البيانات، استهلاك الطاقة، معدات وإمدادات حفظ الطاقة، تخزين البيانات، تدقيق الطاقة، الرسوم البيانية الفائقة، التصميم المادي، التصميم المنطقي، فهرس الصلة الثنائي، التباين، مكافحة الرتابة، الحوسبة الخضراء، مشغل الأفق.

Résumé

Aujourd'hui, tout le monde s'est mis d'accord pour dire que la ressource la plus précieuse du monde n'est plus le pétrole mais les données. Mais, comme le pétrole, les données sont une source de pollution, principalement causée par le traitement de grandes quantités de données par les serveurs et les centres de données. Ce qui a incité de nombreuses organisations et pays à consacrer des efforts et des investissements considérables pour réduire la consommation d'énergie. Les fournisseurs de solutions de stockage et de traitement des données sont donc au cœur du débat sur l'informatique verte. Ces solutions doivent satisfaire en même temps deux besoins non-fonctionnels (BnF) conflictuels : (i) la performance des requêtes et (ii) la réduction de la consommation d'énergie. Ces BnFs sont fortement liées aux traitement des requêtes. Contrairement à la performance, qui a été largement étudiée par le monde universitaire et l'industrie, la réduction de l'énergie, ne reçoit pas la même attention. Les travaux actuels traitant l'efficacité énergétique (EE) des bases de données sont principalement axés sur les optimisations logiques et la définitions des modèles de coût. Cependant, personne ne peut nier que la satisfaction du premier BnF passe nécessairement par des optimisations physiques grâce aux structures d'optimisation telles que les indexes et que le schéma de la base de données à un impact direct sur son optimisation. Dans cette thèse, nous visons à réduire la consommation énergétique des bases de données d'un point de vue logiciel en intégrant l'énergie dans des différentes étapes de son cycle de vie. Pour promouvoir et défendre notre vision d'un monde vert, nous introduisons au premier lieu, une approche multi-objectif de sélection d'un type d'indexes appelé l'index de jointure binaire (IJB). Dans cette approche nous proposons une nouvelle formalisation du problème de sélection des IJBs en intégrant la dimension énergétique. Ensuite, nous avons essayé de capter l'énergie plutôt dans le cycle de vie de la BD et étudier l'impact de la variablité du schéma logique sur la consommation énergétique des bases de données. Dans chacune des contributions de cette thèse, des algorithmes sont développés afin de d'implémenter les approches proposées et des expérimentations intensives sont menées sur les données du benchmark SSB en utilisant des dispositifs matériels permettant de mesurer l'énergie consommées.

Mots-clés : Bases de données-Conception, Bases de données-Gestion, Consommation d'énergie, Économies d'énergie-Appareils et matériels, Entrepôts de données, Évaluation énergétique, Hypergraphes, Conception physique, Conception logique, Index de jointure binaire, Variabilité, Anti-monotonie, Informatique verte, Opérateur Skyline.

Abstract

Nowadays, everyone agrees that the world's most valuable resource is no longer oil but data. But, like oil, data is a source of pollution, mainly caused by the processing of large amounts of data by servers and data centers. This has prompted many organizations and countries to devote considerable effort and investment to reduce energy consumption. Providers of data storage and processing solutions are therefore at the heart of the green IT debate. These solutions must simultaneously satisfy two conflicting non-functional requirements (nFR) : (i) query performance and (ii) energy reduction. These nFRs are strongly related to query processing. Unlike performance, which has been widely studied by academia and industry, energy reduction, does not receive the same attention. Current work dealing with energy efficiency (EE) of databases is mainly focused on logical optimizations and cost models definitions. However, no one can deny that satisfying the first nFR necessarily requires physical optimizations through optimization structures such as indexes and that the database schema has a direct impact on its optimization. In this thesis, we aim to reduce the energy consumption of databases from a software point of view by integrating energy in the different stages of its life cycle. To promote and defend our vision of a green world, we first introduce a multi-objective approach to select a type of indexes called the bitmap join index (BJI). In this approach we propose a new formalization of the BJIs selection problem by integrating the energy dimension. Then, we tried to capture the energy earlier in the life cycle of the database and study the impact of the variability of the logical schema on the energy consumption of databases. In each of the contributions of this thesis, algorithms are developed in order to implement the proposed approaches and intensive experiments are conducted on SSB benchmark using hardware devices to measure energy consumption.

Keywords : Database design, Database management, Energy consumption, Energy conservation-Equipment and supplies, Data warehousing, Energy auditing, Hypergraphs, Physical Design, Logical Design, Bitmap Join Index, Variability, Anti-monotonicity, Green Computing, Skyline Operator.
