UNIVERSITÉ DU QUÉBEC À MONTRÉAL

OPTIMISATION ET AMÉLIORATION DE LA BASE DE DONNÉES DES PRODUITS POUR UNE PLATEFORME DE COMMERCE ÉLECTRONIQUE

RAPPORT DE PROJET
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE LOGICIEL

PAR
Luc-André JOLIVET

MONTRÉAL, LE 26 NOVEMBRE 2019

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Mme Sylvie Trudel, Directrice de projet
Département d'informatique à l'Université du Québec à Montréal

Mme Naouel Moha, Directrice du programme de maîtrise en Génie Logiciel
Département d'informatique à l'Université du Québec à Montréal

M. Hafedh Mili, Professeur
Département d'informatique à l'Université du Québec à Montréal

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 25 NOVEMBRE 2019

À L'UNIVERSITÉ DU QUÉBEC À MONTRÉAL

REMERCIEMENTS

Ce travail n'a pu être réalisé sans le soutien de personnes qui apportent chacune leur contribution, et ce, d'une manière différente.

En premier lieu, je remercie chaleureusement ma directrice Mme Sylvie Trudel, professeure à l'Université du Québec à Montréal. Son expérience et expertise professionnelle, ses vastes connaissances autant sur le plan académique que personnel, ont fait d'elle une personne clé dans la réussite, non pas que de ce projet synthèse, mais tout au long de mon cursus en génie logiciel.

En second, je remercie le vice-président aux opérations ainsi que le président de la ligne d'affaires qui ont offert leur soutien par l'entremise de dérogations facilitant grandement la conciliation tripartite travail étude projet synthèse.

En troisième, merci à François-Xavier Guillemette ainsi qu'à Patrick Sciascia pour leurs contributions tirées à partir de leur expérience professionnelle ou académique en tant qu'enseignant, directeur ou les deux à la fois. Ils étaient entièrement libres de refuser ou d'accepter et ont choisi de m'accorder de leur temps. J'adresse aussi mes remerciements à mes amis qui ont su se montrer patients et compréhensifs pendant de longues périodes d'absence et qui m'ont toujours soutenu.

Finalement, je tiens à souligner un merci tout à fait particulier aux deux grandes femmes dans ma vie, Liliana et ma mère, pour leur soutien, contributions et patiences indéfectibles. Elles ont toujours été présentes, peu importe les aléas de la vie et leurs apports inestimables.

Merci à toutes et à tous.

OPTIMISATION ET AMÉLIORATION DE LA BASE DE DONNÉES DES PRODUITS POUR UNE PLATEFORME DE COMMERCE ÉLECTRONIQUE

Luc-André JOLIVET

RÉSUMÉ

L'infonuagique est une technologie qui offre de nouvelles avenues, mais dont le modèle d'affaires, facturation aux ressources consommées, peut faire resurgir certaines problématiques. Celles-ci peuvent se transformer en problématique financière pour des applications très énergivores comportant des lacunes. Afin de réduire les coûts mensuels des ressources infonuagiques louées pour héberger la base de données des produits de la plateforme électronique, il est tout à l'intérêt de la ligne d'affaires de réduire son empreinte au niveau des ressources infonuagiques.

Dans le cadre de ce projet, la base de données des produits gruge une part excessive des ressources CPU et oblige la ligne d'affaires à devoir provisionner d'imposantes VMs et ceci érode sa marge bénéficiaire. Mon projet a été de quantifier la réduction possible de l'empreinte CPU de cette base de données des produits.

La méthodologie retenue, encadrée par la norme ISO 29110, a été de répertorier les options disponibles quant à la direction de la nouvelle base de données. Par la suite, une fois l'éventail de l'analyse des solutions achevé, le choix de reconceptualiser la base de données a été retenu.

Afin d'obtenir cette quantification le plus rapidement possible, le projet de reconceptualisation a pris son envol en tant que preuve de concept évolutive. Si les résultats étaient concluants, le projet se poursuivrait.

Pour y parvenir, deux variantes de diagrammes d'entités-relations ont été élaborées pour sa conception. S'en est suivi la création du code de migration des données entre la base de données actuelle et la nouvelle. Ensuite, il a y eu élaboration d'une batterie de tests afin d'assurer que les données demeurent identiques entre les deux conceptions. Un audit indépendant a aussi été conduit afin d'accroître l'objectivité et la validité des résultats.

Finalement, les mesures ont été mises en place pour quantifier la différence d'utilisation de CPU entre les deux bases de données.

La reconceptualisation de la base de données de produits s'est avérée concluante. La réduction moyenne des requêtes transactionnelles pour la charge CPU oscille entre 255 % et 408 % dépendamment du client testé et la réduction en temps d'exécution de 380 % à 1 974 %. Pour les requêtes impactant le non transactionnel, cette réduction du CPU oscille entre 368 % et 20 376 % et la réduction de son temps d'exécution entre 393 % et 98 708 %.

Dans l'optique de la continuité du projet et au-delà du but premier de l'apport quant à une meilleure prévisibilité des économies engendrées, des retombées additionnelles seront possibles. De ces bénéfices secondaires, il y aura, en outre, l'élimination des incohérences de traitement des données, une redécouverte complète des relations entre les entités, le renforcement automatique de ces relations par le SGBD, ce qui se traduira par une documentation implicite maintenue automatiquement par le SGBD lors de la génération du diagramme d'entités-relations, tout ceci afin d'accroître la compétitivité de la plateforme de commerce électronique.

OPTIMIZATION AND IMPROVEMENT OF THE PRODUCT DATABASE FOR AN ELECTRONIC COMMERCE PLATFORM

Luc-André JOLIVET

ABSTRACT

Cloud computing is a technology that offers new avenues, but its business model — billing resources consumed — may bring up some issues. These can turn into financial problems for energy-intensive applications with shortcomings. In order to reduce the monthly costs of cloud resources rented to host the electronic platform's product database, it is in the interest of the line of business to reduce its footprint in terms of cloud resources.

Within the context of this project, we studied the products database of an e-commerce platform that consumes an excessive amount of CPU resources and forces the line of business to provision large VMs which erodes its profit margin. My project was to quantify the possible reduction of the CPU footprint of this product database.

The chosen methodology, guided by the ISO 29110 standard, was to identify the available options for the new database. Subsequently, once all solution analyses were completed, we chose to redesign the database.

In order to obtain this quantification as quickly as possible, the redesign project has taken off as a progressive proof of concept. If the results were conclusive, the project would continue.

To achieve this, two variant entity-relationship diagrams have been developed for the database design. Then data migration code between the current database and the new one was developed. Then, a battery of tests was developed to ensure that the data remains the same between the two designs. An independent audit was also conducted to increase the objectivity and validity of the results. Finally, the measurements took place to quantify the difference in CPU usage between the two databases.

The redesign of the product database has been conclusive. The average reduction of transactional requests for the CPU load varies between 255 % and 408 % depending on the client tested and the reduction in execution time from 380 % to 1 974 %. For queries impacting

non-transactional, this CPU reduction varies between 368 % and 20 376 % and the reduction of its execution time between 393 % and 98 708 %.

In the context of the continuity of the project and beyond the primary purpose of the contribution to a better predictability of savings generated, additional benefits will be possible. Among these secondary benefits, we should mention the elimination of inconsistencies in data processing, a complete rediscovery of the relationships between the entities, the automatic enforcement of these relations by the DBMS, which will result in implicit documentation maintained automatically by the DBMS when generating the entity-relationship diagram.

All of which would increase the competitiveness of the e-commerce platform.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

BCNF	Boyce-Codd normal form (Forme normale de Boyce-Codd)
BD	Base de données
CPU	Central processing unit
CTO	Chief Technology Officer (Directeur de la technologie)
DBA	Database administrator (Administrateur de base de données)
DDL	Data definition language (Langage de définition de données)
DK/NF	Domain-Key normal form
DMV	Dynamic Management Views
E/S	Entrées / Sorties (Input / Output)
EAV	Entité Attribut Valeur (Entity-Attribute-Value)
ERD	Entity relationship diagram (Diagramme d'entité relations)
FD	Functional dependency (Dépendence fonctionnelle)
SAAS	Software as a service
SGBD	Système de gestion de base de données
SSMS	SQL Server Management Studio©
TDD	Test development driven (Dévelopement piloté par les tests)
TI	Technologie de l'information (IT)
UQAM	Université du Québec à Montréal
VM	Virtual machine (Machine virtuelle)

INTRODUCTION

Ce rapport est l'aboutissement d'un projet synthèse entrepris dans un cadre corporatif ayant au cœur de celui-ci les enseignements acquis durant la maîtrise en génie logiciel. La ligne d'affaires, subissant un fardeau financier à cause de l'une de ses bases de données peu performante, a mandaté son seul DBA afin de pallier ce problème. Étant donné que la quantification de l'amélioration de la performance était subjective et que le domaine des bases de données était moins bien maîtrisé par tous les autres employés, le projet a débuté en preuve de concept évolutive afin d'établir si les gains possibles se traduiraient en réduction tangible des coûts.

Les processus, règles d'implémentation, syntaxes et conceptions générales ciblent principalement le côté applicatif d'un logiciel. Le projet a donc incorporé une tangente additionnelle afin de déterminer si des concepts applicatifs sont applicables ou non aux bases de données.

Le rapport est structuré de la façon suivante :

- Le chapitre 1 présente l'organisation mandataire du projet. La portée et le contexte de sa plateforme électronique ainsi que les objectifs anticipés techniques et d'affaires.
- Le chapitre 2 offre la visibilité du cadre académique sur deux aspects : le processus de planification et la conception d'une base de données.
- Le chapitre 3 fait état des activités entreprises dans le cadre de la méthodologie employée pour la mise en œuvre de la conception de la base de données.
- Le chapitre 4 présente les artefacts produits des activités du chapitre 3.
- Le chapitre 5 établit la confirmation que la norme ISO 29110 transparait à travers les activités du chapitre 3.
- Le chapitre 6 discute des retombées et particularités survenues en cours de route.

CHAPITRE 1

ORGANISATION D'ACCUEIL ET PROBLÉMATIQUE DE SA BASE DE DONNÉES DES PRODUITS

1.1 Mission

L'organisation mandataire[1] de ce projet exerce son expertise dans le domaine du commerce électronique ciblant deux créneaux : celui du commerce de détail ainsi que celui de l'épicerie. Elle soutient neuf clients comptabilisant dix-neuf bannières, dont certaines de grande taille qui sont très bien connues à l'échelle québécoise jusqu'à son client le plus imposant, une branche du plus grand détaillant d'Europe, desservant un bassin de population de plus de soixante millions d'habitants. L'organisation définit sa mission première selon les grands principes suivants :

« *Permettre à de grandes marques de s'adapter, d'être plus performantes et d'innover au sein de l'économie numérique* ».

À titre d'exemple Shopify© et Magento© sont des compétiteurs directs.

1.2 Contexte de l'entreprise

À l'heure actuelle, l'entreprise vogue vers des horizons beaucoup plus cléments, mais il n'en a pas toujours été ainsi. Par le passé, l'entreprise a crû à une vitesse supérieure à ce qui aurait été nécessaire en trop peu de temps, atteignant environ 180 employés. Cette croissance soudaine a créé des distorsions au niveau de la chaîne de commandement technique. Ceci a provoqué des dissensions techniques, conceptuelles et architecturales.

Au cœur de la tourmente entourant sa vente due à cette croissance trop rapide, le personnel, toutes catégories confondues, s'est vu amputé d'environ 83 % ce qui a provoqué

---

[1] Le nom de l'organisation n'est pas divulgué à des fins de confidentialité et l'ensemble des noms ont été anonymisés

d'innombrables pertes en connaissance dans le spectre technique au grand complet en peu de temps. L'exode s'est ensuite poursuivi provoquant davantage d'érosion dans les connaissances. Pour ce présent projet, ce qui est important de connaitre, c'est qu'il n'est resté qu'un seul DBA nouvellement recruté, c'est-à-dire moi, l'auteur de ce document.

Pendant ce temps, la dette technique, les problèmes courants, les tâches à effectuer « ToDo », les tâches manuelles demandant à être automatisées, n'ont pas diminué proportionnellement au nombre d'employés restant, ce qui les surcharge. N'ayant plus le contrôle des finances et du recrutement, ceux-ci étant relayés à la compagnie mère, cela réduit considérablement la marge de manœuvre et la portée des actions possibles devant des projets ou des incidents. Pour clore le tout, sa délocalisation (sur la Rive-Sud de Montréal) nuit au recrutement de personnel qualifié en plus de devoir entrer en compétition de priorité avec les autres lignes d'affaires de la compagnie mère pour pourvoir des postes similaires.

À l'heure actuelle, seulement une poignée très limitée d'individus connaissent minutieusement certaines sections de la plateforme, mais aucun pour ce qui touche les raisons des choix conceptuels dans les bases de données, dont celle des produits.

De la documentation technique demeure disponible, mais elle est incomplète, comporte des erreurs, est non axée sur les besoins ou obsolète et donc peu utile.

> « *The maintenance of an existing database depends on the depth of understanding of its characteristics. Such an understanding is easily lost when the developers disperse. The situation becomes worse when the related documentation is missing.* » (Reda, 2003) p.1

1.3 Contexte de la plateforme

La plateforme de commerce électronique, entièrement dans l'infonuagique, s'offre contractuellement sous le format Software as a service (SAAS) ciblant des clients / bannières de taille moyenne ou plus. Des tierces parties, sous forme de partenariat ou son département de service à l'interne, s'appuient sur cette plateforme pour développer la personnalisation requise.

Cette plateforme électronique est le seul produit de l'organisation autour de laquelle gravitent toutes les autres améliorations / options. Un client peut être le propriétaire d'une ou plusieurs bannières, donc user d'une ou de multiples instances de la plateforme. Il n'y a pas de multi entité (multitenant) et d'aucune façon un client ou une bannière ne peut se prévaloir de services informatiques s'il n'utilise pas la plateforme au cœur de ses opérations en ligne.

Le développement oriente l'architecture de sa plateforme vers le type « micro service ». De nouvelles fonctionnalités commencent à tirer avantage de cette architecture pour ne déployer que ce pour quoi le client adhère contractuellement. Pour le cœur de la plateforme, dont les éléments sont nécessaires à son fonctionnement minimal, ceci est davantage dans un but souhaité de maintenabilité et de flexibilité de redimensionnement des ressources machines (scaling out) si nécessaire. Une topologie minimale peut tenir sur une seule machine virtuelle (VM) tandis que les topologies les plus imposantes s'étalent sur des dizaines de VMs. Ce surdimensionnement a peu de limitation (excepté au niveau des bases de données). Il suffit de provisionner de nouvelles VM et d'y installer le service voulu pour accroitre la performance de la solution desservant cette fonctionnalité.

La plateforme gère plusieurs aspects dont le catalogue des produits, des facettes du domaine d'affaires du client définissables sur mesure, effectue la gestion des commandes, des factures, de l'expédition, des taxes et des campagnes publicitaires et des promotions; elle possède un module d'analytique, un entrepôt de données, mais délègue le service de paiement à de tierces parties pour ne pas avoir à se soucier des contraintes de sécurité entourant la gestion de la transaction financière. Dans son fonctionnement interne, elle gère aussi les accès des utilisateurs finaux aux différents modules et fonctionnalités, gère les tâches, les messages transmis entre les diverses composantes et possède la possibilité d'importer des catalogues tiers (la mise à jour des prix par exemple). La totalité des services est soutenue par un minimum de sept bases de données indépendantes.

Dans la vie courante de la base de données propre au catalogue des produits, elle avait été gérée (avant le départ massif des ressources) par les DBAs antérieurs et maintenant, elle l'est principalement par les développeurs dédiés à la plateforme. De plus, au cours de l'évolution d'un projet client, les développeurs dédiés à la personnalisation (le « Service »), pouvaient à leur tour y apporter des modifications si nécessaire et donc, à partir de ce point

une « nouvelle branche » de celle-ci venait de voir le jour. Finalement, dans les partenariats, à cause d'une contrainte d'affaires, maintenue jusqu'à tout récemment, il était possible pour quiconque d'étendre la plateforme, incluant les bases de données, comme bon leur semblait (ajout de tables, colonnes, procédures stockées, tout objet SQL). Somme toute, un tronc commun existe, mais la quasi-totalité des instances des bases de données de la plateforme comporte des modifications propres à un client ou à une bannière.

1.4 Réduction des problèmes techniques

Dans un but de réduire ses coûts opérationnels et de personnel, la plateforme se déploie dans l'infonuagique par l'entremise de location de VMs. Lors de la signature d'un nouveau contrat, un ensemble de VMs est provisionné pour desservir la plateforme ainsi que la personnalisation de celle-ci.

Il y a deux volets au côté technique. Un besoin primaire explicitement demandé par la direction, et d'autres secondaires qui, selon les zones impactées, seront un moment opportun pour diminuer la dette technique et améliorer la qualité. La plateforme de commerce électronique se veut un projet sur le long terme. Miser sur la qualité revêt donc une grande importance.

> « *It is well known that up-front quality techniques save more in later phases than they cost (and improve quality and customer satisfaction)* » (Hinkle, 2007) p.1

1.4.1 Besoin primaire en réduction du coût opérationnel des VMs

La modélisation logique de la hiérarchie des commerces (à l'image d'un arbre en informatique, une racine et des descendants), expliquée en détail dans l'annexe I « *Commerce modeler* », se base sur une gestion dynamique de ceux-ci dans la base de données des produits. Un « scope » (un *nœud*) est le mot désigné pour représenter un commerce, un territoire ou le siège social (la *racine*) d'un client. Cette gestion dynamique des *scopes* est définie comme suit : pour chaque *scope*, peu importe le niveau où il se situe dans la hiérarchie, ça implique un ajout d'un nouvel ensemble de six tables au minimum dans la base de données. Si ce *scope* est de premier ou de second niveau, sept tables additionnelles sont rajoutées pour un

total de treize. Pour un modèle d'affaires d'un client possédant des centaines de commerces, donc de *scopes*, il est alors commun d'obtenir plusieurs milliers de tables. Pour un prospect ayant des dizaines de milliers de *scopes*, il est certain d'obtenir des centaines de milliers de tables. La base de données permet jusqu'à 2 milliards (2^{31} - 1) d'objets (incluant les tables), donc la limitation du SGBD n'est pas imminente de ce côté.

Afin de départager à quel *scope* une table fait référence, la conception initiale a incorporé le nom du *scope* dans l'objet créé, ici le nom de la table. Le fait de devoir gérer dynamiquement le nom des tables (ex. : « *[NomScope]_ITEM* »), oblige les développeurs à utiliser du code SQL dynamique dans les procédures stockées. Ce SQL dynamique permet de concaténer des chaînes de caractères représentant du code SQL pour formuler une nouvelle ligne de code et ensuite pour pouvoir l'exécuter. Sans discuter des problèmes de sécurité, de la difficulté de maintenabilité et de la lisibilité engendrés par l'utilisation de cet outil, le SQL dynamique, la concaténation de chaînes de caractères nécessite des ressources CPU et ce, avant même l'exécution de la requête générant le jeu de données attendu par l'applicatif. Ce constat, la taxation du CPU par la concaténation des chaînes de caractères, avait déjà été soulevé au début 2017. La démonstration, à l'époque, de ceci a été reproduite et se trouve à l'annexe II « *Tests et résultats de performance de la concaténation de chaînes de caractères* ».

En conséquence, sur certains serveurs, il a été observé que la ressource CPU avait atteint 99,5 % d'utilisation monopolisant celle-ci au détriment des autres bases de données. Afin de pallier temporairement ce problème, de puissants serveurs, un par client, sont provisionnés en parallèle afin d'héberger chacune de ces instances de cette base de données des produits, ce qui, de façon sous-jacente, érode substantiellement la marge de profit selon le modèle d'affaires contractuel dont s'est dotée la ligne d'affaires. Un des clients a déjà atteint la puissance maximale des VMs disponibles en location pour le type de licences acquises.

De ce fait, la réduction de la taxation de l'utilisation CPU par la base de données des produits est le besoin technique primaire demandé par le président de la ligne d'affaires. Par contre, toute autre information était demeurée implicite. Aucune exigence non fonctionnelle n'avait été formellement documentée. Quant au président de la ligne d'affaires, il s'attendait à ce que la solution élaborée soit égale ou meilleure que ce qui était en place. La valeur finale, dans

l'ensemble des objectifs de la ligne d'affaires, était la réduction des coûts de l'utilisation des ressources soutenant la plateforme électronique pour augmenter sa marge bénéficiaire. Cette réduction des coûts était due au fait que la maison-mère avait imposé de modifier le modèle d'affaires de la ligne d'affaires pour facturer au client non plus par machine ou ressources nécessaires au fonctionnement de la plateforme, mais (en résumé) au nombre de commandes que le client fait transiger à l'aide de la plateforme. Selon ce modèle, la facturation implicite de la performance nécessaire des VMs est transposée du client vers la ligne d'affaires. Le client, originalement, payait X ressources pour générer une commande, selon le nouveau modèle de facturation, il paye par commande *peu importe les ressources sous-jacentes utilisées* pour générer celle-ci. Ceci est non-dit selon les objectifs d'affaires énoncés qui entrevoient qu'une moindre utilisation du CPU équivaut *inévitablement* à une réduction *globale* de l'utilisation machine et donc, supposément à une réduction de coûts afin qu'il nécessite moins en ressources pour générer une commande et donc augmenter sa marge bénéficiaire et devenir plus compétitif de cette façon. Ce ne sont pas tous les employés qui sont au fait des détails de cette information. Donc, le projet, malgré l'absence de cette directive, pourrait être rejeté selon l'exemple du cas suivant : *Une réduction importante de l'utilisation du CPU par une compensation d'utilisation d'autres ressources non mentionnées à l'origine, ex : la détérioration d'exigences non fonctionnelles autres ayant des impacts négatifs pour les ventes, générant de la maintenance applicative additionnelle ou imposant de fortes charges sur la RAM ou le I/O.* Donc, un projet dont la portée de la connaissance du processus d'affaires des intervenants se serait limitée aux directives reçues, soit la réduction du CPU, pourrait s'avérer être un échec même si cette ressource CPU était significativement réduite, donc prise au pied de la lettre uniquement. Ceci, en retour, positionnerait le président de la ligne d'affaires dans une situation délicate ayant entre les mains une dépense de ressources générant une initiative ratée vis-à-vis des attentes de la maison mère et de ses objectifs. Telle est la vraie définition du succès de ce projet, implicitement attendu par le président de la ligne d'affaires et des dirigeants la maison mère.

1.4.2 Amélioration secondaire souhaitée des attributs de qualité

Au fil du temps, l'évolution de la base de données des produits s'est développée en parallèle par plusieurs équipes aux expériences variées qui ont chacune apporté leur vision sur la façon

dont celle-ci devrait gérer les données, ce qui a amené un lot de problèmes additionnels disparates.

La qualité générale s'est détériorée. Par exemple, la normalisation de la nomenclature, l'attribution des types de colonnes, la duplication de code ne sont que quelques cas. De plus, de par l'utilisation du SQL dynamique, la lisibilité est rudement mise à l'épreuve pour quiconque souhaite entamer les plus simples des modifications dans le code.

À travers les changements demandés, la nouvelle base de données des produits sera considérée comme celle de « référence », intègre quant aux données qu'elle contiendra. Il est souhaité que la nouvelle base de données soit davantage maintenable, ayant du code plus clair tout en ayant une performance accrue par le retrait du SQL dynamique.

Les qualités recherchées sont regroupées sous trois termes globaux : maintenabilité, convivialité et performance applicative. Il est à noter que les parties impliquées ici représentent les utilisateurs en termes de développement applicatif, les partenaires, les utilisateurs de la plateforme ainsi que les gestionnaires, chacun ayant un impact quelconque, mais sous un angle différent.

Les raisons pour ces attributs de qualité sont en trois points. Tout d'abord, les commentaires en sont peu élogieux lorsque vient le temps d'interagir avec la base de données. Un exemple d'un collègue de travail par rapport à la modification de code dans la base de données des produits :

Luc-Andre Jolivet 11:08 AM:
How do you feel it's easy to read & maintain?
Luc-Andre Jolivet 11:08 AM:
It's not a tricky question
[redacted] **11:09 AM:**
Hey Luc-Andre I seriously am not the best person to ask, cause I believe stored procedures are sent directly from hell

En second, la littérature enseigne que la majeure partie des coûts, entre 60 à 90% (Martin et al., 2009), (Glass, 2001), (Dehaghani, Hajrahimi, 2013) provient de la maintenance. En troisième, la base de données a une durée de vie utile sur le long terme, ce qui accentue la pression afin de diminuer son coût global de maintenance.

« Une grande partie du coût d'un projet logiciel se trouve dans la maintenance à long terme... Plus l'auteur saura écrire du code clair, moins les autres développeurs passeront du temps à le comprendre. Cela permettra de diminuer les défauts et le coût de la maintenance. » (Martin et al., 2009) p.188

Par contre, ce ne sont pas tous les attributs qui importent dans le présent projet. Un exemple, la portabilité : il n'est pas envisagé, ni de près ni de loin, de passer à un autre SGBD équivalent tel que MySQL© ou Oracle©. Malgré qu'il pourrait être techniquement intéressant d'obtenir une portabilité « au cas où », ajouter des efforts pour retirer le code propriétaire et le remplacer par celui de la norme ANSI SQL n'apporterait aucun retour sur investissement. De plus, il n'est pas garanti que les autres SGBD adhèrent parfaitement aux mêmes implémentations de cette norme et de la même version. Pour finir, le plus important, ce n'est pas une exigence d'affaires et encore moins dans la vision d'affaires à long terme.

1.5 Problématiques de l'offre d'affaires

À l'origine de la plateforme, des orientations d'affaires ont dirigé des choix architecturaux qui persistent à ce jour. Or, dans un marché en constante expansion et évolution, des besoins, jadis non nécessaires, font maintenant surface et l'architecture actuelle est une entrave voire un frein à cette croissance. Les vendeurs doivent éviter certaines questions ou trouver des compromis afin de satisfaire la clientèle actuelle ou amoindrir les attentes de nouveaux clients à propos de fonctionnalités que la plateforme ne supporte pas convenablement.

Les améliorations à apporter à la base de données, d'un point de vue d'offre d'affaires, se scindent en deux volets. Tout comme les problèmes techniques, il existe une exigence principale, explicitement demandée par la direction, et des exigences secondaires souhaitées.

1.5.1 Amélioration de l'offre d'affaires principale

La plateforme permettant une hiérarchie jusqu'à un maximum de trois niveaux, offre une flexibilité totale de gestion des produits des commerces pour ses deux premiers niveaux, soit

le siège social et les territoires. Le troisième niveau, celui des commerces, demeure limité en fonctionnalités. Lorsque des clients actuels ou de nouveaux prospects demandent à pouvoir gérer leurs produits dans leurs commerces avec le niveau de granularité la plus fine, le cadre conceptuel de la base de données des produits l'empêche. Des solutions de contournement sont employées, mais elles demeurent fragiles, ne répondent pas exactement au besoin du client ou surchargent d'autres parties de la plateforme qui n'étaient pas destinées à l'origine pour ce genre de manipulations. Ceci limite les opportunités d'affaires d'expansion des clients courants et entrave le travail des ventes lors de nouveaux prospects ou de nouveaux contrats.

La version actuelle n'est pas aisément extensible pour la gestion des produits, car ceux-ci s'entremêlent dans l'implémentation d'un héritage rendant le cadre conceptuel rigide. À diverses reprises, les gestionnaires de produits en ont fait la demande, de compléter ce qui avait été entrepris, mais en vain, avec comme raison que la conception actuelle ne le permettait pas.

Le besoin d'affaires primaire est de donner la possibilité de manipuler les champs « actif/inactif » ou « visible/caché » des produits et de leurs variantes indépendamment du niveau. Sans ces options, la plateforme n'offre pas de moyen préconçu à la base pour manipuler l'état d'un simple produit dans un commerce précis.

1.5.2 Amélioration de l'offre d'affaires secondaire

Découlant de la hiérarchie à trois niveaux, des contraintes techniques additionnelles entravent l'expansion du domaine d'affaires. Ces contraintes techniques demeurent tolérables pour l'instant, mais il y a déjà des cas que les gestionnaires de produits appréhendent comme un frein à l'expansion future de la plateforme. Afin d'accroitre davantage la flexibilité de la plateforme et donc de pouvoir couvrir un éventail plus large des besoins d'affaires des clients, il en résulte trois besoins d'affaires secondaires qui ont été discutés et approuvés pour en tenir compte lors de l'implémentation de la nouvelle base de données des produits.

Premièrement, la plateforme est limitée quant au nombre de commerces qu'une bannière peut desservir. Ceci porte atteinte aux clients ayant une offre de service à grande échelle. À l'heure actuelle, la limite opérationnelle se situe aux alentours de 400 à 500 commerces. Des

tests, dans un environnement contrôlé, ont tenté d'en atteindre 4 000, mais faisant surgir plusieurs problèmes et rendant la plateforme quasi inopérable. Un prospect antérieur nécessitait l'expansion de ce nombre jusqu'à 15 000. Dans le cas le plus poussé, un modèle d'affaires d'un prospect en cours inciterait à élever ce nombre jusqu'à 240 000. Toutefois, pour ce dernier, le président, siégeant également à titre de CTO « *Chief Technology Officer* » confirme que la plateforme ne dépasserait pas la gestion de plus de 30 000 commerces, reconduisant tout nombre plus élevé à revoir leur modèle d'affaires ou de le pallier autrement.

Deuxièmement, la plateforme dans son architecture, ne fixe que trois types de niveaux de *scopes* au sein de sa hiérarchie pour la gestion des produits. Un effort a été mis en place pour obtenir une flexibilité additionnelle par l'entremise de *scopes* virtuels. Toutefois, ceux-ci ne permettent qu'un regroupement logique additionnel, sans permettre d'y gérer des produits et cette fonctionnalité est peu utile de par sa faible portée. Donc, éliminer la limite fixe de trois niveaux de *scopes* sera un atout certain.

Troisièmement, au dernier niveau, celui des commerces, il n'est pas possible d'éditer les attributs des produits. Donc, étendre la flexibilité complète de la gestion des produits des deux premiers niveaux au troisième sera aussi un avantage futur certain.

1.6 Démonstration de la problématique

Avant de modifier quoi que ce soit, il était nécessaire de cibler le bon problème. Fort heureusement, cette étape était connue et démontrée depuis le milieu de 2017. Pour la même procédure stockée, il s'agissait de calculer le temps de concaténation de chaînes de caractères ainsi que celui de son exécution puis de comparer les résultats. Après exécution, 98,89 % du temps était consacré à la concaténation et 1,11 % au code généré final. Les deux parties combinées, vue de l'externe de la procédure stockée, totalisaient 99,95 % d'utilisation CPU. Les détails de cette conclusion se retrouvent en annexe II « *Tests et résultats de performance de la concaténation de chaînes de caractères* ».

La concaténation de chaînes de caractères n'est pas aussi efficace du côté base de données que du côté applicatif. L'utilisation de la concaténation à des fins de construction de la prochaine requête à exécuter (SQL dynamique) de façon généralisée est contre-productive

et ceci existe dans la littérature applicative depuis au moins **2002** (date de parution du livre de Fowler):

> « *It's **always** **worth making the effort to use static SQL that can be precompiled, rather than dynamic SQL** that has to be compiled each time. Most platforms give you a mechanism for precompiling SQL. **A good rule of thumb is to avoid using string concatenation to put together SQL queries.*** » (Fowler, 2002) p.53

Un exemple de 2017 démontrant l'utilisation du code SQL dynamique afin de retourner les détails d'un produit ainsi que le code utilisé pour l'observation se retrouve en annexe III « *Exemple de code provenant de la base de données des produits* ».

CHAPITRE 2

ÉTAT DE L'ART RELATIF AUX BASES DE DONNÉES

2.1 Méthodologie

Ma revue de l'état de l'art a ciblé principalement la modélisation des bases de données et les différentes activités requises pour en faire une refonte. Elle s'était limitée essentiellement à trois sources. D'abord, aux bases de données scientifiques disponibles via le service des bibliothèques de l'UQAM (Université du Québec à Montréal), en ne filtrant que les documents révisés par des pairs. Ensuite, l'ensemble de ce qui a été enseigné au fil des cours du cursus de génie logiciel. Enfin, les livres publiés à grande échelle ont été aussi consultés.

2.2 Cas réels

La littérature obtenue, propre à la modélisation de base de données, pour des cas concrets, demeurait faible et les modèles présentés étaient de peu d'utilité pour les présents besoins. Soit qu'ils étaient trop simplistes (Robert, 2007), couvraient un domaine d'affaires trop précis ayant ses propres contraintes fortement différentes de celles actuelles (Green, 1996), couvraient une sous fonctionnalité propre aux produits, mais dont la problématique était déjà résolue autrement (Lee et al., 2006) dans la plateforme actuelle ou ciblaient des éléments propres à la gestion de la technologie elle-même et non à ce qu'elle devait représenter.

« *most methods used in database design are, in fact, empirical solutions, often, unsupported by any scientific basis or any engineering discipline.* » (Cotelea, 2012) p.24

À défaut de pouvoir user de cas probants en littérature scientifique, des alternatives ont été envisagées et seront décrites dans la méthodologie.

2.3 Approche théorique de mise en œuvre

2.3.1 Modélisation de la nouvelle base de données

La faible teneur en cas concrets m'a forcé à me rabattre sur le volet théorique de la modélisation de bases de données. Pour débuter, l'aspect fondamental d'une base de données se résume à la façon dont sa modélisation (la représentation du monde réel à travers les relations, les dépendances fonctionnelles) représentera l'information (Paredaens et al., 1989), (Cotelea, 2012).

Afin de déterminer si une base de données est bien modélisée, elle doit fournir et maintenir l'exactitude des données qu'elle contient (Cotelea, 2012). Tester la validité de cette modélisation repose sur la compréhension des dépendances fonctionnelles (Cotelea, 2012). La seule façon de déterminer si une dépendance fonctionnelle est valide est de passer par une analyse détaillée de la signification de chacun de ses attributs ainsi que des valeurs qu'ils peuvent prendre (Cotelea, 2012). Pour connaitre la signification d'un attribut et de ses valeurs possibles, deux sources d'information sont requises: un expert ayant la connaissance des détails relationnels de la base de données et la documentation (Reda, 2003). Dans le cas où l'une ou l'autre, voire ces deux sources, s'avèrent absentes, l'option de la réingénierie à rebours devient nécessaire afin d'identifier et d'élucider les composantes et relations entre les entités d'affaires (Reda, 2003). De plus, cette étape, la réingénierie à rebours, malgré le temps qu'elle nécessitera pour pallier au manque d'information, s'avère tout de même une étape pouvant aider à soutenir l'amélioration des fonctionnalités et de la performance de ce qui est analysé sur le long terme (Reda, 2003).

Suite à cette analyse, les dépendances fonctionnelles identifiées serviront, dans une étape ultérieure, à normaliser la base de données (Cotelea, 2012). Maîtriser la compréhension des concepts de la normalisation est un facteur déterminant dans le succès d'une bonne modélisation (Thompson, Sward, 2005). Le développement d'un nouveau modèle devrait donc s'axer sur une modélisation complètement normalisée (Schraml, 2014). La dénormalisation est un outil qui doit demeurer pour des circonstances vraiment très particulières et vient uniquement après que la modélisation de la base de données ait été prouvée efficace (ici normalisée) (Carpenter, 2008).

« *it is simply sad and sometimes even silly to see designs denormalized as a matter of course* » (Schraml, 2014) p.1

Lorsque la modélisation est laissée aux développeurs, plusieurs d'entre eux vont dénormaliser une base de données avant même toute problématique, s'il y en a même une (Schraml, 2014), ce qui est contraire à la bonne approche, celle normalisée. Plusieurs de ces méthodes de modélisations utilisées proviennent de solutions empiriques non soutenues par une base scientifique ou une discipline d'ingénierie (Cotelea, 2012). Une modélisation utilisant des méthodes *ad hoc*, la plupart du temps, produit un schéma inflexible qui ne soutiendra pas adéquatement les exigences d'affaires (Cotelea, 2012).

Pour établir cette base de la modélisation, la problématique se résume donc à deux éléments (Cotelea, 2012) :

1) Quelles sont les relations que devrait comporter la base de données?
2) Quels sont les attributs de chaque relation?

L'identification des relations et des attributs du domaine d'affaires s'appuie sur les diverses techniques connues de réingénierie générale à rebours. Ce qui est davantage intéressant est de pouvoir comprendre l'historique et l'évolution d'un système complexe, car ceci devient un atout de taille pour autant s'informer de son état actuel que pour envisager des initiatives futures (Cleve et al., 2015). Toutefois, cet historique n'est pas toujours accessible. Dans le cas du présent projet, une partie de l'historique d'affaires provenait du directeur du commerce numérique, mais celui de l'historique technique, lui, était très faiblement représenté.

2.3.2 Diagramme d'entité relations

Afin de visualiser et de rendre partageable l'information extraite sur les relations, un ERD (*diagramme d'entité relations*) devrait être produit. Cette technique est la plus commune pour les bases de données relationnelles (Thompson, Sward, 2005), (Karwin, 2010). Elle sert de point d'ancrage pour discuter des enjeux liés aux exigences et déterminer si le modèle répond aux besoins attendus (Thompson, Sward, 2005), (Chilton, 2006). Il existe des notations plus récentes telles qu'UML, mais celle-ci par exemple, incorpore les processus d'affaires dans le

diagramme, ce qui n'est pas toujours souhaitable lorsqu'il s'agit de conserver une indépendance des données face à l'applicatif (Chilton, 2006).

Pour qu'un modèle d'entités relations soit convivial et facile à maintenir, il est préférable de choisir une notation graphique concise (De Lucia et al., 2010). Cette notation se retrouve sous forme de surlignage, jeu de minuscules / majuscules et ordonnancement des attributs. En guise d'exemple de notation, les clés primaires sont surlignées, les attributs calculés en minuscule, les relations ainsi que leurs cardinalités sont identifiées (Thompson, Sward, 2005). Dans l'ERD, afin de rendre l'information accessible aisément, la cohérence est d'une importance primordiale (Thompson, Sward, 2005). La cohérence est aussi simple que le choix des noms puisque ceux-ci deviendront, à partir de l'ERD, ceux qui seront utilisés dans la base de données finale (Thompson, Sward, 2005).

Les recherches de réingénierie à rebours se basent sur des techniques qui présument que la sémantique des clés primaires, étrangères et des attributs est complète ce qui n'est pas toujours le cas (Yeh et al., 2008). Devant une base de données dont les données descriptives « *DDL* » sont incomplètes, voire parfois manquantes, la technique préconisée, lorsque possible, est de passer à travers les interfaces pour y générer des valeurs et détecter où celles-ci sont persistées dans la base de données. Cette technique permet de recouvrer les informations manquantes et de pouvoir générer un ERD (Yeh et al., 2008).

Pour obtenir un modèle de qualité, l'indépendance des données par rapport à l'applicatif est nécessaire. Ne pas obtenir ceci force les programmeurs des autres couches à manipuler les données d'une façon qu'ils n'auraient pas à le faire, ce qui (*l'indépendance des données*) leur fait sauver du temps (Chilton, 2006).

Puisque les relations des entités sont au cœur d'un modèle de base de données de qualité (Kesh, 1995), un ERD se doit lui aussi d'être de qualité (Kesh, 1995). De plus, le processus d'élicitation des connaissances des experts pour obtenir ce qui est nécessaire afin de formuler les entités relations est davantage un art qu'une science (Kesh, 1995). Bien qu'un article (Kesh, 1995) propose une formulation mathématique afin de calculer la qualité, il indique aussi que la qualité du modèle demeure du domaine subjectif et donc, que sa valeur résultante est davantage un indicateur qu'une estimation précise (Kesh, 1995).

Pour mesurer la qualité d'un ERD, le résumé de l'article « *Evaluating the quality of entity relationship models* » (Kesh, 1995) est disponible à l'appendice B « *Résumé de la mesure de qualité d'un diagramme d'entité relations* ».

2.3.3 Performance

Une conception efficiente d'un ERD définit sa performance. L'efficience étant le lien entre nombre d'entités, relations et attributs avec les tâches que la base de données se doit de supporter.

Une fois cet ERD produit, la suite se scinde en deux opérations : la première, la transformation du diagramme en un format relationnel et la seconde sa normalisation (Teorey et al., 2002). Pour des personnes ayant de l'expérience avec des concepts orientés-objet ou de réseautique, la modélisation requiert une façon de penser différente (Thompson, Sward, 2005).

La transformation du ERD au format relationnel, cette première étape, est la transposition de la représentation des entités, attributs et relations contenus dans un ensemble de tables (Thompson, Sward, 2005).

En seconde étape, vient la normalisation de l'ERD. Afin d'obtenir une base de données performante, le facteur contribuant le plus à son degré de qualité revient aux techniques de normalisation des relations (Cotelea, 2012). L'importance de la normalisation a été prouvée à maintes reprises dans la littérature depuis des décennies (Wang et al., 2016). Elle offre plusieurs avantages qui seront discutés ci-après.

2.3.4 Normalisation

La normalisation est une technique afin de rendre optimale la modélisation des relations. Elle est basée sur l'observation du meilleur regroupement des entités entre elles (McMurdo, 1982) (ce qui implique qu'il peut exister plusieurs regroupements possibles). Elle offre divers avantages tels que l'élimination de la redondance et de l'incohérence des données, l'élimination des anomalies lors de manipulations, l'optimisation de l'espace de stockage

nécessaire et facilite la maintenance et la recherche de données (Wang et al., 2016). Plusieurs avantages additionnels gravitent autour de la normalisation tels que définis par (McMurdo, 1982):

- Facilité d'utilisation des données;
- Flexibilité au niveau des opérations de jointure;
- Précision quant aux liens dans la représentation logique;
- Facilité d'implémentation au niveau de la gestion de la sécurité;
- Facilité pour les SGBD à traiter l'information;
- Facilité d'utilisation du langage de manipulation des données afin de les extraire;
- Clarté des relations;
- Indépendance des données face aux nouveaux ajouts de fonctionnalités limitant les impacts de modifications logicielles.

Un point névralgique, renchéri par (Carpenter, 2008):

- Le ***maximum de flexibilité*** (pour l'extension du modèle) est atteignable

La clarté des relations et l'expressivité qu'elles procurent offrent une représentation améliorée du monde réel (Carpenter, 2008).

Maintenant, qu'en est-il de la normalisation des données au niveau de la performance? Tout d'abord, la normalisation facilite les SGBD relationnels à traiter l'information (McMurdo, 1982). Il est donc logique de formuler que de ne pas normaliser adéquatement revient alors à alourdir la tâche à l'engin d'optimisation et conséquemment, à obtenir une moindre performance.

La littérature mentionne que la normalisation, soit accéder à des formes normales supérieures, alourdira les opérations de sélection des données (Cotelea, 2012), (Lee, 1995). Il n'est pas indiqué comme tel dans la littérature, mais il est logique de mentionner que les avantages / désavantages d'une technique (la normalisation versus la dénormalisation) sont les inverses de l'autre. Dénormaliser une ou des relations pour tenter de pallier un problème, principalement pour la performance (Lee, 1995), ne résulte qu'en surcharge additionnelle de modélisation pour le reste de toutes les autres relations (Lee, 1995).

Quelques études ont examiné cet aspect, la performance des jointures en normalisation versus la dénormalisation par rapport aux bénéfices connexes, et en sont venues aux conclusions suivantes :

- Réduction du nombre de jointures nécessaires dans la majorité des requêtes (Panchenko, 2012b);
- Facilite le réglage minutieux de la performance (Panchenko, 2012b);
- Coût plus élevé pour les autres requêtes non affectées par la dénormalisation (Karwin, 2010).

De plus, malgré les inconvénients de la normalisation, la littérature mentionne en premier plan qu'il est recommandé de normaliser les relations (certains auteurs suggérant de normaliser au plus haut niveau possible) (Schraml, 2014), (Fagin, 2002), (Panchenko, 2012a), (Date, Fagin, 2002), (Karwin, 2010), (Carpenter, 2008) ou que pour obtenir une base de données performante, le rôle de la normalisation est primordial (Cotelea, 2012).

« *The higher the normal form, the more desirable it is.* » (Date, Fagin, 2002) p.467

« *It is better to remove as many future anomalies as possible first.* » (Carpenter, 2008) p.382

Finalement, il existe un autre avantage non négligeable lorsqu'une entreprise envisage une croissance ou expansion. Cet avantage est de pouvoir distribuer les entités à travers plusieurs bases de données (Paredaens et al., 1989) et donc, par le fait même, user de plusieurs machines au lieu d'augmenter la puissance d'une seule et unique machine.

2.3.5 La forme normale DKNF

Il existe une 6e forme normale. Puisqu'il est recommandé de normaliser au plus haut niveau (tel que cité dans la section précédente), pourquoi ne pas d'emblée choisir cette forme normale? La 6e forme normale s'applique pour les enregistrements ayant besoin de soutenir un historique des changements (Karwin, 2010) comme c'est le cas pour les entrepôts de données qui bénéficieront de cette forme normale (Carpenter, 2008). Si cette fonctionnalité d'un historique n'est pas justifiée, alors la 6e forme normale n'est pas utile (Carpenter, 2008). Advenant un changement d'orientation d'affaires et que celle-ci devient une nouvelle réalité, cette 6e forme normale pourra être implémentée ultérieurement (Carpenter, 2008).

La raison principale d'adopter la forme normale « *Domain-Key normal form* » (DKNF) est pour atteindre la plus grande flexibilité de modification (Panchenko, 2012a), (Panchenko, 2012b). Elle offre la meilleure stratégie de qualité sur le long terme pour réduire les coûts en maintenance lors du développement logiciel (Panchenko, 2012a), réduire les impacts sur l'applicatif lors de changements (réduction de coût en maintenance à son tour) (McMurdo, 1982) tout en profitant des avantages liés aux cinq premières formes normales (Fagin, 2002).

Examinons ce que signifie l'acronyme DKNF. De prime abord, dans ce contexte, il est nécessaire de déterminer de quel « *Domain* » il est question. Dans la théorie des bases de données relationnelles, le mot « *Domain* » peut référer à deux concepts différents (Jonsson, 1992) : celui du type de données, un entier, une chaîne de caractère ou bien celui de la sémantique d'un type de donnée, le sens d'un attribut, ex. : « Nom du commerçant » (Jonsson, 1992). C'est du second dont il est question.

Un livre complet s'évertue à expliquer comment adapter et aligner le code source applicatif avec le domaine d'affaires en développement logiciel : « *Domain Driven Design: Tackling Complexity in the Heart of Business Software* » (Evans, 2003). Pourquoi alors cette approche ne serait pas aussi bénéfique pour les bases de données? Ne pas utiliser DKNF reviendrait donc à formuler des dépendances parfois en dehors de la mécanique stricte prévue par le domaine d'affaires. Ceci pourrait alors conduire à des distorsions inattendues entre les interactions du domaine d'affaires et la logique relationnelle implémentée pour un état quelconque. À l'inverse, un état précis de la base de données pourrait ne pas représenter un état formel de la logique d'affaires si les relations ne sont pas en DKNF.

> « *Another desirable feature is that distinct database states have distinct meanings as defined in terms of formal interpretations* » (Jonsson, 1992) p.89

La forme normale DKNF s'appuyant sur la cinquième forme normale, se définit davantage en termes de domaines et de clés liés au domaine d'affaires pour représenter ses relations (Fagin, 2002). En résumé, toute contrainte du domaine d'affaires devrait être implémentée par des relations. Il ne doit pas y avoir de relations possibles pour tout sous-ensemble donné entre deux attributs non-clé à travers les tables (Karwin, 2010).

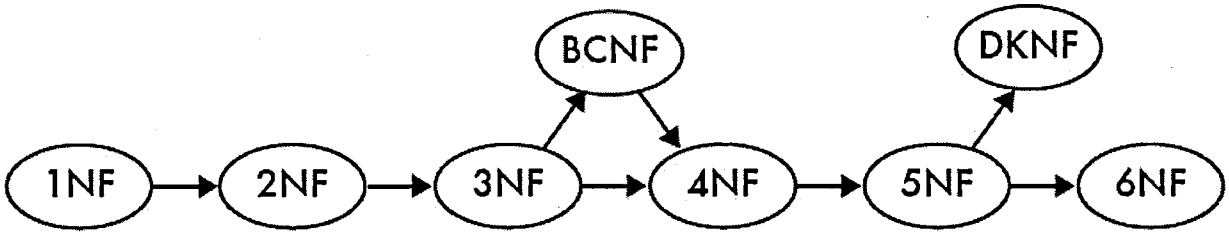


Figure 2-1 - Hiérarchie des formes normales

Tiré du livre: SQL Antipatterns: Avoiding the Pitfalls of Database Programming, P. 298 (Karwin, 2010)

L'idée de cette forme normale, DKNF, provient du fait de tenir compte des exigences fonctionnelles (Panchenko, 2012b) qui bornent le domaine (Fagin, 2002) lors du choix des dépendances fonctionnelles au lieu de la subjectivité du modélisateur (Panchenko, 2012b). La façon traditionnelle de modéliser, qui ne s'en tient qu'aux dépendances fonctionnelles au niveau mathématique entre les entités et les attributs, peut facilement générer trop de relations et en retour, les évaluer toutes demanderait trop de temps ce qui freine le modélisateur (Panchenko, 2012b). Pour sauver du temps, il en résulte souvent que le niveau de normalisation n'atteint alors que la troisième forme normale (Panchenko, 2012b). Toutefois, ce n'est pas une sage décision, pour une entreprise, d'aller en deçà de la 4^{e} forme normale au minimum (Carpenter, 2008) malgré le fait que ce soit une pratique courante que de simplement arrêter à la 3^{e} (Lee, 1995). Toute anomalie non gérée au niveau de la base de données devra être gérée éventuellement. Ceci impacte donc une ou d'autres couches applicatives et ralentit les développeurs à cause de manipulations additionnelles nécessaires des données (Chilton, 2006). Par conséquent, ces manipulations additionnelles pourraient aussi affecter négativement la performance applicative.

Un schéma usant de la forme normale DKNF, ne contient aucune surcharge additionnelle afin de soutenir les contraintes relationnelles (Fagin, 2002). En d'autres mots, DKNF offre le schéma minimal optimal de la représentation des relations du domaine. Cette approche minimise les opérations de jointure à la grandeur de la modélisation, ombrageant le principal avantage de la dénormalisation, la réduction des jointures, en éliminant toutes les relations qui ne représentent pas le domaine d'affaires. Ceci simplifie considérablement l'ajustement de la performance de la base de données (Panchenko, 2012b). Empiriquement, la performance d'accès aux données et aux modifications au schéma d'une BD sont significativement améliorées (Panchenko, 2012a). Une modélisation de la sorte, lorsqu'elle

est effectuée, peut même amener à revoir la sémantique des dépendances entre les entités du domaine d'affaires et soutenir son optimisation à son tour (Panchenko, 2012b).

Selon (Fagin, 2002), tout modélisateur devrait viser à ce que toutes les relations d'une base de données puissent être sous la forme DKNF. Cela permet alors de répondre à la question des analystes, à savoir quelle est la forme normale appropriée à utiliser (Lee, 1995). Donc quelle est la procédure pour obtenir une base de données en forme normale DKNF? Les algorithmes proposés par divers auteurs ne vont pas au-delà de la troisième forme normale ou celle améliorée de Boyce-Codd (Diederich, Milton, 2002), (Köhler, Link, 2018), (Omiecinski, 1990). Un article va jusqu'à mentionner que la forme normale de Boyce-Codd est celle la plus forte qui est atteignable à l'aide des dépendances fonctionnelles seulement (Springsteel, 2003), ce qui est nécessaire, mais non suffisant pour atteindre DKNF.

« *In this paper we focus on BCNF, the strongest normal form of relation schemes that is defined in terms of FDs alone.* » (Springsteel, 2003) p.581

La littérature n'offre que ceci au final : la modélisation d'un bon schéma de base de données est vue davantage comme un art qu'une science exacte (Cotelea, 2012), (Panchenko, 2012b), (Panchenko, 2012a). Il est donc raisonnable que la tâche revienne au modélisateur de bien maîtriser le domaine d'affaires afin d'orienter sa modélisation à la forme normale DKNF.

« *It can be concluded that designing a good database schema is more an art than a science.* » (Cotelea, 2012) p.23

Pour une vulgarisation, des exemples et de plus amples informations techniques à propos de la normalisation, le livre « *SQL Antipatterns – Avoiding the Pitfalls of Database Programming* » couvre le sujet. De plus, ce livre démontre un cas n'atteignant pas la forme normale DKNF.

2.4 Processus de gestion de projet, ISO 29110

L'utilisation de la norme 29110 est recommandée comme point de départ pour améliorer la maturité de son processus de développement (Mesquida, Mas, 2014). Cette norme concerne l'ingénierie des processus de cycle de vie pour les organismes de très petite taille (TPO) pour le développement de logiciels. Son utilisation a pour buts finaux d'améliorer la qualité produite par l'amélioration du processus, de promouvoir les bonnes pratiques (Galvan et al., 2015) et d'uniformiser la documentation (Laporte et al., 2017). Majoritairement, le travail à reprendre se situerait entre 40 et 50 % chez les organisations n'utilisant pas la norme, contrairement de 10 à 18 % pour celles qui utilisent la norme (Laporte, O'Connor, 2016).

Toutefois, les bases de données ont certaines caractéristiques qui leur sont propres par rapport à un logiciel, autant par son approche que par ses outils (Shmeltzer, 2018) (plus de détails sur ce sujet dans la section « *Évolution de la nouvelle base de données des produits* »). La question se pose, « *Est-ce que la norme, définie pour du développement logiciel, est applicable à un projet de base de données?* ». Selon un rapport (Takeuchi et al., 2014), la norme ainsi que ses bienfaits sont transposables à différents domaines, et donc ici, indirectement aux bases de données. Finalement, cette norme a fait ses preuves dans plusieurs pays, dont le Canada (Laporte, O'Connor, 2016).

Un récapitulatif des grandes lignes de la norme ISO 29110 se retrouve à l'appendice A « *Norme ISO 29110* ».

CHAPITRE 3

MÉTHODOLOGIE POUR LA PREUVE DE CONCEPT ÉVOLUTIVE

Afin d'assurer la satisfaction des parties impliquées ainsi que d'éliminer la perte de productivité, il était crucial d'établir quel serait le processus à mettre en place. Le processus est en majeure partie responsable de la qualité du résultat (Trudel et al., 2006). La ligne d'affaires ne possède pas de certification ISO ou CMMI, mais utilise son propre processus de développement logiciel. Toutefois, ce processus est conçu pour des équipes et l'ensemble d'un produit logiciel et non une seule personne ciblant une composante, ici la base de données.

J'ai donc choisi d'utiliser la norme ISO 29110 pour la gestion et le développement du projet. Étant surchargé avec les tâches quotidiennes, le bienfait cité de la réduction du temps de développement provenant de l'utilisation de la norme ISO 29110 (Laporte et al., 2017) a été un facteur additionnel dans sa sélection. De plus, la norme ISO 29110 s'applique aux très petits organismes de 1 à 25 personnes (Organisation internationale de Normalisation, 2012) et pouvait donc cadrer pour une seule personne, en l'occurrence moi (en dehors des approbations et de l'assurance qualité), pour la taille de ce projet.

Selon les profils disponibles de la norme ISO 29110, celui le plus adéquat pour le projet actuel était celui de base qui incorpore deux processus : un de gestion de projet et un de mise en œuvre de logiciels (Organisation internationale de Normalisation, 2012). La liste des activités des deux volets de la norme est disponible en appendice A sous « *Norme ISO 29110* ». C'était sur cette base, agissant en toile de fond, que la méthodologie suivante pour la mise en œuvre de ce projet s'était appuyée :

Méthodologie pour la preuve de concept évolutive				
Phase	**Intrants**	**Activités**	**Extrants**	**Retombées**
Planification	Besoins d'affaires	Définir le projet (3.1.1)	Énoncé des travaux (Annexe V, Plan de projet, p. 2-5)	
	Énoncé des travaux	Rechercher des solutions (3.1.2)	Solutions envisagées, mais non retenues (4.1.1) Solution retenue (4.1.2)	
	- Énoncé des travaux - Solution retenue	Planifier le projet (3.1.3)	Liste détaillée des tâches (4.2.1)	
	Plan de projet	Présenter le plan (3.1.4)	Plan révisé / approuvé (4.2.2)	
Réalisation	Plan de projet et Solution retenue	Modéliser la preuve de concept (3.2.1)	Modélisations (4.1.3) - Données d'affaires - Normalisation	Redécouverte complète des relations d'affaires (6.1)
	Modélisations	Créer la base de données (3.2.2)	Base de données (4.1.4)	
	Base de données	Migrer les données (3.2.3)	Bases de données migrées (4.1.5)	
	Bases de données actuelle et nouvelle	Mesurer les écarts (3.2.4)	Résultats des mesures (4.1.6)	Réduction anticipée des coûts (6.2)
Contrôle	Progression des travaux	Mettre à jour le plan des tâches (3.3.1)	Mise à jour du suivi de progression (4.2.3.1)	
	Demande de changement (DDC)	Analyser et évaluer la DDC (3.3.2)	DDC approuvée ou rejetée et plan modifié (4.2.3.2)	Incohérences de traitement de la BD actuelle (6.3)
Clôture	Résultats des mesures	Présentation des résultats (3.4.1)	Poursuite ou non du projet dans l'entreprise (4.2.4)	

Figure 3-1 - Aperçu de la méthodologie pour concevoir et développer la preuve de concept de la nouvelle BD.

3.1 Planification

3.1.1 Définir le projet

Au lancement du projet, les objectifs m'ont été transmis de façon informelle, verbalement, à l'initiative du président de la ligne d'affaires. Il a énoncé le requis technique principal, soit la réduction de la charge de CPU, et a mandaté son directeur du commerce numérique pour prendre la relève concernant les besoins d'affaires fonctionnels. Puisque la quantification de la réduction du CPU n'était pas connue à l'avance, une phase de mesure était donc nécessaire. Cette mesure est passée par une preuve de concept évolutive qui fait office de ce présent projet.

Aucun autre membre du personnel (autre que les approbations et l'assurance qualité) n'avait été attitré au projet. J'ai donc dû entreprendre seul tous les autres rôles et toutes les tâches. J'ai aussi dû combler les lacunes de la définition du projet de ce qui était attendu (voire la *sous-section 1.4.1* pour explications sur les exigences non fonctionnelles). Fort heureusement, le domaine d'affaires était bien connu de toutes les parties impliquées, le projet interne était bien circonscrit d'un point de vue d'affaires et je connaissais quelques rouages de la mécanique du processus d'affaires. La charge liée à la phase de l'ingénierie des exigences et les problèmes d'interprétation des travaux à effectuer en étaient largement diminués. Il s'agit d'un avantage considérable lorsque les erreurs provenant des activités des exigences représentent 40 à 50 % des défauts (Wiegers, Beatty, 2013).

Par la suite, l'introduction du présent document a été utilisée à deux fins précises. D'abord, s'assurer que les objectifs inscrits étaient ceux souhaités par la direction. Ensuite, s'assurer que ce qui serait transmis académiquement ne présente pas d'information sensible pour la ligne d'affaires. À partir de ces deux considérations, j'ai personnellement initié une rencontre avec le directeur du commerce numérique afin de valider et confirmer ma compréhension des besoins fonctionnels. Cette confirmation écrite est disponible dans le courriel « *Approbation initiale écrite* » à l'annexe IV « *Liste des noms de documents produits* » sous-section « *Documents intermédiaires* ». Cette étape pouvait sembler superflue, mais s'assurer d'avoir les bonnes exigences dès le début est pratiquement toujours rentable.

« *investing in good requirements will virtually always return more than it costs.* » (Wiegers, Beatty, 2013) p.22

La vision, les objectifs, la portée ainsi que les livrables de très haut niveau ont été inscrits dans l'énoncé des travaux qui est un sous-ensemble du document des plans en page 2 à 5 en annexe V « *Plan de projet* ». Ces informations correspondent à la tâche PM 1.1 de ISO 29110. Cette version, dans le document des « Plans », est disponible à l'annexe V « *Plan de projet* ».

3.1.2 Rechercher des solutions

Une base de données est un artefact qui est plus difficile à modifier (Ambler, Sadalage, 2006) et tout ce qui est plus complexe est moins souvent changé (Tempero et al., 2017). Il est donc envisageable que la base de données des produits ne soit pas souvent sujette à du remaniement de code. C'est pour cette raison qu'évaluer diverses alternatives de modélisation avant d'entreprendre un quelconque travail d'implémentation était justifié. Pour plus de détail sur ces solutions alternatives, voire l'annexe VI « *Autres solutions envisagées* ».

3.1.3 Planifier le projet

Une fois mon choix de reconceptualiser déterminé, venait le temps de produire et de raffiner le reste de la planification du projet. La vision d'affaires autant que technique étaient en place. J'étais maintenant en mesure de définir les tâches à effectuer. Un document Word© à l'entête de la compagnie ainsi qu'un fichier MS-Project© ont été utilisés pour la planification et le détail des tâches. Dans la planification en gestion de projet, deux grandes questions s'imposent :

1) Combien ça coûte?
2) Quand est-ce que ce sera prêt?

Les coûts n'étaient pas gérés au niveau du projet selon un concept de budget. Ils étaient implicites en termes de salaire pour le nombre d'heures de travail versus le travail potentiel non effectué au détriment d'autres projets, soit le « coût de renonciation ». La durée estimée, bien que le document des tâches le comptabilisât automatiquement, n'a pas été utilisée. Le manque de données (mesures) antérieures sur l'élaboration de la base de données des

produits initial, mon imprédictibilité à devoir gérer les feux en production et les divers projets parallèles nécessitant mon intervention, rendaient à la donnée du calendrier une estimation floue, voire presque inutile. À défaut de données historiques, toutes les estimations ont été basées sur mon expérience professionnelle uniquement.

Ces nouveaux ajouts correspondent aux sections PM 1.2 à PM 1.12 de la gestion de projet ainsi que partiels pour SI 2.2 de la mise en œuvre d'ISO 29110.

3.1.3.1 Mitigation des risques

Un des risques encourus, le plus probable soulevé dans le document des plans en page 4 en annexe V « *Plan de projet* », et qui s'était réalisé, était celui de l'indisponibilité de l'unique ressource – moi – afin de mener à bien le présent projet. Mon poste m'amène à être une ressource de type « *partagée* » à travers tous les départements : le produit, le service, les technologies de l'information « *TI* » et le soutien technique.

Impliquer activement une personne dans tous les secteurs techniques d'une compagnie signifie que son temps de travail est divisé selon les criticités. C'est à la fois un problème (de disponibilité) mais aussi deux avantages considérables. Puisque j'interagis et travaille directement avec toutes les sphères techniques, les deux facteurs problématiques cruciaux suivants en étaient grandement réduits. Le premier, la communication inter équipes, souvent à l'origine de la détérioration de la qualité, voire simplement de l'échec de projets (Defranco, Laplante, 2017). Le second, la coordination d'activités, contributeur majeur dans les crises et échecs de projets logiciels (Kraut, Streeter, 1995). Ces avantages ne sont pas marginaux par le fait que la ligne d'affaires tente d'améliorer sa communication à l'interne et que la nouvelle base de données aura un impact sur toutes les divisions de celle-ci.

Toutes les options de mitigation dans le document du *plan de projet* en annexe V avaient été mises en place. Au début du projet, un incident critique a nécessité mon intervention journalière pendant quelques mois, et ce, exclusivement pour un seul client. De ce fait, tout autre projet était reconduit à la stagnation. Constatant l'absence de progrès général et considérant l'incident en cours, le président avait milité envers la compagnie mère en faveur

d'un nouvel employé : recruter un nouveau DBA afin de me délester des tâches opérationnelles.

Ce nouveau DBA, une fois recruté, était sous la supervision hiérarchique du directeur des TI mais opérationnellement sous ma direction. En moyenne, un professionnel nécessitera vingt semaines avant d'être productif (Rollag et al., 2005), une période de formation pendant laquelle je n'étais pas encore libre des tâches opérationnelles.

Je me suis chargé d'accompagner le nouveau DBA à toutes les réunions formelles et informelles. Je me suis assuré aussi qu'il soit présenté à toutes les personnes ressources afin d'établir et d'étendre son réseau de collègues le plus rapidement possible afin de le rendre le plus productif en un temps minimal.

« *The key to making new employees productive quickly, known as "rapid on-boarding," is to help them immediately build an informational network with co-workers.* » (Rollag et al., 2005)

Malheureusement, quelques mois après son embauche, ce nouveau DBA tomba en congé de maladie prolongé. De ce fait, tout le temps investi en formation n'a pas permis d'être récupéré sur le long terme pour me libérer aux fins de ce projet, ce qui accentua donc la pression sur le temps supplémentaire.

3.1.3.2 Environnement de mise en œuvre

La norme ISO 29110 prévoit une activité faisant l'objet de l'environnement de la mise en œuvre. Pour ce projet, rien d'additionnel n'était nécessaire puisque le matériel et les logiciels disponibles étaient déjà suffisants pour l'accomplissement de la nouvelle base de données. Ceci correspond à la section SI 1.2 de la mise en œuvre d'ISO 29110.

3.1.3.3 Copies de sécurité

Une base de données se décompose en objets, tables, vues et procédures stockées qui ont tous intérêt à résider dans un gestionnaire de versions (« source control »). Toutefois, prendre une base de données et la décomposer pour obtenir un fichier par objet n'est pas aisé.

Plusieurs solutions existent sur le marché, mais elles sont soit coûteuses ou sinon ne correspondent pas aux attentes. De plus, j'ai essuyé un refus à deux reprises pour obtenir que toutes les bases de données (pas juste celle des produits) soient versionnées, mais en vain.

Pour pallier ces deux problèmes ainsi que les défaillances matérielles, j'ai opté pour des copies de sauvegardes qui étaient effectuées chaque soir dans un répertoire sur le réseau de la ligne d'affaires. Ce répertoire réseau fait partie des sauvegardes générales des données de la ligne d'affaires au grand complet et s'appuyait donc sur un processus fonctionnel et éprouvé en place depuis longtemps.

Même si ce n'est pas ce qui était le meilleur, c'était mieux que rien et ces étapes correspondent aux activités PM 2.5 et 2.6 de ISO 29110.

3.1.3.4 Traçabilité des exigences

Pour la traçabilité des exigences, puisqu'il y en a peu (d'exigences), j'ai utilisé un jeu de couleurs afin de les identifier facilement dans les documents de conception et j'ai utilisé le texte de la couleur dans les scripts des tests d'intégration. Chaque exigence avait sa propre couleur et ces couleurs sont définies dans le document des plans en page 1 en annexe V « *Plan de projet* ». Leurs vérifications étaient assez simples, il suffisait de vérifier que :

- la base de données contenait les objets identifiés;
- les documents de conception coloriaient les objets liés aux exigences;
- la base de données contenait les champs demandés dans la conception;
- le script de migration indiquait à quel endroit le code faisait état des nouvelles exigences;
- les scripts des tests d'intégration s'exécutaient sans erreur.

La traçabilité des exigences représente les activités SI 3.7, SI 4.6 et SI 5.6 de ISO 29110.

3.1.4 Présenter le plan

J'ai tenu une réunion informelle, supportée par une présentation PowerPoint©, disponible en annexe VII sous « *Présentation PowerPoint du projet* », au directeur du commerce numérique pour discuter des points suivants :

- Objectifs de la nouvelle base de données;
- Portée académique et corporative;
- Mesures de performance;
- Impacts sur l'applicatif;
- Documents et suivi de progression.

J'ai aussi présenté des composantes du projet telles que le document Word© du « *Plan de projet* » (en annexe V) ainsi que le fichier MS-Project© du détail des tâches « *Détail des tâches* » (en annexe VIII) pour illustrer comment il pouvait obtenir une visibilité en continu sur les détails et l'avancement du projet. En définissant clairement les balises de la preuve de concept évolutive, ces documents ont servi aussi à contenir une éventuelle escalade des exigences.

> « *The first step in managing scope creep is to document the business objectives, product vision, project scope, and limitations of the new system.* » (Wiegers, Beatty, 2013) p.473

De plus, j'ai fait mention d'un auditeur afin d'accroître l'objectivité et la validité des résultats, un point qui a été très favorablement accueilli aux yeux du directeur.

À la fin de la présentation, le directeur du commerce numérique a mentionné « *Le pourcentage d'amélioration que tu vas nous donner, c'est le pourcentage que je vais te donner pour la note de ton projet.* ». Bien qu'humoristique, cette anecdote sous-entend implicitement que la meilleure attente serait une amélioration de 100 % (donc, doubler la performance actuelle ou à l'inverse que ce soit à moitié moins lourd).

Une fois le plan présenté, il avait été rendu disponible à son emplacement désigné dans le document des plans. Ces tâches correspondent aux sections PM 1.13 à 1.15 de la gestion de projet d'ISO 29110.

3.1.4.1 Assignation des rôles

La norme ISO 29110 définit sept rôles différents pendant le déroulement d'un projet soit le client, l'analyste, le concepteur, le programmeur, le gestionnaire de projet, le chef technique et l'équipe de travail. Dans le contexte actuel, j'ai exercé la totalité des rôles, excepté celui du client.

Le rôle du client était conjointement interprété par le directeur du commerce numérique (majoritairement) et le président. Le président, ayant un historique professionnel technique, s'intéresse et s'implique dans la partie technique de la plateforme.

Comme mentionné initialement, j'ai effectué tous les rôles à l'exception des approbations et de l'assurance qualité. Pour cette dernière, j'avais introduit un nouveau rôle, celui d'auditeur, afin d'accroître la validité des résultats des mesures et valider l'alignement de ce qui serait mesuré. La norme ISO 29110 ne mentionne pas ce rôle particulièrement parce qu'il est implicitement défini un peu par tous les rôles (approbations, vérifications et tests), rôles dont je suis l'unique exécuteur et que je n'avais pas la possibilité de discuter avec des pairs.

Parce que j'ai interprété tous les rôles techniques, toute première tâche de chacune des sections SI entre SI 1.1 et SI 6.1 était implicitement accomplie.

3.2 Réalisation

3.2.1 Modéliser la preuve de concept évolutive

3.2.1.1 Modélisation d'affaires

Suite au départ massif du personnel technique, la connaissance technique des relations entre les attributs des produits était mise à rude épreuve. Fort heureusement, puisque les deux bases de données des produits, celle actuelle et la nouvelle, devaient délivrer exactement le même résultat au final, le travail d'analyse a été un peu simplifié.

Des deux sources primordiales d'information décrites dans la sous-section « *Modélisation de la nouvelle base de données* » du sous chapitre 2.3.1, seul le directeur du commerce numérique, en dehors de la réingénierie à rebours et de l'analyse de l'interface utilisateur, était en mesure de pouvoir répondre aux diverses questions sur les relations d'affaires des attributs. Par contre, personne n'était réellement en mesure d'expliquer les détails de leurs implémentations. La documentation de la plateforme existait, mais elle n'était pas orientée pour une utilisation technique, encore moins pour résoudre des détails relationnels d'affaires (exemple de règle d'affaires : si j'ai un produit, suis-je obligé ou non d'avoir une variante pour celui-ci? (1 à N ou 0 à N variante(s) pour celui-ci?). Cette information manquante causait un préjudice à quiconque avait besoin de s'informer sur les relations d'affaires.

À l'aide du directeur du commerce numérique et d'une réingénierie à rebours de la base de données des produits, j'ai produit un diagramme minimal d'entités-relations, représentant les relations entre les concepts d'affaires de haut niveau. Ceci était la première étape dans la formulation de la compréhension de la mécanique des liens entre les données. La réingénierie à rebours a été nécessaire puisque la base de données actuelle comporte des lacunes au niveau relationnel ouvrant la porte aux erreurs de manipulation, de corruption des données, d'interprétation et donc de compréhension des relations d'affaires. Ces corruptions, par la suite, sont soit ignorées par le code applicatif, contournées ou tout simplement génèrent des erreurs dans l'interface client. Dans les trois cas, les effets de bord sont indésirables : perte de temps pour du code applicatif additionnel afin de gérer et manipuler des cas d'affaires improbables (donc davantage de codes à maintenir) ou sinon éroder la confiance de l'utilisateur vis-à-vis la fiabilité de la plateforme électronique lorsque celle-ci affiche une erreur.

J'ai basé les deux diagrammes sur la notation basique de *Crows Foot* puisque c'est l'unique notation disponible dans le logiciel de graphisme vectoriel que la ligne d'affaires possède et qui est donc disponible à tous les employés sans nécessiter de dérogation additionnelle. Dans les choix des options de la notation, celui de la conformité avec ce qui existe déjà dans la compagnie prime, mais pour le reste, les détails d'implémentation se sont basés sur la norme ISO 31320-2 (ISO/IEC/IEEE 31320-2:2012(E), 2012).

La modélisation d'affaires ainsi que son approbation, s'appuyant sur les ressources telles que la documentation actuelle, la réingénierie à rebours ainsi que le directeur du commerce

numérique, correspondent aux sections SI 2.2 à 2.4 de la mise en œuvre d'ISO 29110 (SI 2.5 à 2.7 étant optionnelles, si applicables).

3.2.1.2 Modélisation normalisée

Une fois le premier modèle complété, il était alors possible d'entreprendre avec confiance la normalisation de celui-ci. Le processus de normalisation est documenté à travers divers livres et articles. À titre d'exemple, cet ouvrage « *SQL Antipatterns, Avoiding the Pitfalls of Database Programming* » passe à travers toutes les formes normales (Karwin, 2010) à l'aide d'exemples concrets. Cette modélisation fait le pont entre le diagramme d'affaires et le domaine relationnel en explicitant les détails des relations.

C'est à ce stade que le nom des objets du SGBD était initié. La nomenclature étant plutôt importante, le code applicatif s'attendait à des noms de colonnes et de certaines tables déjà préétablis. Ce faisant, il ne restait que la nomenclature des tables additionnelles qui était possible (celle des entités était dérivée des noms d'affaires du diagramme précédent).

Dans cette modélisation, et c'était là l'importance de la DKNF, s'appuyant sur le domaine d'affaires, la séquence par laquelle les données étaient persistées devait épouser l'ordre naturel avec lequel le domaine d'affaires traite ces données à l'aide de la plateforme électronique. La ligne d'affaires a conclu des partenariats avec d'autres compagnies pour effectuer la personnalisation de la plateforme. Dans cette optique, les partenaires ont accès au code source de la plateforme, et l'un d'eux en a profité pour « l'adapter » à ses besoins. Puisque la base de données actuelle valide peu les données qu'elle accepte, selon son degré de normalisation actuel, celles-ci sont parfois incohérentes ce qui, plus loin, cause des problèmes non documentés au niveau de la plateforme. Il est estimé que jusqu'à 80 % du code d'un programme sert à traiter des cas qui n'existent pas au niveau des affaires (April, Laporte, 2011). Ce faisant, ceci alourdit le travail des développeurs et du soutien technique, car ils font face à des erreurs ou des comportements inconnus jusqu'alors. Des balises documentées existent pour aligner les partenaires à utiliser la plateforme adéquatement, mais le partenaire particulier de cet exemple-ci en avait fait fi aux frais de la ligne d'affaires qui doit dépenser du temps pour étudier le problème et conclure que les balises qu'elle a émises n'ont pas été respectées. Ceci a été éliminé par le fait que « *si la nouvelle base de données des*

produits accepte une valeur, c'est que ce nouvel ajout correspond à un état valide, cohérent pour la plateforme. L'inverse est aussi vrai, si elle refuse une valeur, c'est que l'état n'est pas valide. ». Un partenaire pourrait modifier la structure de la base de données pour davantage contourner les façons de faire, mais comme il s'agit d'un artefact de type « partagé », aucun des partenaires actuels, même celui le plus « *téméraire* », n'ose la modifier substantiellement. Ce serait à mesurer, mais il est envisageable que cette nouvelle modélisation ait pour effet additionnel de réduire la lourdeur de la documentation des balises, de réduire le nombre d'incidents du soutien technique et de soutenir les partenaires avec une réponse immédiate lors de la tentative de persister des données dans un état incohérent de la plateforme vis-à-vis des produits. Il est admis que tout problème connu le plus tôt possible coûte largement moins que dans les phases ultérieures de la vie d'un logiciel; l'en prévenir encore plus tôt dans le processus devrait donc davantage économiser en coûts de toutes sortes, à divers paliers et accroîtront de ce fait la compétitivité respective de la ligne d'affaires et de ses partenaires.

« *Identifier une erreur tôt dans le processus pourra sauver beaucoup d'effort, d'argent et de temps.* » (April, Laporte, 2011)

En dehors des entités qui ont été normalisées à la forme DKNF, la plateforme électronique offre la possibilité à ses clients de pouvoir ajouter, à la demande, des attributs autres que ceux distribués initialement. Cette flexibilité permet à la plateforme de mieux s'adapter aux besoins d'affaires particuliers de ses clients en acceptant la personnalisation d'affaires de ceux-ci vis-à-vis leurs propres produits. De cette exigence d'affaires, afin de pallier les problèmes engendrés par cette personnalisation, une forme statique de relations n'est pas en mesure d'y répondre adéquatement. Le modèle approché, pour répondre à cette exigence a été celui de l'EAV « *Entity-Attribute-Value* ».

« *Most applications that need schemaless data really need it for only a few tables or even just one table. The rest of your data requirements conform to standard table designs. If you account for the extra work and risk of EAV in your project plan, it may be the lesser evil to use it sparingly, But keep in mind that experienced database consultants report that systems using EAV become unwieldy within a year.* » (Karwin, 2010) p.81

La base de données originale avait employé ce principe d'utilisation du modèle EAV. Puisqu'il s'agissait d'un cas isolé des exigences d'affaires, que l'utilisation des attributs variables était circonscrite à l'engin externe de recherche, que seulement trois tables (les produits ainsi que deux autres concepts gravitant autour de ceux-ci et qui ont sensiblement la même exigence d'affaires) requièrent cette façon de faire, le modèle EAV est demeuré dans la nouvelle base de données. Toutefois, j'ai restreint davantage le domaine des valeurs acceptées pour empêcher, par exemple, d'insérer plus d'une valeur à un seul attribut ou d'attribuer des valeurs à des attributs qui ne font pas partie d'un produit valide. Ceci représentait des améliorations au regard de la base de données précédente.

La modélisation normalisée correspond en quelque sorte à la conception du logiciel tel que défini dans la norme. De ce parallèle, avec le modèle généré, ceci correspond aux sections SI 3.2 à 3.4 de la mise en œuvre d'ISO 29110.

3.2.2 Créer la base de données

La création de la base de données, à l'aide du modèle ERD normalisé et de *SQL Server Management Studio*© (SSMS), s'est penchée à ce stade sur des critères périphériques qui seront décrits plus loin. Le cœur du problème relationnel était déjà résolu par les ERD. Ce que j'avais à prévoir par contre était de la rendre davantage maintenable et facile à utiliser par les développeurs, le soutien technique ainsi que par le nouveau DBA côté opérationnel.

> « *Improve database readability. If you have a highly normalized database, it is usually difficult for users to navigate through all the tables to get to the required information.* »
> (Ambler, Sadalage, 2006) p.249

Pour pallier ce premier problème — améliorer la lisibilité — j'ai conçu des vues non matérialisées, accessibles pour tous types d'intervenants humains. Ces vues seront (dans la suite du projet) révoquées pour la partie applicative de la plateforme par l'entremise de la sécurité du SGBD afin de ne pas nuire à la performance. Ces vues préétablissent les jointures afin d'afficher des noms évocateurs au lieu d'identifiants numériques (pour la performance). Cet ajout facilite l'affichage des données tout en conservant l'intégrité ainsi que de ne pas avoir à dénormaliser les relations des tables de base. J'avais aussi produit des vues traduisant

le format des tables actuelles éliminées vers le nouveau format pour faciliter la compréhension de la manipulation de l'information dans la nouvelle base de données comparativement à sa version actuelle. Pour facilement identifier les tables éliminées, une étiquette a été introduite utilisant le nom actuel, par exemple « vw*Deprecated_NomAncienModèle* ».

Puisque la base de données actuelle possède plusieurs types de nomenclatures, c'était le moment opportun d'en sélectionner une seule et de l'appliquer partout. Le choix que j'ai retenu a été d'utiliser le « *Pascal Case* » ou son équivalent le « *Upper Camel Case* » qui met en majuscule la première lettre de chaque mot. Les outils de développement de la plateforme qui ajoutaient des tables (et que nous ne pouvions pas modifier) utilisaient déjà ce format. De plus, la lecture du nom d'un objet qui était uniquement en majuscule, parfois sans soulignement « *underscore* » à titre de séparateur, était lourd, difficile et demandait un court temps d'arrêt pour s'assurer de lire correctement le nom, un irritant.

Dans le modèle actuel, le nom du *scope*, qui représente aussi le nom du magasin, d'une région ou du siège social, fait partie du nom d'un jeu de tables (ex. : « [NomDuScope]_ITEM »). Certains clients utilisent des noms de magasins usant du « *Pascal Case* » et donc, nous retrouvons dans la base de données un amalgame de nomenclatures pour un même objet! De plus, cette façon de faire (déterminer partiellement le nom d'un objet en SQL lié à une configuration du domaine d'affaires) implique des limitations aux clients au niveau des noms possibles que les clients peuvent utiliser pour leur magasin. En éliminant le nom du *scope* dans le nom de la table, la limitation d'affaires venait elle aussi de disparaitre et permettait donc une meilleure flexibilité pour le client lors de l'attribution des noms de ses magasins.

Dans la version actuelle de la base de données des produits, il n'était pas aisément possible de déterminer les valeurs d'états valides qu'une référence pouvait prendre, par exemple, un état d'inventaire (Ex : « *Empty* », « *Full* », « *Available* »). L'équivalent du côté applicatif serait la commande « ENUM ». L'ENUM permet d'encadrer la liste des valeurs qui sont admissibles. Côté base de données, il n'y a pas d'équivalent en termes d'ENUM que le SGBD implémente « *out of the box* ». Ce qui avait été implémenté dans la version courante indiquait uniquement les valeurs en cours d'utilisation. Ceci implique que tant qu'une valeur n'est pas utilisée ou stockée, prenons par exemple « Empty », il ne sera jamais possible de déterminer que cette

valeur est disponible autrement que d'aller vérifier dans le code applicatif les valeurs qui y sont codées. Ceci implique aussi l'inverse, c'est-à-dire que la base de données actuelle accepte n'importe quelle valeur sans en vérifier la validité. Une contrainte aurait pu être ajoutée pour valider, mais la flexibilité de changement n'aurait pas été possible en plus de requérir à une modification DDL de la table. Dans un cas initial, lorsqu'une base de données est vide, il est impossible de déterminer la plage des valeurs possibles qu'un état pourrait prendre. La solution implémentée possède son propre schéma « ENUM » pour stocker ces valeurs permettant au soutien technique d'être indépendant dans ses manipulations, tout en validant les valeurs possibles, facilitant et encadrant l'altération des choix possibles puisqu'il ne s'agit que du DML.

Finalement, il me restait les tests à réaliser. Puisqu'aucune donnée, autre que celles de base étaient présentes, seulement quelques tests étaient possibles (avant la migration des données) :

- La vérification de la nomenclature des objets et de celles des noms de colonnes référençant une clé;
- L'opérationnalisation des vues de soutien;
- Un script pour tester l'effacement en cascade des données (pour éviter d'effacer des données de base qui doivent toujours être présentes).

« *Les tests servent en premier lieu de documentation par l'exemple.* » (Martin et al., 2009) p.188

Cette section comble les activités SI 4.2 à 4.7 d'ISO 29110. Chaque sous-activité définie dans le plan des tâches a son test correspondant. Les tests les plus critiques, les dépendances ainsi que les contraintes d'affaires font partie intégrante de la base de données sous forme de clés étrangères, clés uniques, clés primaires, contraintes de colonnes ainsi que des contraintes de tables. Puisque les contraintes s'appliquent en temps réel et que les scripts seront incorporés aux tests d'intégration ultérieurement, le code des tests eux-mêmes devient la documentation.

3.2.3 Migrer les données

Modifier les relations impliquait que les données devaient être elles aussi adaptées à ce nouvel univers et donc, il était nécessaire d'avoir une façon de transiger du format actuel vers le nouveau. Dans les livrables originaux, lors de la discussion informelle verbale avec le président de la ligne d'affaires, nous avions évoqué que la migration s'effectuerait à l'aide d'un unique script SQL. Le but visé avec ce script de migration des données était qu'il soit autonome et exécutable, peu importe le client. En contrepartie, utiliser plusieurs scripts aurait impliqué de conserver des états entre eux, ce que SQL ne permet pas aisément. Sachant que chacun des clients a ses particularités propres, non documentées, connues seulement des développeurs (restants), a rendu l'aventure hasardeuse tout au mieux. De plus, j'avais prévu qu'en l'absence d'une normalisation adéquate, les données étaient sujettes à diverses anomalies.

> « *Data quality problems are quite common with legacy database designs that have been allowed to degrade over time.* » (Ambler, Sadalage, 2006) p.63 et (Ambler, 2003) p.162

Les prochaines sous-sections discuteront des techniques de test unitaire et d'intégration que j'ai mises en place entourant la migration des données afin de pouvoir améliorer la validité des résultats.

3.2.3.1 Tests unitaires

En ce qui a trait aux tests unitaires, j'ai dû revoir l'approche différemment. Ajouter une donnée dans une table pour ensuite la lire et comparer si cette donnée est bonne est davantage un test unitaire propre à une équipe de développement de code source d'un SGBD que de celui de la transition d'un modèle vers un autre (le parallèle applicatif de déclarer une variable, immédiatement lui assigner une valeur et exécuter un test si l'assignation a été effectuée correctement est inutile). La bibliothèque de l'UQAM recense uniquement trois entrées (dont deux pointant vers le même article) ayant les mots « *database* » et « *unit test* » sans compter qu'un livre laisse entrevoir que les tests DBs sont à un stade moins évolué que les tests applicatifs :

« *...there is still significant opportunity for tool vendors to improve their database testing offerings.* » (Ambler, Sadalage, 2006) p.59

À cause des restrictions mentionnées précédemment et qu'il était voulu qu'un seul et unique script englobe la migration des données, l'implémentation de tests unitaires, voire son « interprétation conceptuelle », a dû s'implémenter autrement. Puisque les fonctions, au sens applicatif, n'étaient pas disponibles, j'ai donc orienté les tests sur trois volets afin de demeurer au plus près de données. En premier, plus de 99,6 % des colonnes de fonctionnalités distinctes de la nouvelle base de données ne permettent pas les valeurs de type « NULL » contrairement à la version actuelle ou ce pourcentage avoisine les 50 %. Cette augmentation limite grandement les questionnements du genre « *Manque-t-il une donnée ou est-ce un état valide?* ». La réduction des colonnes « NULL » a limité les permutations entre « présence / absence » au sein d'une même ligne dans une table, où la question suivante se posait: « *Est-ce que la combinaison des données représente un état valide?* ». Dans la nouvelle version de la base de données, si la donnée n'était pas présente, c'était un problème et le SGBD levait une erreur dès la tentative d'insertion d'un « NULL ».

En second, j'ai implémenté des contraintes, autant au niveau de la table, de la colonne ainsi qu'aux objets de référencement (clé primaire, clé étrangère et renforcement de l'unicité). Ceci a encadré le domaine des valeurs possibles servant aussi à titre de tests pour la validité des données. J'ai dû tenir compte que certaines valeurs étaient réservées à la plateforme électronique en tant que « *système* ».

« *...improve the data quality by* ***narrowing*** *the acceptable values within a column.* »
(Ambler, Sadalage, 2006) p.57 et (Ambler, 2003) p.165

En troisième, j'ai renforcé les types de données. Lors de la migration, ceci pouvait servir d'aide afin de s'assurer que la donnée était bien migrée vers un type de donnée identique limitant les erreurs humaines potentielles. Une dérogation du type de données peut générer une erreur d'incompatibilité ou sinon dégrader la performance avec des conversions implicites effectuées par le SGBD.

Ces trois tests, implémentés et renforcés par le SGBD, permettaient une rapide identification d'une relation ou d'une donnée d'affaires invalide. Ce n'était pas des tests unitaires à proprement parler, mais ils correspondent plus ou moins à la définition de « F.I.R.S.T. » (*Fast, Independent, Repeatable, Self-Validating, Timely*) (Martin et al., 2009). « *Fast* » car nul besoin de les démarrer, ils sont constamment opérationnels. Plus ou moins « *Independent* » car ils nécessitent un sous-ensemble de données pour leur exécution (ex : si la table est vide, une contrainte sur une colonne ne peut pas être exécutée). Toutefois, lorsque les données étaient en place, toutes les contraintes pouvaient être testées indépendamment et selon l'ordre voulu. « *Repeatable* » car ils offraient le même résultat indépendamment de tout autre contexte ou environnement et ce, garanti par le SGBD. « *Self-Validating* » car le résultat étant binaire, le test accepte la valeur ou il ne l'accepte pas. Un avantage de ce type de tests pour « *Timely* » c'est que peu importe le moment de sa création (avant ou après le « code de production »), par défaut, les contraintes appliquent la vérification immédiatement sur l'ensemble des valeurs contenues dans la base de données. Toutefois, je les ai créées avant pour bénéficier de leur soutien lors de l'exécution du script de migration, un semblant d'équivalent au « *Test Developement Driven* » ou « TDD ».

Cette section, avec la création de la base de données du sous-chapitre précédent, complète les activités SI 4.3 à 4.8.

3.2.3.2 Script de migration

Pour l'écriture de ce script de migration, j'ai dû migrer toutes les bases de données de production des clients, afin de m'assurer qu'aucun cas d'affaires particulier ne soit omis. Ceci a alourdi considérablement la phase de développement, car il n'y avait pas une seule base de données représentative pour y travailler, mais plus d'une dizaine. De plus, le client le plus volumineux nécessitait plus de onze heures de temps de migration vers la fin. J'ai donc dû court-circuiter, à l'aide d'options que j'ai codées, certaines parties pour ne cibler que les endroits qui étaient à travailler.

Par défaut, l'outil SSMS n'affiche aucun statut de la progression lors de l'exécution d'un script. Étant donné que la migration est parfois longue selon le client, j'ai dû coder un statut de progression. Toutefois, le concept d'une fonction réutilisable, comme elles sont définies dans

l'applicatif, n'existe pas. Pour plusieurs parties du code, l'utilisation de fonctions comme dans le code applicatif pour faciliter la réutilisation n'est pas aussi triviale dans SQL. La commande « *Function* » existe, mais à d'autres fins.

> « *User-defined functions cannot be used to perform actions that modify the database state.* » SQL Docs
> https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql?view=sql-server-2017

Pour les DBA « *créatifs* », il existe un contournement à l'aide de « *sqlcmd* » qui est un accès à la ligne de commande et qui offrirait une fonctionnalité similaire à un « *batch file* ». Autant l'avantage d'utiliser des fonctions est considérable du côté applicatif, autant tenter de la simuler est problématique au niveau des bases de données afin d'encapsuler une logique de manipulation de données. Pour pallier ce problème, j'ai créé des vues afin d'encapsuler certaines logiques d'affaires et éviter de les dupliquer.

En plus de ces limitations, la modélisation initiale utilisait une forme d'héritage *ad hoc* (le SGBD utilisé ne supporte pas l'héritage comme en orienté-objet) à travers les niveaux des *scopes* en plus de mixer des concepts d'affaires au sein d'une même table. Pour certaines entités, l'héritage était encodé directement dans les procédures stockées. Donc, j'ai dû prévoir un découplage pour extraire les entités mixées entre elles, autant au niveau DDL que DML. Toutes ces manipulations ont produit un script de migration qui s'est étalé sur près de 7 700 lignes.

Cette sous-section fait partie de la solution développée, tout comme la structure de la nouvelle base de données et donc s'incorpore dans SI 4.3 à 4.7 d'ISO 29110.

3.2.3.3 Tests d'intégration

Le cas de test d'intégration le plus probant, élaboré depuis le début du projet, était celui de l'exactitude des données entre les deux bases de données. Seules des données identiques pouvaient accroître la validité des mesures qui étaient effectuées par la suite. Sa forme finale ne pouvait pas être connue avant la fin de la migration des données parce que les divers concepts d'héritage ont été décelés pendant la migration. Les avoir implémentés au départ aurait demandé à adapter les scripts d'intégration constamment.

Donc, une fois les données migrées, j'ai élaboré quatre tests « d'intégration ». Une vérification d'un décompte indiquant si les entités importantes avaient été migrées intégralement. Une vérification que les vues pour les développeurs s'exécutaient sans erreur de structure. Une autre pour vérifier si une requête d'effacement était bien propagée à travers les relations « *delete cascade* » et une dernière pour simuler le code et comparer la sortie entre les deux versions au niveau des procédures stockées qui seront mesurées par la suite. Ces quatre tests couvraient tous une perspective différente, la structure, la logique d'affaires entre les relations, la logique d'héritage au niveau de la structure de la version actuelle ainsi que l'héritage des données dans le code des procédures stockées.

Finalement, loin dans le processus, la plateforme utilise une batterie de tests d'intégration qui servira à vérifier que les données sont migrées correctement par l'entremise de fonctionnalités propres à l'applicatif de la plateforme. Cette batterie comporte certaines défaillances, mais l'éventail de tests qu'elle exécute sera un ajout bienvenu.

Cette sous-section fait référence à SI 3.5 à SI 3.8 d'ISO 29110, à sa manière. L'ensemble des trois dernières sous-sections correspond à SI 5.3 en ce qui a trait à la composition de la solution pour la tester.

3.2.4 Mesurer les écarts

Une fois les bases des données migrées disponibles, venait le temps de sélectionner celles qui dénotaient un intérêt particulier. J'ai identifié trois clients qui se démarquaient, soit par la taille de leur base de produits, par l'utilisation des ressources machines nécessaires ou par l'utilisation qu'ils font de la base de données des produits. Le choix que j'ai proposé a fait l'approbation du directeur du commerce numérique. C'est à cette étape que l'auditeur mentionné à la *section 3.1.4* a fait son entrée. Son but premier était d'assurer la validité des mesures et des résultats.

3.2.4.1 Auditeur

Ce nouveau rôle a permis un regard neuf, impartial et surtout initiant le projet au seul membre senior des développeurs de la plateforme. Pour ce faire, j'ai dispensé sa formation dans une salle de réunion avec un téléviseur grand écran afin de pouvoir faciliter la transmission de l'information, visuellement. Cette formation avait pour objectif de présenter le projet, les propriétés de la nouvelle base de données, les exigences du projet, les outils utilisés pour mesurer, les raisons des choix émis ainsi que les façons de faire. Tout ceci était dans le but de valider que ce qui serait mesuré par la suite était conforme, approprié et significatif.

Ceci correspond à l'activité SI 6.2 et servira d'assises pour SI 6.3 et SI 6.4 de la norme ISO 29110.

3.2.4.2 Choix des procédures stockées

Les mesures ont été effectuées sur quatre procédures stockées précises. La raison de leur choix va comme suit :

1) Celle taxant le plus les ressources machines;
2) Celle invoquée le plus souvent;
3) Celle invoquée pour le détail des produits (seconde la #2);
4) Celle qui effectue une gestion transversale au niveau de tous les *scopes*.

Les trois premières, bien qu'elles ne ciblent qu'un *scope* à la fois, taxaient le plus la base de données des produits au niveau des ressources machines et représentaient l'usage typique de la plateforme au niveau transactionnel. Cette analyse, les procédures stockées taxant les ressources, avait déjà été formulée en janvier 2017 comme initiative afin de tenter de réduire la consommation des ressources par celles-ci.

3.2.4.3 Choix des attributs mesurés

À chacune de ces procédures stockées, les attributs mesurés pour satisfaire aux exigences d'affaires ont été : l'utilisation CPU (le premier besoin d'affaires technique), le temps d'exécution moyen en mode « single-thread » et en mode « multi-thread ». Ces derniers

attributs ont servi à répondre au besoin implicite de performance égale ou accrue de la non-détérioration des ressources machines.

Pour la base de données comme telle, deux mesures ont été effectuées : sa taille réelle ainsi que sa taille compressée. Cette dernière a servi pour l'évaluation du changement des coûts associés aux sauvegardes de la base de données dans l'infonuagique.

Avec chaque mesure de base retrouvée ici, des mesures dérivées en ont été extraites. Ces mesures dérivées indiquent le pourcentage de différenciation entre la base de données actuelle et la nouvelle. Le reste des détails se retrouve à l'annexe IX « *Plan de mesure* ».

3.2.4.4 Outil et procédure de test

La première étape de la procédure était de traduire le code des quatre procédures stockées de la version initiale vers le nouveau format et de s'assurer que le jeu de données retourné par les deux versions était identique. Il y avait une exception à cette règle, le retour de la liste des attributs. Au lieu que celle-ci tienne sur un ensemble de colonnes (pivoté par le SQL dynamique, une conséquence du modèle EAV), la liste a été retournée sur une seule colonne sans passer par la commande du pivot tel que mentionné dans le document des plans, dans la section « *Répercussions de l'intégration de la base de données* » :

> « *Parce que le SQL dynamique sera retiré, il est donc nécessaire que le pivot des attributs soit transféré vers le code applicatif parce que l'engin SQL ne permet pas d'effectuer cette opération sur une liste d'attributs de longueur arbitraire sans passer par du SQL dynamique.* »

La seconde étape a été l'uniformité de l'état des bases de données. Chacune des bases de données a eu ses statistiques mises à jour ainsi que tous ses index reconstruits à pleine capacité.

La troisième étape a été d'adapter le code mesuré pour l'empêcher de retourner les résultats à l'écran. L'outil SSMS© n'est pas en mesure de recevoir autant de jeux de données d'un seul coup. De plus, l'affichage des résultats des requêtes n'était pas utile dans les mesures

recherchées et représentait donc du bruit à éliminer. Pour ce faire, les jeux des données retournées étaient stockés dans une table temporaire dans les deux cas et éliminés par la suite. Étant donné que chacune des procédures stockées retournait la même quantité de données, cette surcharge, au niveau des mesures, s'éliminait (sauf en ce qui concerne la liste d'attributs, ce qui venait taxer la nouvelle base de données).

La quatrième étape a été la mesure elle-même. À la base, une loupe invoquait 1 000 fois chaque procédure stockée de chaque base de données. Le nombre 1 000 est arbitraire, mais devait être suffisamment grand pour offrir une démarcation au niveau des données tout en limitant le temps d'exécution total; les procédures stockées de la version actuelle ne sont pas aussi performantes que les nouvelles versions. Cette boucle s'effectua à l'aide du code T-SQL par l'entremise de l'outil SSMS©.

Cette loupe a offert le temps d'exécution applicatif moyen en mode « single-thread ». En même temps que la loupe s'exécute, l'outil « *Profiler* »© a été utilisé pour capturer les autres mesures. Finalement, le temps d'exécution applicatif moyen « Multi-thread » a été mesuré à l'aide de l'outil « SQLQueryStress ».

3.2.4.5 Résultats

Tous les résultats ont été consignés dans le tableur des mesures et sont disponibles à l'endroit indiqué dans le plan. S'ensuit le contrôle du projet.

3.3 Contrôle

La phase de contrôle visait à offrir au directeur du commerce numérique un cadre sur la visibilité de la progression du projet, de la démarche de demande de modification ainsi que des risques liés à la validité des résultats.

3.3.1 Mettre à jour le plan des tâches

Je mettais à jour le document MS-Project© du détail des activités « *Détail des tâches* » (en annexe VIII) chaque fois qu'il y avait un progrès sur une des tâches. Il suffisait d'ouvrir le

document ainsi que la tâche pour y indiquer le pourcentage cumulatif accompli, le reste était recalculé automatiquement par le logiciel. C'était ma tâche de maintenir le progrès des activités à jour car à tout moment celui-ci était disponible aux gestionnaires.

Ceci correspond aux activités PM 2.1 et 3.1 de la norme ISO 29110 pour ce qui est possible. (Rappel : tout ce qui est monétaire ne fait pas partie du projet.)

3.3.2 Analyser et évaluer la demande de changement

À l'origine, personne ne prévoyait déroger de la portée des fonctionnalités. Donc, aucun processus n'avait été élaboré. Toutefois, puisque toutes les fonctionnalités d'affaires étaient sous la gouverne du directeur du commerce numérique, une simple rencontre informelle avec lui était suffisante, comme c'était le cas pour d'autres projets, afin de trancher sur les déviations rencontrées en cours de route.

S'il advenait qu'une demande majeure en vienne à se formuler, une entrée dans le document des plans avait été envisagée à titre de traçabilité, sans plus.

Ceci correspond aux activités PM 2.2 à 2.4 ainsi que PM 3.2 et 3.3 de la norme ISO 29110. Il n'y a pas eu d'écart majeur susceptible de faire échouer le plan autre que ceux déjà définis dans la section des risques du plan initial, pour lesquels des activités de mitigation avaient déjà été mises en place.

3.3.3 Menace à la validité

Afin d'éviter de publier des mesures inadéquates, j'avais mis de l'avant certaines dispositions afin de limiter les déviations face à la validité des résultats. Celles-ci ont été partiellement décrites dans la section 3.2.3, « *Migrer les données* ». Toutefois, malgré la batterie de tests, il restait des éléments à considérer face à la limite de validité des données.

Premièrement, le test comparant les résultats entre la version actuelle et la nouvelle ne pouvait faire l'objet d'une exhaustivité sur tous les produits, de chaque variante, de toutes les bases de données de tous les clients. J'ai pu tester les clients de taille petite et moyenne

à 100 % (chaque produit ainsi que chaque variante séparément pour chaque magasin) mais pour les deux plus volumineux, j'ai dû me rabattre sur un sous-ensemble des produits. Par exemple, le plus volumineux détient 96,5 millions de produits répartis à travers 250 magasins. Théoriquement, il est possible de configurer chacun de ces produits différemment, mais la plupart du temps, ils sont identiques. Il n'y a aucune façon de déterminer les produits identiques de ceux modifiés en interrogeant la base de données actuelle. Tester 133 000 produits a nécessité 6 heures de traitement (donc environ 6 produits testés par seconde). S'il était souhaité d'obtenir une couverture des produits à 100 % pour ce client, cela aurait demandé plus de 4 400 heures de traitement et ceci ne prend pas en compte les 164 millions de variantes de ces produits. J'aurais pu employer diverses stratégies telles que de découper par sous-groupe de magasins et paralléliser les tests. Considérant la connaissance des jeux de données des produits, les résultats de 100 % de couverture avec les clients de taille moyenne et l'étendue des tests effectués sur la portion des clients les plus volumineux, nous avons considéré que c'était suffisant comme validité.

Deuxièmement, seulement les entités de base ont été migrées, ce qui ne correspond pas à la totalité des données entre les deux bases de données. J'ai expliqué au directeur du commerce numérique les raisons de mes choix (couvrir les données que le code utilise pour les procédures stockées à tester) et il me donna son accord comme quoi c'était suffisant pour la validité.

Troisièmement, seulement quatre procédures stockées ont été testées, ce qui en laisse plusieurs ignorées. Par contre, ce sont celles qui étaient les plus imposantes au niveau des ressources et qui avaient fait l'objet d'une étude de performance début 2017. Elles ont donc été approuvées et jugées suffisantes par le directeur du commerce numérique.

Finalement, la base de données actuelle des produits n'a pas de référentiel. Ce référentiel aurait pu servir de base de vérification et de comparaison si les performances obtenues lors des tests de ce projet étaient représentatives. Par exemple, confirmer que les mesures de performance de ce projet usant de la base de données actuelle sont conformes avec son référentiel et donc améliorer la validité des mesures effectuées sur la nouvelle base de données.

3.4 Clôture

3.4.1 Présentation des résultats

Étant donné la nature du projet, une preuve de concept évolutive pour l'évaluation de la réduction du CPU ainsi que de l'ajout de nouvelle fonctionnalité, j'ai limité la clôture du projet à une présentation au directeur du commerce numérique des résultats pour qu'il puisse répondre un « Go / No-Go » pour sa suite.

J'avais élaboré un diaporama pour démontrer les résultats de performance à partir des résultats des mesures ainsi que la façon technique dont les nouvelles fonctionnalités demandées étaient disponibles.

Toute la suite dépendait de cette présentation (compléter les entités restantes, produire la documentation, traduire le code des procédures stockées de la version actuelle vers la nouvelle, la formation des développeurs ainsi que son intégration au sein de la plateforme).

Cette étape correspondait à PM 4.1 de la gestion de projet. Étant donné que la livraison est une présentation de résultats à l'aide d'un diaporama qui réside avec le code source, le dépôt du code du projet demeure donc inchangé et correspond à PM 4.2. Advenant un « No-Go », ce dépôt serait archivé ailleurs (non défini). Les mesures font office de SI 6.5 comme « référentiel » de la nouvelle performance possible ainsi que de SI 6.6.

CHAPITRE 4

PRÉSENTATION ET INTERPRÉTATION DES RÉSULTATS

Je vais d'abord présenter le côté technique de la réalisation de la nouvelle base de données (section 4.1), la suite fera état de la gestion du projet (section 4.2).

4.1 Réalisation de la base de données des produits

La réalisation de la nouvelle base de données et de ses résultats ne s'étaient pas faits sans travail d'encadrement. Cette section présente sa réalisation incluant sommairement le résultat des solutions envisagées et non retenues, puis celle retenue à partir de laquelle ce projet s'est concrétisé.

4.1.1 Solutions envisagées, mais non retenues

Raisons des solutions décrites à l'annexe VI « *Autres solutions envisagées* » et qui n'ont pas été retenue :

- Modélisait des besoins non adaptés pour la plateforme électronique
- Opérait différemment
- Détenait des caractéristiques qui entravaient leur expansion d'affaires
- Dégradait la performance applicative, voire en annexe X « *Dégradation de la performance* »

4.1.2 Solution retenue

Suite aux lectures qui m'ont servi à élaborer le chapitre de l'« *État de l'art en recherche* », les conclusions des divers articles et livres établissaient qu'une modélisation normalisée était le choix le plus approprié et prometteur pour l'atteinte des exigences. Les caractéristiques telles que la réduction globale des jointures, le réglage de performance applicatif minutieux (Panchenko, 2012b), la flexibilité d'expansion de la BD (McMurdo, 1982), une conception améliorée par une normalisation rigoureuse (Thompson, Sward, 2005) ont contribué à son adoption.

La réalisation de la base de données des produits s'était concrétisée en quatre grandes phases : la modélisation, la création de la base de données, la migration des données puis les mesures. Ces quatre étapes seront détaillées ci-après et débuteront par la modélisation de ERDs.

4.1.3 Modélisation

La modélisation d'affaires s'était avérée intéressante pour son attrait sommaire des dépendances entre les entités d'affaires. Elle n'avait été utile que pour confirmer ma compréhension des relations d'affaires. Malgré leurs petites tailles, nous pouvons constater les jeux de couleurs qui représentent la traçabilité des exigences dans les deux diagrammes suivants. Pour le détail des relations entre exigences et couleurs pour la traçabilité, c'est explicité dans le document des plans en page 2 en annexe V « *Plan de projet* ». J'ai volontairement diminué la taille des diagrammes pour éviter que les noms soient lisibles et ainsi en préserver la confidentialité. Voici le résultat du premier diagramme :

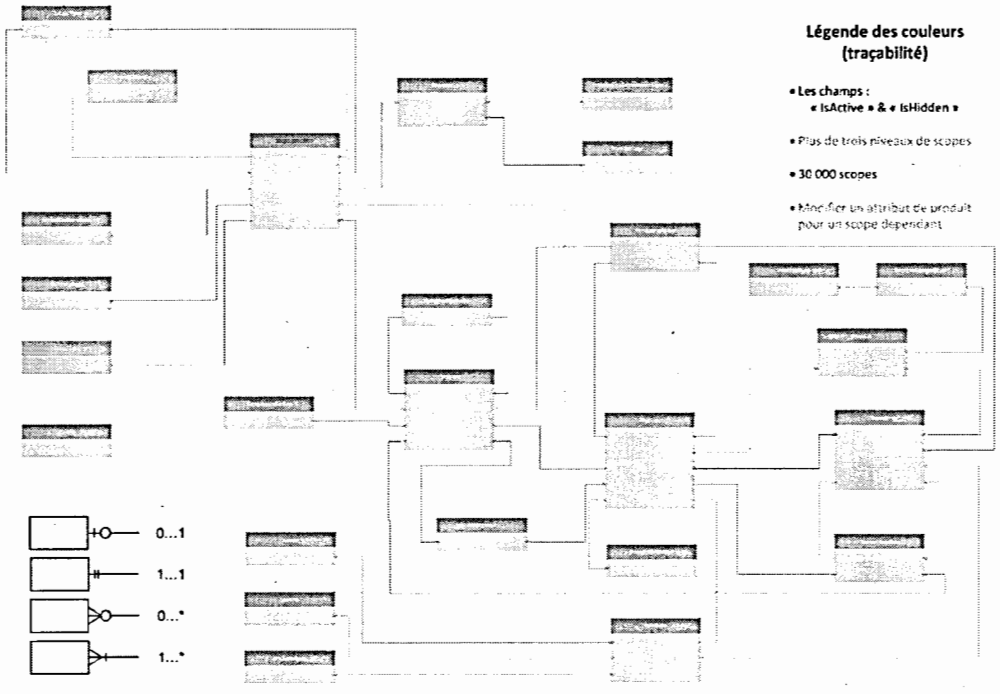


Figure 4-1 - ERD des relations d'affaires

Le second diagramme, celui normalisé, n'avait servi qu'à l'implémentation de la base de données dans le logiciel du SGBD. Une fois complété, il n'avait plus servi. L'outil d'interface du SGBD offrait des fonctionnalités afin de naviguer à travers les relations et était en mesure de générer un diagramme ERD similaire à la demande (sans toutefois respecter la norme ISO 31320), ce qui évitait (et évitera) de devoir maintenir le diagramme à part. Déjà à ce stade, le diagramme était passablement étoffé et les deux problèmes que la littérature mentionne faisaient surface : le problème de visibilité (s'amenuisant par le découpage en plusieurs sous-diagrammes « *View integration* ») et celui de l'isolement d'un regroupement d'entités avec un ou très peu de liens vers d'autres entités, le « *Clustering* » (Kesh, 1995). Des problèmes plus ou moins présents avec l'outil du SGBD, car lors de sa génération sur demande, il est possible de ne sélectionner que le jeu des tables souhaitées.

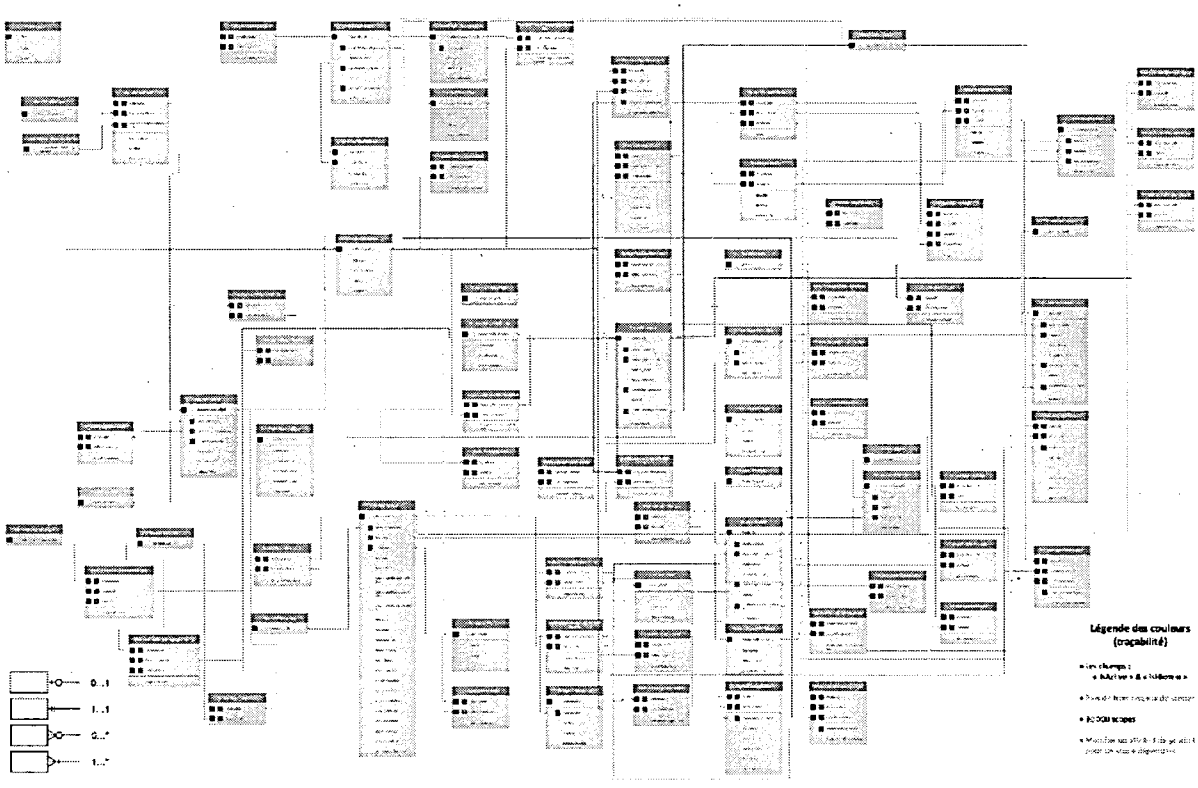


Figure 4-2 - ERD normalisé

4.1.4 Base de données

La nouvelle base de données comporte environ 75 tables. Une réduction significative lorsque, comparée avec celles de plus de 3 000 tables de certains clients de grande taille.

Sa création n'a pas été sans problème. Des fonctionnalités jadis partiellement implémentées, mais dont l'exigence d'affaires n'était plus nécessaire avaient alourdi sa création lors de l'insertion des valeurs par défaut. J'en ai profité pour effectuer un ménage nécessaire afin d'épurer ce qui était rendu obsolète, tel que recommandé par (Martin et al., 2009).

Pour faciliter la compréhension, le travail et s'assurer d'une conformité à travers les objets, j'ai créé un script pour automatiser le changement de nom des index, clés étrangères, clés primaires et des contraintes à partir des noms choisis des schémas et des tables. Ceci pourrait paraitre anodin à première vue pour un développeur, mais pour le personnel qui travaille avec les bases de données, c'est un avantage. Cet avantage se traduit par le fait que lorsque les vues systèmes du SGBD sont interrogées, elles n'affichent pas l'entièreté de l'information (ex. : « *fully qualified name* » ou des propriétés importantes telles que l'unicité ou un filtre sur un index). Avec des noms complets, il est plus aisé de comprendre, maintenir et naviguer dans les données retournées par ces vues systèmes du SGBD. Si ce SGBD les crée automatiquement, les objets auront des noms comportant des séries alphanumériques aléatoires n'apportant aucune valeur significative. Ce script s'ajoutera aux tests d'intégration de la plateforme pour renforcer cette nomenclature dans la prolongation de la base de données, suite à la preuve de concept évolutive.

J'ai aussi rencontré des disparités au niveau de la taille et des plages de données de certains champs ayant pourtant les mêmes noms ou buts recherchés. J'ai donc uniformisé la nomenclature, selon le « *principe de moindre surprise* » (Martin et al., 2009) p.314. Cette dérive provenait de divers ajouts au fil du temps par diverses équipes.

Dans la perspective de son implémentation selon les formes normales, j'ai normalisé le cœur des relations selon DKNF. Il n'y a pas de mécanisme ou de procédure quelconque sur lequel repose la conversion de relations vers le modèle DKNF comparativement aux autres formes (Fagin, 2002) p.403. Puisque la façon de faire n'est pas formalisée, certains laissent sous-

entendre que DKNF est davantage une philosophie. Néanmoins, j'ai orienté les relations de la nouvelle base de données pour qu'elle épouse l'esprit de DKNF à savoir « *Si la base de données accepte (ou retire) une valeur, c'est que celle-ci représente un état valide pour la plateforme électronique* » et qui est dans le même alignement que DKNF « *It is shown that a schema is in DKNF if and only if it has no insertion or deletion anomalies* » (Fagin, 2002) p.387.

Il est resté quelques anomalies impossibles à régler par la base de données seulement ou qui découlaient principalement de la conséquence du modèle EAV « entité attribut valeur ». En dehors de ces quelques cas particuliers, j'ai renforcé les relations afin de ne plus pouvoir générer d'anomalies. À titre d'exemple, un produit est encadré par une définition comportant une liste d'attributs bien précis. Or, il était possible d'utiliser n'importe quel attribut sur n'importe quel produit même ceux en dehors de la définition encadrant le produit. Grâce à DKNF, ce n'est plus possible maintenant d'obtenir ce type de configuration non valide pour la plateforme électronique.

J'ai aussi implémenté, lors de l'effacement d'une entrée dans la nouvelle base de données, un effacement automatique des dépendances qui y sont liées. Par exemple, lorsqu'une catégorie est effacée, il n'est plus nécessaire de coder manuellement l'effacement des dépendances liées à cette catégorie; la base de données s'en charge automatiquement grâce aux « *On Delete Cascade* » et « *Instead of delete* » pour les cas d'exception non gérés par le SGBD. Avec cette approche, n'importe quel intervenant qui doit exécuter une manipulation d'effacement d'une entité, d'une catégorie, d'un produit ou d'un variant, peut maintenant le faire selon l'axe de la plateforme et non selon l'axe du modèle de données. Il est beaucoup plus aisé cognitivement de se limiter au contexte de ce qui est à faire, ici effacer une entité, que d'englober aussi le contexte du modèle de données afin de cibler toutes ses dépendances.

À titre d'exemple, le code actuel nécessite 798 lignes issues du générateur de code SQL pour effacer une catégorie. Pour un client donné, ceci équivaut à légèrement moins de 16 000 lignes de code SQL généré pour effacer une catégorie (ce code issu du générateur de SQL est disponible à l'annexe III « *Exemple de code provenant de la base de données des produits* » en sous-section « *scope.sp_DeleteCategory* » et le code généré est disponible à

la même annexe III, sous-section « *scope.sp_DeleteCategoryExpanded* »). L'équivalent dans la nouvelle base de produits est une seule ligne de code :

« `DELETE FROM Product.Category WHERE Category.CategoryID = @Identifiant` »

Un cas réel survenu a nécessité environ une dizaine d'heures de travail à un développeur pour effacer un seul produit d'une petite base de données d'un client, malheureusement sans réussir du premier coup. Tout ceci sans compter le temps de la gestionnaire, de l'analyste, du soutien technique et aussi du client qui a dû vérifier si le travail avait été exécuté correctement (avec un ou plusieurs retours avec la gestionnaire, « *back and forth* »). Il s'agit d'une productivité (et aussi d'une crédibilité vis-à-vis le client) mise à rude épreuve si l'on considère le coût en termes de salaire et de temps gaspillé comparativement à la nouvelle façon de faire qui s'en occupera automatiquement. Avec la nouvelle base de données, il ne suffira, à l'avenir, que d'écrire une requête similaire à la suivante :

« `DELETE FROM Product.Product WHERE Product.ProductID = @Identifiant` »

Puis la nouvelle base de données se chargera d'éliminer par elle-même les données dépendantes. Donc, avec moins de code, on devrait aussi avoir moins de bogues (Karwin, 2010) p.71.

Cette nouvelle façon d'écrire le code gère aussi implicitement la transaction entourant l'effacement affectant plusieurs tables à la fois. Ce problème est survenu occasionnellement lorsque des manipulations étaient effectuées dans les bases de données lors d'incidents. Compléter la transaction était parfois oublié et donc bloquait toutes les autres requêtes jusqu'à ce que la plateforme écope et que le client réémette un nouveau billet de soutien technique, en urgence parce que plus rien ne fonctionnait.

4.1.5 Bases de données migrées

La migration ne s'est pas effectuée sans heurt. Migrer certains clients a fait surgir diverses déviances survenues au fil du temps – des champs vides qui auraient dû détenir une valeur, des duplications de lignes lorsque ce n'est pas permis au niveau affaires, des données

placées au mauvais endroit – m'ont forcé à établir du code additionnel (environ 673 lignes, ~ 9,2 % des lignes totales du code de migration) pour détecter de quel client il s'agissait afin de gérer et d'activer les bonnes corrections aux endroits opportuns. J'ai rapporté ces déviances auprès du directeur du commerce numérique dont l'une d'elles sera implémentée dans la suite probable du projet, ce qui augmentera significativement le pourcentage de lignes de rectification des données.

4.1.5.1 Survol des critères de qualité du script de migration

Étant régulièrement interrompu, accusant en moyenne une quinzaine de demandes *ad hoc* en plus de projets majeurs, j'ai conclu qu'il était d'une grande importance de maximiser le peu de temps disponible de développement en limitant ma charge cognitive lors de changements de contexte.

Pour tenter de faciliter la lisibilité du script, avec les contraintes décrites dans le sous-chapitre de « *Migrer les données* », trois observations ressortent. La première, pour la démarcation des grandes sections qui demeurent peu nombreuses, j'ai des blocs de commentaires en bannières afin de les retrouver facilement lors de recherches de haut niveau en « *scroll-down* » (Martin et al., 2009).

Chaque commentaire avait sa place, demeurait le plus près de ce qu'il voulait décrire comme intention, était toujours valide et je m'étais imposé qu'aucun code source ne devait demeurer en commentaire. Bref, lorsque je revenais dans le code du script de migration, je pouvais m'y fier à tous les coups. Par la suite, l'utilisation de lignes vides m'a servi de sous-délimiteur aux sections regroupant un ensemble de manipulations communes. Lors de « *scroll-down* » rapide, ces démarcations m'étaient suffisantes pour dénoter un changement de section.

La seconde observation, le point le plus important a été l'utilisation de noms représentatifs aux variables ainsi que de leurs intentions. Normaliser des données implique de nettoyer ce qui est dénormalisé. De ce fait, lorsqu'un nouveau produit était découvert, il était ajouté à la liste de ceux identifiés et conservé comme étant unique. Toute copie était remplacée par un identifiant « pointant » vers le produit original. Donc, la révélation des intentions, par exemple à l'aide de « *Cumulative* » ou « *CurrentScope* » dans la déclaration des tables temporaires

de travail, m'a permis d'aisément comprendre de quel type de produits (déjà traités dans un *scope* précédent ou non) il était question dans la migration à un point donné (Martin et al., 2009).

La troisième observation, j'ai appliqué le « *DRY principle* », « *(D)on't (R)epeat (Y)ourself* » (Martin et al., 2009). Ce principe stipule que ce qui est fait ne doit l'être qu'une seule et unique fois. Avec ce principe, lorsque possible, toute modification me permettait de me concentrer sur un unique bout de code bien précis sans avoir à me soucier des oublis de redondances et a donc contribué à limiter le nombre d'exécutions du script de migration, de débogage et donc de perte de temps.

Pour le débogage, j'ai implémenté un ensemble de lignes de code servant à générer une trace d'exécution. J'ai fait cela à des fins de progression lors de l'exécution du script pour contrer les limitations de débogage dans SSMS© ainsi que pour facilement identifier une ligne de code fautive. Cette dernière était nécessaire, car en utilisant du SQL dynamique, lorsqu'une erreur survenait, l'outil n'était pas en mesure de déterminer la commande exacte où l'erreur s'était produite. Un exemple de ce journal est disponible à l'annexe XI « *Journal de migration* ».

4.1.5.2 Tests unitaires

Quant aux trois tests unitaires, ils m'ont servi à déceler des pratiques plus ou moins uniformes dans la personnalisation de certaines bannières. Par exemple, le retrait de l'abondance de colonnes « NULL ». Ce retrait a fait resurgir des configurations manquantes qui n'avaient pas encore eu de répercussions négatives et qui ont pu être corrigées. Le resserrement du domaine des valeurs avait fait ressortir quelques problèmes dont l'un était la gestion d'ordonnancement de valeurs. Certaines valeurs étaient exclusivement pour la plateforme, mais certaines personnalisations en avaient dérogé pour leur propre bénéfice. Il y a eu aussi un abus d'attributs faussement libellés « système » et des duplications d'entrées qu'un mécanisme *ad hoc* à la plateforme maintient pour que cette dernière ne déraille pas.

Le renforcement de l'unicité des données dans les tables du script de migration ainsi que de la nouvelle base de données m'ont permis de déceler rapidement les quelques erreurs

provoquant des duplications de données. Pour le troisième test unitaire, les types de données, il s'est avéré que des fonctionnalités identiques utilisaient des types différents.

À travers les migrations, j'ai aussi décelé que les tests d'intégration internes à la plateforme et non ceux de ce projet utilisaient des données erronées et non conformes. La plupart de ces données erronées étaient pour « *faciliter* » le code. Pour les autres cas, c'est que la base de données actuelle ne permet pas d'effacer cette information créée. À titre d'exemple, l'ajout d'un magasin crée de nouvelles tables et celles-ci ne peuvent être retirées ensuite. (Oui, il n'est pas possible de retirer aisément un magasin une fois ajouté). Donc dans un test d'intégration du code applicatif qui souhaite tester l'ajout d'un nouveau magasin, ce test se voit forcé de ne pas nettoyer ses données intermédiaires par après. Ainsi les données de démonstration incorporent un jeu de tests latents qui demeure visible dans la base de données des produits. Donc, si une erreur survient pendant l'utilisation de la base de données des produits, comme ça a été le cas pendant la migration, il était nécessaire de conserver en tête que les données n'étaient peut-être pas valides et donc forçaient une investigation additionnelle, soit une perte de temps.

4.1.5.3 Tests d'intégration

Des quatre tests d'intégration, celui visant le décompte des entités principales s'était grandement démarqué dans son utilité. Il était le premier test à être exécuté après une migration des données. Si le décompte était bon, alors les tests subséquents étaient envisageables. Autrement, ça ne valait pas la peine d'aller plus loin avec des tests plus détaillés. Il m'a aussi permis de démasquer des entrées système qui n'arboraient pas les mêmes caractéristiques que les entités avec lesquelles elles étaient stockées. Ce laxisme était permis à cause de la structure « polymorphe » de la version actuelle de la base de données qui tentait de se conformer à tous les critères de ses entités principales stockées sous un seul format de table. Cette « latitude » avait été atteinte grâce aux multiples colonnes pouvant stocker des valeurs de type « NULL ». Les problèmes encourus avec ce test (décompte des entités) étaient de deux ordres. D'abord, il nécessitait de quelques secondes jusqu'à une heure pour s'exécuter tout dépendant de la taille du client ciblé. Ensuite, la multiplication des relations à tester rendait une couverture maximale inatteignable, seules les relations majeures avaient été implémentées. J'ai dû rajouter des options au fil du temps

servant à éliminer diverses sections pour alléger les tests et les découpler pour ne tester que des sections précises, rapidement, lorsque voulu. Toutefois, à cause de l'héritage, ce découplage des tests avait nécessité de la duplication de code. C'est pour cette dernière raison que ce test a été orienté pour comporter une structure identique (dupliquée pour chaque test) facilitant la maintenance si une modification devait survenir.

Le second test d'intégration invoque le code des trois premières procédures stockées identifiées sous « *Choix des procédures stockées* » de la version actuelle ainsi que le code migré de la nouvelle version de la base de données, puis compare les résultats. Si les résultats sont identiques, le test passe au produit suivant et ainsi de suite. Ce test faisait appel à la totalité des informations importantes de la base de données des produits. De plus, ce test était excellent quant à la validité de la consolidation de l'héritage pendant la migration des données. Il m'a permis de détecter des anomalies de données de la version actuelle de la base de données.

Par contre, le code des procédures stockées de la version actuelle n'acceptait qu'un produit à la fois en plus de retourner de multiples jeux de données. Dans une optique de vouloir étendre la couverture des tests, il a donc fallu itérer sur chacun des magasins puis sur chacun des produits. Cette façon de faire a permis d'obtenir la plus fine granularité au niveau de la vérification des données, mais son temps d'exécution total, même pour un magasin de taille moyenne se décompte en plusieurs heures et le plus volumineux a été estimé à un peu moins de trois mois d'exécution (39,2 millisecondes par test de produit, il existe 193 millions de produits). Les tests se sont avérés utiles pour améliorer la validité des mesures de performances futures autant que pour garantir l'exactitude du résultat des jeux de données retournées et donc migrées. Toutes les petites bases de données ont été testées à 100 % avec aucune erreur. Pour les deux bases de données les plus volumineuses, des sous-ensembles significatifs de magasins ont été testés, sans erreur.

Quant aux deux autres tests, ils m'ont été utiles, mais dans une moindre mesure. L'un était exécuté lors d'une modification de structure (DDL) afin de déceler quelle vue non matérialisée en était affectée. L'autre n'a servi qu'à tester les rares cas de « *delete cascade* » que le SGBD ne pouvait implémenter et qui avaient été codés manuellement à l'aide de déclencheurs alternatifs « *instead of trigger* » (par exemple lorsque deux clés étrangères pointent sur une

même table, le SGBD empêche le « *delete cascade* » pour éviter les redondances cycliques, même si logiquement il n'y en avait pas).

La couverture des tests n'est pas de 100 % pour ce qui est des données. Ce n'était pas réaliste de vouloir tout tester lorsqu'une des bases de données détient plus de 320 millions d'entrées sur plus de 25 000 colonnes. Ceci reviendrait à vouloir effectuer la vérification de plus de 3 milliards de valeurs sans comptabiliser les permutations des dépendances fonctionnelles entre les colonnes non-clés!

Pendant l'écriture du code de migration des procédures stockées, à chaque fois qu'une valeur inadéquate était identifiée, j'ai incorporé le code servant à déceler l'erreur au test d'intégration ou au code de pré-migration. Ce faisant, les fonctionnalités principales et celles les plus utilisées étaient constamment testées en plus de fournir une assise solide afin de garantir l'exactitude du jeu de données pour les mesures.

4.1.6 Résultats des mesures

Les résultats se sont avérés positifs, autant pour la réduction du CPU que pour l'amélioration de la vitesse d'exécution. Avant de divulguer les chiffres, la présentation des résultats se scinde en deux domaines. Le premier, transactionnel, comporte les requêtes invoquées massivement en parallèle. Le second, non transactionnel, le « *Delete Category* » est utilisé de façon sporadique et non parallèle. Ce choix a été retenu puisque les valeurs finales des agrégations des tests auraient pu induire en erreur leur représentativité par rapport à l'utilisation de la plateforme électronique.

Chaque procédure stockée échantillonnée acceptait en paramètre un « *scope* ». Puisque les « *scopes* » se déclinent sous un format hiérarchique similaire à un arbre en informatique (voire « *Commerce Modeler* » de l'annexe I pour une représentation graphique), le « *scope* » racine unique a été échantillonné ainsi que le « *scope* » feuille le plus représentatif (se quantifie par la taille de la région ou du magasin et donc du volume « d'utilisation » par leur clientèle respective). De cette façon, les extrémités de l'arbre ont été échantillonnées en plus d'appliquer les différents types d'héritage.

Le temps d'exécution « *single-thread* » indique le temps moyen obtenu des procédures stockées des tests si une personne exécute les commandes sur son poste et que ces commandes détiennent toutes les ressources pour elles seules à chaque exécution. Le temps d'exécution « *multi-thread* » simule de multiples appels en parallèle, forçant le partage des ressources, ce qui est vraiment plus représentatif de la production. J'ai choisi d'afficher les deux temps d'exécution pour trois raisons. La première, éviter la confusion si quelqu'un tentait de reproduire les tests sur sa machine en se fiant sur les temps « *multi-thread* » mais sans utiliser l'outil simulant les appels simultanés. En second, parce que le temps d'exécution « *multi-thread* » est davantage représentatif de l'environnement de la production. En troisième, les temps d'exécution sont significativement différents entre les deux types d'approche.

À titre de rappel, la « v3 » correspond à la base de données actuelle et la « v4 », la nouvelle base de données. Un résultat de « 100 % » indique une réduction de v3 vers v4 de moitié (v3 * 0,5 = v4), « 200 % » équivaut à un tiers (v3 * 0,33 = v4), « 300 % » à un quart (v3 * 0,25 = v4), « 400 % » à un cinquième (v3 * 0,2 = v4) et ainsi de suite. Les valeurs agrégées affichées proviennent des données brutes. Par exemple, pour obtenir le pourcentage 408 %, c'est le ratio entre la sommation de toutes les unités CPU utilisées par v3 et la sommation de toutes les unités CPU utilisées par v4 pour un même client au sein d'une même catégorie de procédures stockées (transactionnelles ou non transactionnelles).

Par exemple, une procédure stockée A obtient une utilisation de 10 unités de CPU pour la v3 et de 1 unité CPU pour la v4. Pour une autre procédure stockée B, pour un même client, elle obtient 200 000 unités CPU consommées pour la v3 et 50 000 unités CPU pour la v4. Leur agrégation de pourcentage a été calculée en utilisant la sommation des unités consommées par version (toujours au sein d'un même client) ce qui se traduit ainsi : ((10 + 200 000) – (1 + 50 000)) / (1 + 50 000) * 100 = 300,01 % d'amélioration pour ce client. Tous les pourcentages présentés sont en faveur de la nouvelle base de données des produits.

Tableau 4-1 - Résultats transactionnels agrégés

	Réduction de la charge CPU	**Réduction du temps d'exécution « *single-thread* »**	**Réduction du temps d'exécution « *multi-thread* »**
Minimum	255 %	314 %	380 %
Maximum	408 %	645 %	1974 %

Les résultats transactionnels proviennent des procédures stockées « *GetProductDetail* », « *GetVariantDetail* » et « *ItemPriceWithInheritance* ». L'écart au niveau de la ressource CPU provient d'un facteur principal : un client peut configurer les propriétés et les regroupements possibles de ses produits différemment des autres clients. Ces configurations ont une incidence non négligeable, car certains clients utilisent des fonctionnalités que d'autres laissent de côté. Cette incidence est aussi accentuée par la manipulation de l'héritage qui diffère entre leurs magasins, ce qui change la quantité de données concaténées par le SQL dynamique et donc se répercute sur le CPU. Il faut aussi rajouter à ces différences que pour la version actuelle des procédures stockées, celles-ci doivent régulièrement utiliser des tables temporaires. Ces tables temporaires sont nécessaires afin de contourner des problèmes de performance, consolider l'héritage au niveau du code SQL ou déjouer les différents états des sessions entre le code SQL standard et celui qui est dynamique. Cette surcharge n'est pas présente dans la nouvelle base de données, car tout type d'héritage est déjà persisté, les différentes sessions liées au SQL dynamique n'existent plus, et les formes normales évitent d'avoir à contorsionner les données pour éviter des problèmes de performance.

Pour l'écart dans les temps d'exécution, c'est le même constat que pour la ressource CPU, mais avec une nouvelle variable : la taille des données. Ici, on peut constater ses effets sur le temps d'exécution « *multi-thread* » lorsque plusieurs requêtes roulent en parallèle. La taille des données a un impact sur la quantité de mémoire que le SGBD doit manipuler à l'interne tout comme pour la quantité de données à stocker dans les tables temporaires. Ce que le temps d'exécution « *multi-thread* » sous-entend, c'est que les ressources autres que le CPU sont largement plus sollicitées dans la version actuelle que pour la nouvelle base de données. Les données suivantes font référence à la procédure stockée « *DeleteCategory* ».

Tableau 4-2 - Résultats non transactionnels agrégés

	Réduction de la charge CPU	**Réduction du temps d'exécution « single-thread »**
Minimum	368 %	393 %
Maximum	20 376 %	98 708 %

L'explication de ces écarts si importants provient du fait que plus un client possède de « *scopes* » ou de magasins, plus le poids du SQL dynamique sera lourd sur les ressources. Plus une chaîne « *string* » contient de caractères, plus ce sera long à la concaténer avec une autre. Les autres points sont similaires aux écarts au niveau du CPU. La configuration, la taille des données, l'héritage au niveau du code tout comme celui au niveau de la structure ont un impact sur le temps d'exécution.

Les résultats suivants font partie des mesures afin d'évaluer la fluctuation des coûts sur les sauvegardes des bases de données dans l'infonuagique. Plus la taille est volumineuse, plus le coût est élevé. La réduction de la taille n'est pas un objectif en soi, mais elle a une incidence indirecte sur les coûts de stockage dans l'infonuagique.

Tableau 4-3 - Résultats de la réduction de la taille compressée

	Réduction de la taille compressée de la base de données
Minimum	102 %
Maximum	407 %

L'explication des écarts provient principalement des caractéristiques telles que l'utilisation de types appropriés de données, l'élimination de la duplication des données, la consolidation de tout type d'héritage ainsi que l'utilisation de clés avec comme identifiant des entiers au lieu des chaînes de caractères.

Toutes ces améliorations sont intéressantes, mais il est nécessaire de conserver à l'esprit que les procédures stockées de la base de données actuelle (v3) ont été optimisées au cours des six dernières années. Celles de la nouvelle base de données (v4) ont été écrites tout simplement sans chercher intensivement à les optimiser. Quelques index additionnels avaient été ajoutés, mais sans plus, puisque ceux implicitement définis par les clés primaires étaient dans la très grande majorité des cas suffisants. De plus, afin de conserver la similitude des données retournées entre le code actuel et le nouveau, certaines d'entre elles ne sont plus nécessaires à cause du fait que tout type d'héritage était déjà persisté adéquatement. Ce faisant, certaines contorsions pourront être éliminées, ce qui creusera davantage l'écart de performance entre les deux versions. Donc, les chiffres présentés ici pour la version actuelle de la base de données sont un *maximum* atteignable tandis que pour la nouvelle, ils sont un *minimum*, sujets à de meilleurs rendements.

Les résultats plus détaillés par client et par procédure stockée se retrouvent à l'annexe XII sous « *Données brutes des mesures* ». Puis, les extrêmes listés ci-avant se retrouvent à l'annexe XIII sous « *Données agrégées* ».

4.2 Compte rendu de gestion de projet

Le projet s'est échelonné sur plus d'un an et demi et a requis plus de 640 heures de travail, excluant l'écriture de ce rapport. Les premiers balbutiements ont eu lieu en mai 2018, mais son envol a réellement été effectif à partir de septembre 2018.

Tel qu'expliqué dans la section 3.1.3 « *Planifier le projet* », toutes les estimations ont été basées sur l'expérience professionnelle et personnelle seulement. Or, la sommation des heures estimées dans le détail des tâches totalise 645h alors que 640h ont été utilisées! Humblement, j'estime qu'une partie est attribuable à l'expérience, mais une autre relève nécessairement d'une part de coïncidence.

4.2.1 Liste détaillée des tâches

J'ai produit une liste détaillée des tâches afin de découper le projet selon une unité simplifiée, c'est-à-dire le travail accompli. J'ai choisi cette unité de travail pour être en mesure d'effectuer

le suivi de progression par tâche et non par une unité de temps quelconque. J'ai préféré cette approche puisque mon temps alloué au projet était non prévisible, voire arbitraire et que, selon les principes Agile :

« *Working software is the primary measure of progress.* » (Ambler, 2003) P.17, le manifeste Agile aussi repris par IEEE 730 (IEEE Computer Society, 2012) p.95

En établissant un certain parallèle conceptuel entre « base de données » et « logiciel », j'ai adapté cette formulation au monde des bases de données « *Established structure or working scripts are the primary measure of progress.* » Le détail et la progression du projet ont été consignés dans le document MS-Project© du « *Détail des tâches* » (en annexe VIII). Encadré en vert, un crochet désignant les tâches complétées à 100% et encadré en bleu le pourcentage complété et sa durée. D'un seul coup d'œil, un gestionnaire pouvait déterminer, au niveau d'imbrication souhaité, l'avancement du projet.

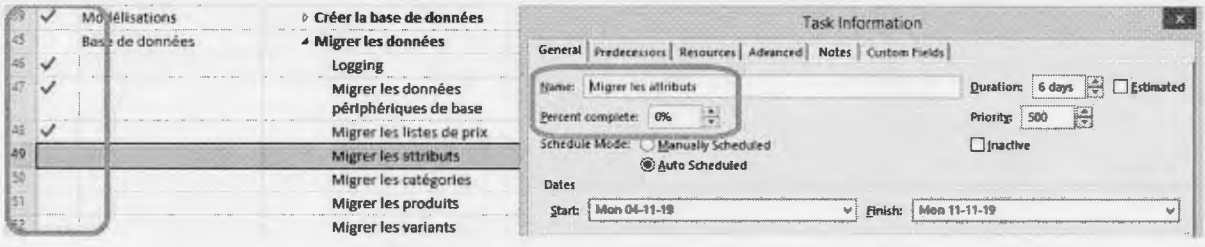


Figure 4-3 - Visualisation de la progression

J'ai retiré volontairement tout ce qui était rattaché aux coûts ou aux durées estimées en termes de calendrier. L'outil les calculait automatiquement, mais étant donné l'imprédictibilité de mon horaire, toute date n'avait donc aucune pertinence.

4.2.2 Plan révisé et approuvé

Le directeur du commerce numérique n'avait rien à rajouter suite à la présentation du projet. Les exigences, les objectifs, la portée, les mesures et le suivi disponibles étaient adéquats et conformes à ses attentes. Son approbation initiale écrite se retrouve à l'annexe XIV « *Confirmation de l'approbation du projet par le directeur* ».

4.2.3 Contrôle

4.2.3.1 Mise à jour du suivi de progression

Dès sa mise en ligne, le plan a été à jour et disponible en tout temps au directeur du commerce numérique. Le moment où il souhaitait consulter la progression du projet à partir du fichier MS-Project© du « *Détail des tâches* » (en annexe VIII) était à sa discrétion.

4.2.3.2 Demande de changement approuvée ou rejetée et plan modifié

Malgré la faible portée, une demande de changement est survenue. Une fonctionnalité découverte s'était retrouvée bornée à l'intérieur d'une autre fonctionnalité limitant grandement la flexibilité de cette première. De plus, il s'agissait de concepts différents. J'ai alors mis au courant le directeur du commerce numérique et il a demandé d'extraire la fonctionnalité et d'en faire une entité à part. Cette décision a été inscrite dans le document des plans en page 6 de l'annexe V « *Plan de projet* ».

Deux autres modifications ont été identifiées au fil du temps et ont été ajoutées aux fins de traçabilité informative uniquement. L'une d'elles était une décision de la nomenclature d'une entité d'affaires qui affichait une terminologie différente de celle employée dans la base de données. Quant à l'autre, il s'agissait d'une convention non respectée facilitant le repérage d'attributs système versus ceux définis par le client. Pour cette dernière, son implémentation ou non n'affecte en rien les résultats et la validité des mesures. De ce fait, ce changement sera apporté dans la seconde phase du projet si le directeur du commerce numérique décide de le poursuivre suite à des mesures de performance convenable.

4.2.3.3 Résultat de l'audit

L'audit a duré un peu plus de quatre heures. Au final, l'auditeur avait émis quatre commentaires. Le détail des commentaires est à l'annexe XVI « *Commentaires de l'auditeur* ».

Il a aussi été question de passer en revue la traçabilité des exigences d'affaires. À ce stade, aucun manque, toutes les fonctionnalités demandées étaient bel et bien présentes et implémentées. C'était une activité pour laquelle l'auditeur avait trouvé un intérêt positif.

Pour finir, j'avais obtenu l'aval de l'auditeur afin de poursuivre avec les mesures, et ce, du premier coup. Ce qui était mesuré était valide et représentatif des points faibles à améliorer de la base de données actuelle. Les cas de tests et de mesures ont aussi été passés au peigne fin et vérifiés pour s'assurer que la logique utilisée était bonne. Le courriel de la confirmation de son approbation est à l'annexe XVII « *Confirmation de l'approbation de l'auditeur* ».

4.2.4 Présentation des résultats

La présentation, disponible en annexe XVIII « *Présentation PowerPoint des résultats* », s'était déclinée en trois volets. D'abord, fut faite la démonstration que les exigences au niveau des fonctionnalités avaient bien été implémentées. Ensuite, l'impact sur les ressources et la performance applicative a été grandement apprécié par le directeur. Surtout l'effacement d'une catégorie dont les problèmes sont toujours d'actualités pour le client le plus imposant. S'en est suivie la discussion sur la comparaison du code entre la version actuelle (SQL dynamique) et la nouvelle (sans SQL dynamique). Cette discussion a été enrichie des difficultés rencontrées par son équipe de développement, les nouveautés en termes de fonctionnalités additionnelles qui s'y sont greffées au fil de la dernière année et demie, puis finalement, sa vision de la base de données des produits dans le futur de la plateforme électronique.

Quant à l'anecdote mentionnée dans le sous-chapitre 3.1.4 « *Présenter le plan* »,

« Le pourcentage d'amélioration que tu vas nous donner, c'est le pourcentage que je vais te donner pour ta note de ton projet. ».

Nous en avons reparlé et si ce n'était que de lui, il me donnerait effectivement 100 % !

CHAPITRE 5

ANALYSE DE LA CONFORMITÉ AVEC ISO 29110

ISO 29110 a servi de toile de fond afin d'identifier les activités minimales que le projet devait contenir. Malgré une bonne adéquation entre la norme et le projet, certaines déviations sont survenues et sont expliquées ci-après.

Débutons par les généralités. Toutes les activités obligatoires de la gestion de projet ont été respectées à l'exception partielle d'une seule facette. Cette facette était ce qui entourait les estimations et le suivi de temps du projet (PM1 à PM3). Cette déviation a été causée par mon type de travail qui m'amène à être constamment dérangé et rend imprévisible toute échéance de date. Donc, trois des quatre catégories de la gestion de projet en pâtissent. Qui plus est, PM3, évaluation et contrôle du projet, n'a que trois activités! Donc, en obtenir une partielle fait diminuer son pourcentage considérablement comparativement aux autres. Pour ce qui était de l'implémentation, « Évaluation de l'implémentation logicielle » et qui était davantage sous ma gouverne, toutes les activités ont été respectées.

Nom du processus	% Exécutée	% Point vers les 27 tâches	Nom de l'activité	Oui	Non	Partiel	# Total de tâches
Gestion de Projet							
	97%	26%	PM1-Planification du projet	14	0	1	15
	92%	10%	PM2-Exécution du Plan de projet	5	0	1	6
	83%	4%	PM3-Évaluation et contrôle du projet	2	0	1	3
	100%	4%	PM4-Clôture du projet	2	0	0	2
Moyenne du processus	93%			23	0	3	26
Implémentation logicielle							
	100%	4%	SI1-Initiation de l'implantation du logiciel	2	0	0	2
	100%	7%	SI2-Analyse des exigences du logiciel	4	0	0	4
	100%	14%	SI3-Architecture et conception détaillée du logiciel	8	0	0	8
	100%	13%	SI4-Réalisation du logiciel	7	0	0	7
	100%	11%	SI5-Test et intégration du logiciel	6	0	0	6
	100%	5%	SI6-Livraison du produit	3	0	0	3
Moyenne du processus	100%			30	0	0	30
Tâches non exécutées		3%					
Moyenne total du processus	97%						

Figure 5-1 - Tableau sommaire de la conformité avec la norme ISO 29110

Ensuite, étant donné que le personnel technique ne se limitait qu'à moi, toutes les activités techniques de groupe, assignation des tâches et rôles étaient implicitement effectuées *de facto*, ce qui correspond au début de chaque grande catégorie PM 1 à 4 et SI 1 à 6. Même

constat pour tout ce qui était d'établir une compréhension mutuelle liée aux problèmes de communication au sein de l'équipe technique. Pour ce qui était de la vérification et de la validation, l'interface de la plateforme ainsi que le directeur du commerce numérique faisaient office de « co-équipiers ».

La particularité de l'équipe à une seule personne était en fait une conséquence de la solution retenue, la reconceptualisation laissant de côté l'acquisition d'une base de données des produits d'une autre ligne d'affaires. Autre particularité qui découle de l'activité de la solution retenue, c'est qu'elle devait (la solution retenue) être connue avant de pouvoir finaliser le plan de projet, car selon ISO 29110, celui-ci dicte l'exécution des SI. Donc, si la conformité à la lettre avec ISO 29110 avait été exigée, il aurait été nécessaire de scinder le projet en deux sous-projets pour respecter la clause « *la faisabilité du projet a été démontrée avant sa mise en branle* » (Organisation internationale de Normalisation, 2012). En scindant en deux projets, l'extrant du projet de la solution retenue aurait été l'intrant de la preuve de concept. L'intérêt de l'ajouter en une activité additionnelle entre la définition des besoins et l'établissement du plan de projet était approprié puisque la finalité était une mesure et non un choix de solution. Elle n'est qu'une étape transitoire. Malgré cette légère adaptation de la norme pour ce cas précis, les avantages octroyés par celle-ci demeurent toujours valides.

Le « produit attendu » était de deux ordres : le « résultat attendu » du pourcentage de gain en réduction de la charge du CPU; une nouvelle base de données des produits comportant les exigences fonctionnelles. Quant au processus rigoureux, c'était la méthodologie ainsi que le document MS-Project© du « *Détail des tâches* » (en annexe VIII) dans le processus de mise en œuvre.

L'avant-dernière particularité a trait à l'ajout d'un auditeur. Ce rôle ne faisait pas partie de ceux recommandés par ISO 29110, mais rien n'empêchait d'en ajouter non plus. Cet ajout a servi de vérification et de validation quant aux mesures afin d'améliorer la fiabilité des résultats pour les gestionnaires.

La dernière particularité, étant donné qu'il s'agissait d'une « preuve de concept évolutive », la définition de « Livraison du produit » s'était limitée à la présentation finale des résultats.

Somme toute, les différences notées entre la norme et l'application de la méthodologie dans le cadre du présent projet étaient davantage liées au contexte de celui-ci (preuve de concept évolutive) qu'aux différences entre les bases de données et le développement applicatif.

Pour une mise en correspondance détaillée entre les activités de la norme ISO 29110 et les activités de ce projet, se référer à l'annexe XIX « *Résultat de la conformité avec ISO 29110* ».

CHAPITRE 6

RÉFLEXION ET DISCUSSION DU GÉNIE LOGICIEL DANS CE PROJET

6.1 Redécouverte complète des relations d'affaires

À première vue, l'usage possible de la redécouverte des relations d'affaires n'a pas eu un grand impact pour ne pas dire aucun impact, peut-être parce qu'à ce jour, peu de personnes en étaient informées. Son utilité réelle résidait par son renforcement implicite des relations d'affaires dans la base de données pour les développeurs et les partenaires. Cette logique relationnelle renforcée automatiquement par le SGBD était toujours à jour et était bien mieux que de la documentation qui elle peut se dégrader au fil du temps par rapport à la réalité. De plus, cette logique relationnelle dirigera l'écriture des requêtes et, consciemment ou non, forcera les développeurs à tenir pour acquis les liens relationnels entre les données.

Autant l'auditeur que moi avons trouvé que la charge de travail afin de maintenir les ERD à jour était une tâche dénuée de sens. Dans l'éventualité où une représentation graphique serait nécessaire, l'outil SSMS© offre la possibilité de générer le ERD normalisé à l'aide des clés qui renforcent les formes normales dans la base de données. De cette manière, le diagramme généré sera toujours à jour et il n'y aura plus à se souvenir de les maintenir pour tout changement à la base de données à l'avenir.

6.2 Réduction anticipée des coûts

Les résultats obtenus permettaient d'envisager la réduction de la puissance des VMs d'un ou deux crans tout en améliorant la performance applicative. Le coût uniquement pour les deux principaux clients dépasserait facilement le millier de dollars par mois. De plus, l'avantage de requêtes plus rapides et moins énergivores en ressources pourrait permettre de ne pas avoir à provisionner des machines davantage performantes pour les occasions très achalandées telles que pour le *Black Friday* jusqu'au *Boxing Day*. Ici, il y a environ un mois entre les deux dates. Toutefois, afin d'être fin prêt pour ces événements, le provisionnement additionnel s'effectue à l'avance et le retour à la normale se fait au retour des vacances du jour de l'an. Cette plage prolonge le tout de quelques semaines additionnelles, qui nous sont évidemment

facturées dans l'infonuagique. Rapidement, il est facilement envisageable que les économies récurrentes avoisinent 25 000 $ annuellement uniquement pour la production des clients mesurés dans ce projet. De ce chiffre, les améliorations au niveau de la performance applicative ne sont pas prises en compte, ce qui laisse une marge de manœuvre additionnelle. Nous pouvons rajouter aussi le fait que tous les autres clients qui ne font pas partie des principaux clients mentionnés dans ce projet bénéficieront aussi de ces améliorations, ce qui représente un coût moindre. Finalement, si nous ajoutons aussi les autres environnements, *Quality Assurance*, *Integration* et *Staging*, le montant final économisé annuellement sera bien plus élevé.

Les avantages connexes liés aux qualités non fonctionnelles de la nouvelle base de données (non mesurés dans ce projet mais *de facto* entérinés par l'auditeur et le directeur du commerce numérique) seraient aussi des facteurs améliorant son retour sur investissement. Le retrait du SQL dynamique, le nombre réduit de lignes de code (une plus grande facilité à l'écriture des requêtes), la facilité de débogage du code et l'automatisation de l'effacement des relations pourraient permettre aux développeurs de passer moins de temps avec la nouvelle base de données et donc de récupérer ce temps à d'autres fins. La facilité de compréhension du nouveau modèle pour les partenaires pourrait aussi permettre de réduire le nombre des billets que ceux-ci créent parce qu'ils ont de la difficulté à manipuler la base de données. Finalement pour le client, les risques d'incohérences de données sont amoindris. Ceci pourrait aider à réduire les billets au soutien technique.

Il n'y a pas que les coûts directs comme gains liés à cette reconceptualisation de la base de données des produits. Plusieurs autres économies possibles s'y rattachent, mais n'ont pas fait l'objet de mesures dans ce projet, donc demeurent incertaines encore. Toutefois, si elles s'avèrent fondées, des économies en temps, donc convertissables en salaire, sont aussi une source non négligeable de retour sur investissement.

6.3 Incohérences de traitement de la BD actuelle

Pour une raison inconnue, par le passé, des fonctionnalités persistaient leurs données dans des tables servant à d'autres fins. Pour y parvenir, des suffixes à des noms avaient été codés afin de départager ces fonctionnalités abusives des autres. Cet abus laissait transparaitre ces

noms de code à travers l'interface utilisateur, ce qui était peu élégant. Toutefois, à ce moment, le besoin original avait été comblé.

Au fil du temps, les besoins vinrent à évoluer et le cadre de la persistance des données, adéquat pour la fonctionnalité qu'elle devait desservir, devenait trop étroit pour la fonctionnalité abusive. Agrandir le cadre standard pour subvenir aux besoins de la fonctionnalité abusive était démesuré. L'ampleur de la modification de la structure pour départager les deux fonctionnalités était trop lourde pour la demande à ce moment. Cette dernière était donc toujours relayée aux oubliettes. Puisque la modification de structure est une activité rare, c'était le moment ou jamais d'effectuer cette séparation.

Lorsque cette amélioration a été proposée au directeur du commerce numérique, il a immédiatement acquiescé pour enfin les dissocier et pouvoir faire évoluer indépendamment les deux fonctionnalités.

Les tests d'intégration ont aussi décelé des erreurs dans les données. Est-ce que les corrections apportées seront suffisantes pour enrayer le message d'erreur qu'un des clients obtient systématiquement? Ça reste à déterminer, mais une chose est sûre, les données seront valides grâce à la forme normale DKNF.

Le script de migration détient lui aussi des options afin d'appliquer des corrections à certaines données pour des clients précis. J'ai corrigé neuf cas d'exceptions de données incorrectes qui ne seront plus présentes dans la nouvelle base de données. Ces exceptions étaient possibles, car l'interface graphique et les méthodes d'imports, deux processus distincts, ne valident pas les données de la même façon. Donc, des anomalies pouvaient s'y glisser par l'une ou l'autre des méthodes.

Tous ces cas qui ont été corrigés serviront à accroitre la fiabilité et l'intégrité des données. Une fiabilité inter équipes de développement, que ce soit les développeurs de la plateforme électronique, le bureau outre-mer, le soutien technique ou les partenaires, les données ne feront plus place à interprétation. Une interprétation qui nécessite de creuser dans le code source afin de déceler les subtilités. Maintenant, la nouvelle base de données renforce les relations entre les entités selon l'ordre logique appliqué par la plateforme électronique.

6.4 Poursuite du projet dans l'entreprise

Ce n'est pas parce que la preuve de concept est maintenant terminée que ça implique que le travail effectué n'est plus bon ou jetable. C'est tout l'intérêt justement que la preuve de concept soit évolutive. Le travail avait été produit en conservant en tête qu'il serait poursuivi jusqu'à la finalité du projet global, une base de données des produits fonctionnelle pour la production.

La preuve de concept s'était limitée aux entités principales utilisées dans le code exerçant la plus grande pression sur les ressources CPU d'une VM ou qui était invoqué le plus souvent. Pour que la nouvelle base de données puisse définitivement remplacer la base de données actuelle, sa continuité est nécessaire. Voici un survol des tâches principales à compléter :

- Migrer et tester les associations;
- Incorporer les nouvelles fonctionnalités survenues au fil du développement de la preuve de concept évolutive;
- Traduire le code des procédures stockées de la version actuelle à la nouvelle version;
- Utiliser des IDs au lieu de noms entre la plateforme et la BD;
- Adapter les archives et l'entrepôt de données des produits et leur mécanisme de transfert;
- Documenter l'utilisation de la BD aux développeurs;
- Adapter les tests d'intégration de la plateforme;
- Adapter la sécurité de la nouvelle BD;
- Adapter les rapports.

Au fil du projet, qui s'était échelonné sur plus d'un an et demi, diverses fonctionnalités additionnelles se sont greffées à la base de données. Ces nouvelles fonctionnalités n'interagissaient pas avec les entités primaires entourant les produits, donc, n'invalidaient en rien la preuve de concept. À ce jour, elles sont toutefois nécessaires au bon fonctionnement de la plateforme électronique.

Pour la pérennité de cette nouvelle base de données, des recommandations sont formulées en annexe XX sous « *Évolution de la base de données des produits* ».

6.5 Processus de développement

En ce qui a trait à un projet de base de données, même avec une unique personne côté technique, ISO 29110 s'applique bien avec quelques nuances, par exemple au niveau de la suite de tests. Elle a permis de ne pas oublier les grandes lignes directrices des activités à y incorporer. Un de ses grands atouts a été d'effectuer un détail des tâches dans la planification, ce qui force une personne à s'arrêter et à visualiser l'ensemble des composantes ainsi que leur mécanisme avant de se lancer dans le code et le découvrir au fur et à mesure (voir le fichier MS-Project© du « *Détail des tâches* » (en annexe VIII)).

Par contre, la norme n'aide en rien concernant les détails d'implémentation. Ces derniers demeurent importants, car ils contribuent à la maintenabilité de la solution. Malgré une norme suivie à la lettre, si le code n'est pas maintenable, son coût sera très élevé. J'ai donc observé deux vecteurs tout au long du projet. L'un est appliqué par la norme pour son processus de développement et l'autre circonscrit les détails de l'implémentation. Cette déclaration est subjective, mais ce sont les deux grands volets qui ont été bénéfiques pour mener à bien la preuve de concept évolutive et ainsi la poursuivre, les deux avec les coûts les plus faibles. L'un cible l'exactitude des exigences et donc limite les coûts en « *rework* », l'autre cible ses coûts de maintenance (tout type confondu) au fil du temps.

Quant au processus de développement uniquement, il a été établi que celui-ci avait un impact significatif sur la qualité du logiciel produit (Trudel et al., 2006). Est-ce qu'il serait bon pour la ligne d'affaires d'investiguer si une accréditation CMMI ou ISO 29110 lui serait profitable afin d'améliorer la qualité de sa plateforme électronique? Est-ce que le jeu de la certification en vaudrait la chandelle? Dans plusieurs cas répertoriés, il semblerait que oui. Toutes les compagnies ayant implémenté ISO 29110 en auraient tiré un avantage quelconque (Laporte et al., 2017). Pour ce qui est de CMMI, son retour sur investissement se situe aux alentours de douze mois (Hinkle, 2007). Selon d'autres informations, des résultats émergeraient en deçà de six mois avant d'atteindre les objectifs complets du niveau 2 (Boisvert, Trudel, 2011). Toutefois, avant même de se lancer dans un tel projet, une étude du processus actuel serait nécessaire afin de constater les similitudes et écarts avec les normes.

Puisque la compagnie opère avec un client paragouvernemental et un autre œuvrant en périphérie du domaine pharmaceutique, la question est lancée, mais la réponse demeurera l'œuvre d'un autre ouvrage.

6.6 Productivité de développement

Plus de 27,5 % des heures travaillées sur le projet l'ont été en heures supplémentaires, soit un peu plus de 177 heures. Ces heures supplémentaires étaient faites en sus aux 229 heures additionnelles en temps supplémentaire sur d'autres projets, plus une autre tranche de 142 heures pour d'autres demandes entre septembre 2018 et 2019. Ceci totalisait 548 heures travaillées en dehors des heures ouvrables normales. Bien que subjectif, malgré la fatigue et les longues heures, ces heures supplémentaires passées sur le projet synthèse soit par le soir ou les fins de semaine, m'ont semblé largement plus productives que celles durant les heures ouvrables régulières. Les parties les plus complexes du code de migration n'ont pu être implémentées que durant ces périodes de calme et surtout n'ayant aucune source de dérangement ou de distraction. Étant donné le type de poste que j'occupe, les interruptions étaient fréquentes, soit environ quinze fois par jour; je les ai comptées. Quinze peut sembler peu ou normal, soit en moyenne une interruption aux trente minutes, mais incluant les petites tâches ou investigations qui les accompagnaient, c'était suffisant pour combler plus qu'une journée entière de travail rapidement. Réparties tout au long d'une journée, cela laissait peu de place à la concentration. J'ai la forte impression que beaucoup de temps a été perdu à cause de ces interruptions fréquentes.

Un moyen d'éviter ces fâcheux contretemps, c'est-à-dire une piste de solution possible, serait de dégager du temps pour les innovations. Une période consacrée à l'horaire régulier, et approuvée par la direction, qui serait prise en compte et respectée lors des planifications. Du temps non interchangeable, afin d'éviter que des projets en retard, ou qui pourraient miroiter de plus hautes priorités, pigent dans cette plage d'heures consacrées à l'innovation, servant aussi à accroître d'autres façons la compétitivité de la plateforme électronique.

CONCLUSION

En premier lieu, nous allons revenir sur le critère premier du pourcentage possible de la réduction de la charge du CPU. Le résultat du côté des requêtes transactionnelles l'a amélioré entre 255 % et 408 %. 408 % de moins pour le client ayant atteint la puissance maximale des VMs disponibles, discuté dans le contexte du chapitre un, a été vraiment favorablement accueilli par le directeur du commerce numérique. Considérant le sous-entendu lors de la présentation du plan, soit une amélioration de 100 % souhaitée, la réaction positive du directeur du commerce numérique face aux résultats, je pouvais affirmer que l'attente implicite avait donc été comblée. Pour ce qui est du cas original, celui de l'effacement d'une catégorie, une opération moins courante et donc à part du transactionnel, la réduction du CPU oscille entre 368 % et 20 376 %, différence due au nombre de magasins des clients testés.

Il reste des détails techniques, mais pas plus difficiles que ce qui a été produit jusqu'à présent. Ce qui demeure comme obstacle à prévoir et à gérer est celui très humain de la résistance aux changements. Considérant la lourdeur de travailler dans la base de données des produits actuelle jumelée au premier commentaire reçu au fil du projet d'un développeur « *Ça aurait dû être comme ça depuis le début* », laisse envisager que les changements seront davantage perçus positivement.

À travers ce projet, nous avons pu constater différentes facettes qui ont été abordées : la normalisation de la base de données, l'application de la norme ISO 29110, le parallèle entre le monde applicatif et les bases de données, et ce, autant au niveau de son développement que pour le concept de tests, unitaires et d'intégration.

Tout d'abord, la prise en charge du projet avec la norme ISO 29110 établit un processus clair des activités les plus importantes. Une liste d'activités qui offre un guide et qui permet de remettre en question si certaines façons de faire sont suffisantes ou adaptées pour le projet en cours. Des cas limites, tels que ce projet avec une seule personne technique pour tout effectuer, amènent à repenser sous un autre angle certaines activités telles que la vérification et la validation.

Ensuite, une modélisation adéquate améliore considérablement l'objectif de réduction de la charge du CPU tout en ayant un impact positif pour la performance applicative. Les données dans la base de données restent les mêmes, mais elles sont présentées sous un format différent, mieux approprié pour le traitement par le SGBD.

Il a aussi été question d'un parallèle entre les façons de faire dans le monde applicatif comparativement à celui des bases de données. Ces deux mondes qui doivent coopérer pour un même objectif final, mais leurs approches, autant au niveau du code que des limitations des outils, diffèrent. Le plus flagrant a été ressenti du côté des tests. Leurs conceptions ont pris une autre tournure, et ce, autant pour les tests unitaires que pour les tests d'intégration.

Dans une moindre mesure, il avait été aussi question de la capacité à déceler les attentes connexes du projet dans son contexte global afin d'améliorer ses chances de succès. Car, si ce n'est pas ce à quoi le client s'attend, c'est un échec. Étant donné que j'ai pu déceler ces attentes connexes en plus de respecter rigoureusement lès critères explicites des exigences, le projet a donc été un succès.

Le souhait du directeur du commerce numérique serait de pouvoir utiliser ma solution en production en 2020. Toutefois, un problème de taille demeure et il l'avait mentionné lui-même lors de la présentation des résultats : c'était ma disponibilité. La division de mes interventions à travers les divers départements ainsi que le peu de soutien technique disponible entourant les compétences nécessaires pour achever le projet demeurent une source de questionnement.

Néanmoins, tout ceci n'indique pas la fin, mais la continuité d'un projet offrant une base solide quant aux capacités à livrer ses promesses. Chaque nouveau dollar investi est dorénavant accompagné d'une meilleure confiance dans les résultats attendus de cette nouvelle base de données des produits. Cette confiance dans la réduction des ressources va se traduire en réduction de coûts opérationnels et donc libérer des fonds supplémentaires pour d'autres projets afin d'accroître la compétitivité de la plateforme de commerce électronique.

ANNEXE I

Commerce modeler

Le « *Commerce modeler* » est l'outil utilisé pour modéliser l'environnement d'affaires d'un client afin qu'il puisse effectuer la gestion du catalogue de ses produits à travers ses commerces. Ceux-ci, qui ont pignon sur rue, se retrouvent au plus bas de la hiérarchie (au 3e niveau) avec comme nom « *dependent scope* ». Chaque « *dependent scope* » est lié à un territoire ou regroupement qui s'intitule un « *sales scope* » (au 2e niveau). Finalement, un « *sales scope* » est lié au « *global* », le plus haut niveau de la hiérarchie (le 1er niveau), le siège social. Donc, un « *global* » détient un ou plusieurs « *sales scope* » qui eux à leur tour possèdent zéro ou plusieurs « *dependent scope* » (il est possible pour un « *sales scope* » de jouer le rôle du « *dependent scope* »).

La gestion des produits ainsi que de leurs attributs sont configurables uniquement au niveau du « *global* » et du « *sales scope* », soit les deux premiers niveaux. Toutefois, les prix demeurent configurables aux trois niveaux. Donc, un commerce peut gérer sa liste de prix indépendamment des autres, mais sur la description et les attributs des produits, ils doivent alors dépendre d'un niveau supérieur, un « *sales scope* » ou du « *global* ». Ce choix, entre les deux, est défini par ce qui est appelé l'héritage et est configurable. Si un « *dependent scope* » hérite d'une donnée, qui n'est pas trouvée au « *sales scope* », alors l'attribut ou sa valeur est extrait au « *global* ». Il existe la possibilité de définir des attributs et des valeurs qui ne sont qu'au niveau du « *sales scope* » et non disponibles au niveau du « *global* ». Ces configurations s'intitulent les « *Scoped Items* ».

Finalement, il existe les « *virtual scope* » qui eux n'ont comme fonction que de servir de regroupement. Il est impossible de définir de produits, ni de prix, ni d'attributs, ni de valeurs, absolument rien. Il peut y avoir entre zéros et plusieurs « *virtual scope* » entre le « *global* » et un « *sales scope* », tout comme il peut aussi y avoir entre zéros et plusieurs « *virtual scope* » entre un « *sales scope* » et un « *dependent scope* ». Dans le présent projet, étant donné que les « *virtual scope* » ne comportent aucune valeur, ils sont totalement exclus de la base de données des produits. Ils sont définis dans une autre base de données. Voici un exemple

sous deux formes différentes de ce que représente une hiérarchie de *scopes* pour une compagnie fictive :

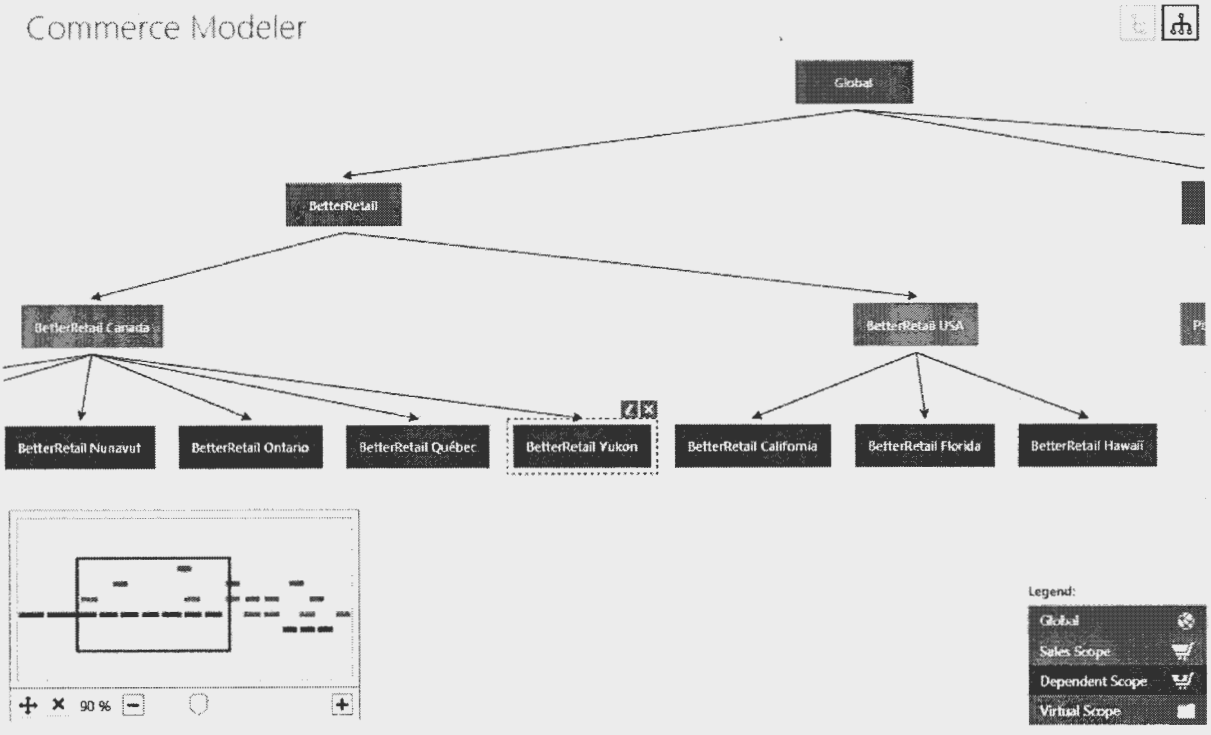


Figure I-1 - Visualisation de la hiérarchie à trois niveaux de la gestion des produits (tirée de l'interface utilisateur qui communique avec la plateforme)

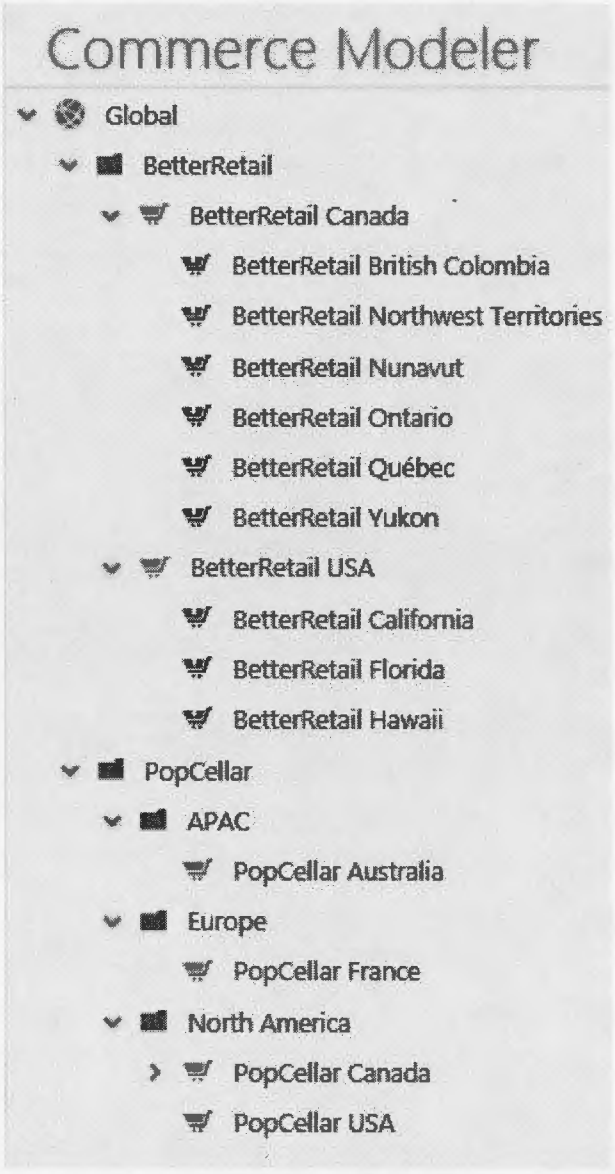


Figure I-2 - Différente vue pour la visualisation de la hiérarchie à trois niveaux de la gestion des produits (tirée de l'interface utilisateur qui communique avec la plateforme)

ANNEXE II

Tests et résultats de performance de la concaténation de chaînes de caractères

Cette section est dédiée au constat de la performance applicative nécessaire lors de la concaténation de chaînes de caractères par l'utilisation du SQL dynamique. Cette étape est survenue début 2017. Certaines actions d'atténuation ont été implémentées pour pallier temporairement le problème, mais aucune solution définitive.

Dans l'exemple qui suit, la requête originale a été modifiée pour introduire une boucle à l'intérieur du code et des variables additionnelles pour enregistrer le temps écoulé lors de la concaténation ainsi que lors de l'exécution finale du code concaténé. Ce choix est préférable, car il cible uniquement les parties nécessaires, soit la différence entre le temps de concaténation et le temps d'exécution. Il écarte aussi ce qui est externe afin de conserver le même état, itération après itération : durée de l'itérateur, durée d'invocation de la procédure stockée, durée de retour en arrière « *rollback* », qui, pour ce dernier, est quand même significatif, mais non représentatif de la réalité. Le test vise à effacer exactement le même enregistrement, mais sans en persister les changements. Dans la réalité, la quasi-totalité des requêtes n'opérera pas de retour en arrière, donc, il est nécessaire d'exclure cette partie de l'échantillonnage de durée.

Le résultat est sans équivoque. Autant par l'outil de profilage que par le code ajouté servant à capturer la durée de concaténation et la durée d'exécution.

- Nombre d'itérations : 1 000
- 98,89 % de la durée en concaténation
- 1,11 % en durée d'exécution

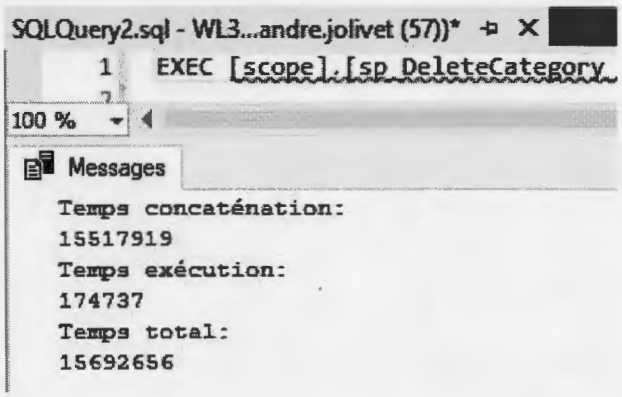


Figure II-1 - Temps d'exécution versus temps de concaténation

Durée de travail totale versus durée de travail en termes de CPU :

- 99,95 % (Colonne *CPU* / colonne *Duration* * 100) de la requête au total s'exécute en CPU pour une opération qui devrait être majoritairement en termes d'E/S et donc avoir un niveau d'utilisation du CPU moins élevé.

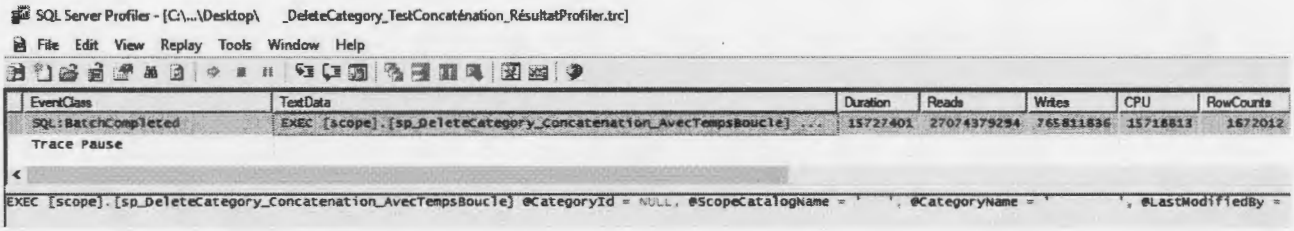


Figure II-2 - Profilage de l'exécution de la concaténation en SQL dynamique

Code source des tests de performance de la concaténation de chaînes de caractères

Le code source de la procédure stockée s'étend sur un peu plus de 800 lignes de code.

ANNEXE III

Exemples de code provenant de la base de données des produits

Les exemples de code sont au nombre de trois. Toutefois, compte tenu de leurs tailles, en lignes de code, ceux-ci ont été rendus disponibles que pour le jury.

scope.sp_GetProductDetail.sql

Cet exemple de code démontre la construction de la requête à exécuter que le SGBD doit préparer à chaque fois que la procédure stockée est invoquée. Le code s'étale sur un peu plus de 1 600 lignes.

scope.sp_DeleteCategory

Code original de la procédure stockée qui démontre l'effacement d'une catégorie à travers les niveaux hiérarchiques. Le résultat des tests de performance de cette procédure stockée est disponible à l'annexe II « *Tests et résultats de performance de la concaténation de chaînes de caractères* ». Le code s'étale sur un peu plus de 800 lignes.

scope.sp_DeleteCategoryExpanded

Ce code provient du résultat de la concaténation du SQL dynamique de la procédure stockée « scope.sp_DeleteCategory » lorsqu'une catégorie demande à être effacée. Le nombre de lignes généré est de 16 719.

ANNEXE IV

Liste des noms de documents produits

Regroupement des fichiers produits pour le présent projet. La première partie des noms des fichiers a été anonymisée.

Livrables attendus

1. « Nom de commerce » - BD Products v4 - Mesures.xslx
2. Code source
    - BDProductsV4EmptyInitialState.bak
    - BDProductsMigrationFromV3toV4.sql
3. Documentation
    - « Nom de commerce » - BD Products v4 - ERD business.vsdx
    - « Nom de commerce » - BD Products v4 - ERD normalized.vsdx

Livrables additionnels

4. « Nom de commerce » - BD Products v4 - Plans.docx
5. « Nom de commerce » - BD Products v4 - Gestion du projet.mpp
6. « Nom de commerce » - BD Products v4 - Présentation des résultats.pptx
7. « Nom de commerce » - BD Products v4 - Présentation du projet.pptx

Documents intermédiaires

8. « Nom de commerce » - BD Products v4 - Approbation initiale écrite.msg
9. « Nom de commerce » - BD Products v4 - Acceptation du projet.msg
10. « Nom de commerce » - BD Products v4 - Audit approbation.msg
11. MessageLog.txt
12. scope.sp_DeleteCategoryExpanded.sql
13. scope.sp_GetProductDetail.sql

ANNEXE V

Plan de projet

Le document a été anonymisé utilisant un descriptif au lieu du nom attendu. Les numéros de page de la table des matières réfèrent au numéro de page du document original et de celui-ci. Ce document était le référentiel de l'expression et de l'accord des exigences.

"Nom plateforme électronique™" Platform

Data and Search paper series by « Nom Compagnie »

Plan du projet : Base de données des produits v4

CE DOCUMENT

Cet ouvrage, écrit par Luc-André Jolivet, établit les grandes lignes du plan de projet afin de mener à terme le projet de la base de données des produits v4 pour la plateforme « nom plateforme ™ ».

SURVOL DE LA SITUATION

L'architecture de la base de données des produits n'est plus adéquate afin de satisfaire aux nouveaux besoins d'affaires. De plus, sa consommation en ressource matérielle impose de provisionner des machines de plus en plus fortes et coûteuses.

Table des matières

Historique des révisions

Version	Date	Pages, paragraphes ou figures	Description des changements	AME
Alpha	1 Septembre 2018		Version informelle	A
1.0	25 Janvier 2019		Version initiale	A

AME : A - Ajout, M - Modification, E - Effacement

Vision

Réduire les coûts opérationnels afin d'augmenter la marge de profits ainsi qu'améliorer l'offre de fonctionnalités entourant la gestion des produits.

Énoncé des travaux

Objectifs primaires

- Réduction de la consommation de CPU par la base de données des produits
- Rendre accessible les champs actif / inactif et visible / caché des produits et de leurs variants peu importe le *scope*

Objectifs secondaires

- Permettre plus de trois niveaux hiérarchiques
- Permettre jusqu'à 30 000 *scopes*
- Étendre la flexibilité des deux premiers *scopes* au troisième (permettre les mêmes fonctionnalités au niveau du « dépendant » qu'au niveau du « sale »)

Critères de succès

- Éliminer le SQL dynamique de la base de données des produits
- Les champs « IsActive » & « IsHidden » disponibles pour l'applicatif (traçabilité en rouge)
- Établir une hiérarchie de plus de trois niveaux de *scopes* (traçabilité en vert)
- La base de données peut contenir 30 000 *scopes* (traçabilité en mauve)
- La base de données permet de modifier un attribut de produit pour un *scope* dépendant (traçabilité en orange)
- Obtenir des mesures d'écart en utilisation CPU

Portée

Inclus : Les exigences primaires et secondaires, autant d'affaires que techniques.

Exclus du cadre de la synthèse académique, mais inclus et nécessaire dans la suite du projet :

- L'inventaire des produits
- Le code T-SQL ajusté de la version 3 vers la version 4
- L'historisation externe (la version « Archive » & « Data Warehouse » de la base de données)
- Migration des tests d'intégration propre à la base de données des produits
- Mise à jour des outils d'aide SQL

Mesures de performance applicative

Les procédures stockées suivantes serviront de témoin pour évaluer la performance applicative calculée en millisecondes pour le temps d'exécution. Les trois premières ciblent un unique *scope*. ItemPriceListWithInheritance est la plus lourde sur le système, suivie des deux procédures stockées représentant les détails des produits et variants qui sont aussi significatives en termes de ressources nécessaires. Finalement, le DeleteCategory pour représenter le code transversal ayant besoin de s'appliquer à tous les *scopes* et aussi parce qu'il est l'élément initial identifié pour la perte de performance de la concaténation de chaînes de caractères.

- ItemPriceListWithInheritance
- GetProductDetail
- GetVariantDetail
- DeleteCategory

Ces mesures seront effectuées sur trois clients : Client 1, Client 2, Client 3. Davantage de clients pourraient être traités si le temps le permet.

Valeur, coût ou bénéfices anticipés

- Offrir de nouvelles fonctionnalités aux clients
- Éliminer de coûteux serveurs ou en diminuer la taille afin de réduire les coûts de location des machines

Risques

Le plus important risque technique encouru est l'impossibilité d'atteindre une performance satisfaisante une fois le SQL dynamique retiré. Pour ce faire, le projet démarrera en preuve de concept et si celle-ci est satisfaisante, le projet poursuivra. Dans le cas inverse, il y aura concertation afin de déterminer les prochaines étapes.

Le second risque est celui de l'indisponibilité de l'unique DBA / Architecte travaillant sur le projet.

- Sous-traiter, à l'interne par l'entremise d'autres lignes d'affaires, des projets actuels
- Travailler de la maison afin de réduire les demandes journalières
- Travailler la fin de semaine
- Engager un autre DBA

Livrables attendus et modalités

Tout type de livrables sera disponible dans un répertoire partagé sur OneDrive© et ils seront disponibles en tout temps. Un courriel vous parviendra avec le lien d'accès aux personnes impliquées dans le projet lorsque leurs droits seront octroyés.

Dès qu'ils sont prêts :

- Les diagrammes de modélisation suivants :
  - *« Nom de commerce » - BD Products v4 - ERD business.vsdx*
  - *« Nom de commerce » - BD Products v4 - ERD normalized.vsdx*
    - En caractère rouge les nouveaux champs ajoutés des exigences

À la fin du projet (au-delà de la partie académique uniquement) :

- Une base de données minimale au format SQL compressée, version 2017 (à l'image des autres bases de données de la plateforme) comportant les éléments initiaux attendus par la plateforme.
  - *BDProductsV4EmptyInitialState.bak*
- Un script unique de migration SQL des données de la version actuelle de la base de données des produits vers la nouvelle pour tous les clients opérant sous SQL Server 2014 ou 2016.
  - *BDProductsMigrationFromV3toV4.sql*
- Un document de mesure démontrant l'écart d'utilisation CPU entre v3 et v4
  - *« Nom de commerce » - BD Products v4 - Mesures.xslx*

Livrables additionnels

Les livrables additionnels sont les documents principalement axés sur la gestion de projet. Les développeurs et la maintenance n'en auront pas une nécessité pour mener à bien leurs activités avec la nouvelle base de données. Mêmes modalités que pour les livrables attendus.

- Ce document.
  - « *Nom de commerce* » - *BD Products v4 - Plans.docx*
- Le document de progression de projet utilisant Project professional 2016.
  - « *Nom de commerce* » - *BD Products v4 - Gestion du projet.mpp*
- La présentation du projet
  - « *Nom de commerce* » - *BD Products v4 - Présentation du projet.pptx*
- La présentation des résultats
  - « *Nom de commerce* » - *BD Products v4 - Présentation des résultats.pptx*

Ressources matérielles additionnelles

L'utilisation d'un laptop « Nom Compagnie » est suffisant pour le développement. Pour l'établissement du temps requis à la migration et d'un référentiel de base, l'utilisation temporaire d'une VM dans l'infonuagique serait préférable.

Ressources humaines et composition des équipes du projet

- Président pour la partie performance infonuagique
- Directeur du commerce numérique pour ce qui est des fonctionnalités d'affaires
- Luc-André Jolivet pour la partie technique
- Auditeur, développeur senior à titre d'auditeur des résultats des mesures

Planification des tâches à exécuter

Pour ce qui est des tâches, leurs durées, date de début et fin, l'estimation des efforts, rôles et responsabilités, veuillez-vous référer au document « *"Nom de commerce" - BD Products v4 - Gestion du projet.mpp* ». Tous les artefacts (code et documents) sont stockés sur le réseau interne pour bénéficier des sauvegardes automatiques et permettre de les partager en tout temps.

Répercutions de l'intégration de la base de données

Les services suivants seront développés et maintenus par le produit. Ils seront nécessaires au bon fonctionnement de la solution dans son cadre global. Ils feront partie de la suite du projet si les résultats préliminaires sont satisfaisants.

Service additionnel pour le pivot des attributs

Parce que le SQL dynamique sera retiré, il est donc nécessaire que le pivot des attributs soit transféré vers le code applicatif parce que l'engin SQL ne permet pas d'effectuer cette opération sur une liste d'attributs de longueur arbitraire sans passer par du SQL dynamique.

Service additionnel de dictionnaire de données

La conception actuelle de la plateforme identifie ses objets par des chaînes de caractères. Or, en base de données, ce type de conception est fortement déconseillé parce qu'il provoque plusieurs effets secondaires autant au niveau CPU, I/O, RAM, statistiques et de sécurité.

Tous les identifiants seront des entiers et donc un service additionnel de type « dictionnaire » nécessitera de s'intégrer entre la base de données et la plateforme pour effectuer la conversion entre les identifiants en chaînes de caractères et leurs entiers correspondants. Il a été testé qu'utiliser une chaîne de caractères avec le jeu de données que la plateforme utilise est ~ 63 % plus coûteux en ressource CPU qu'un entier.

Changements impromptus

- Accepté par le directeur du commerce numérique : Le nom « *Relation* » est préférable à « *Merchandising* »
- Accepté par le directeur du commerce numérique : Déplacer *PriceList Category* et *PriceList type* dans leurs propres tables au lieu de les stocker dans les *LookUp*
- Accepté par directeur développement : Les attributs systèmes ont un ID inférieur à 0, ceux des clients, supérieur à 0. Redistribuer les IDs des entités
- Clarification: Une *PriceList* détient un nom unique (pas par *Scope* et donc est pour l'ensemble des *scopes*), mais ses dates d'activation sont dépendantes par *scope*

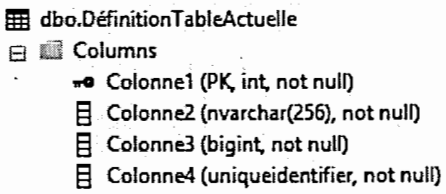
(Il n'y a pas eu de demandes refusées à ce jour)

ANNEXE VI

Autres solutions envisagées

Dans les solutions envisagées, la première que j'ai entreprise a été de mesurer la consolidation des *scopes* « regroupement des commerces » sous un seul jeu de table. Voici un exemple de la transformation d'une table : la description des colonnes demeure identique sauf pour l'ajout du discriminant « *ScopeID* » dans la clé primaire (voir la Figure 3.2). Cette façon de faire est simple, éliminerait l'utilisation du SQL dynamique tout en minimisant la refonte du code source des procédures stockées.

Avant



Après

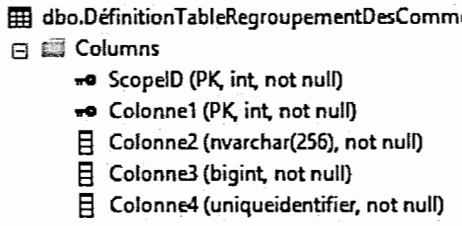


Figure VI-1 - Exemple de structure pour la consolidation des commerces

En second, la compagnie mère opère près d'une vingtaine de lignes d'affaires dont certaines utilisent un catalogue de produits. J'ai donc rencontré le personnel gravitant autour des bases de données de produits de ces lignes d'affaires afin d'en prendre connaissance et d'évaluer si leur solution était viable pour la plateforme de commerce électronique.

Troisièmement, j'ai lu ce que la littérature enseigne à propos des bases de données, mais plus particulièrement en ciblant le contexte du commerce électronique desservant un catalogue de produits.

Finalement, en dernière possibilité, j'ai sélectionné la solution retenue, soit de reconceptualiser la base de données des produits, en somme, de reconfigurer les relations (voir la sous-section 4.1.2 pour une description détaillée de cette solution ainsi que les raisons des choix qui ont été faits).

ANNEXE VII

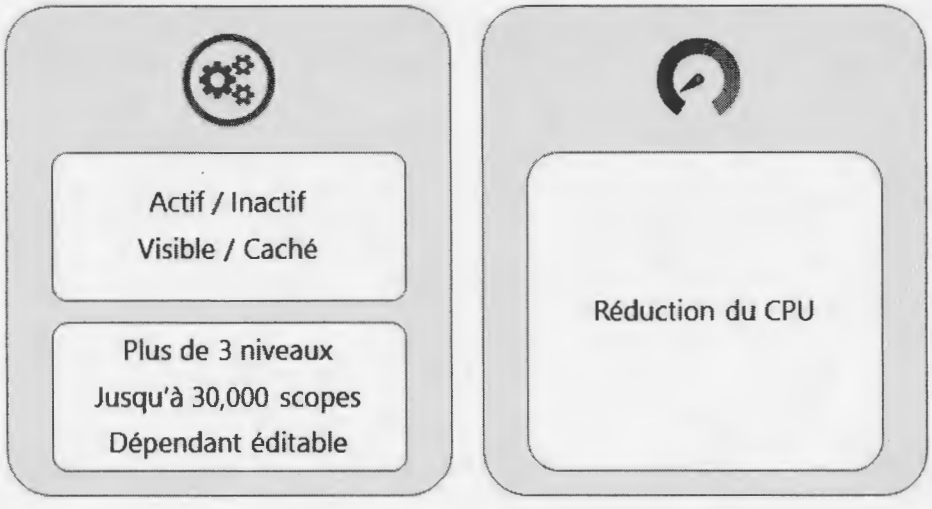
Présentation PowerPoint du projet

Les noms des clients et individus ont volontairement été retirés aux fins de confidentialité.

Aperçu de la présentation

- Objectifs de la base de données
- Portée académique et corporative
- Mesures de performance
- Impacts sur l'applicatif
- Documents & suivi de progression

Exigences



Délimitation du travail jusqu'à la production

Académique (quantification de réduction CPU)

- Gestion de projet
- Migration des entités de base
  - Catégories / Produits / Variants
  - Listes de prix
  - Prix des items
  - Relations (Merchandising)
- Mesures de performances

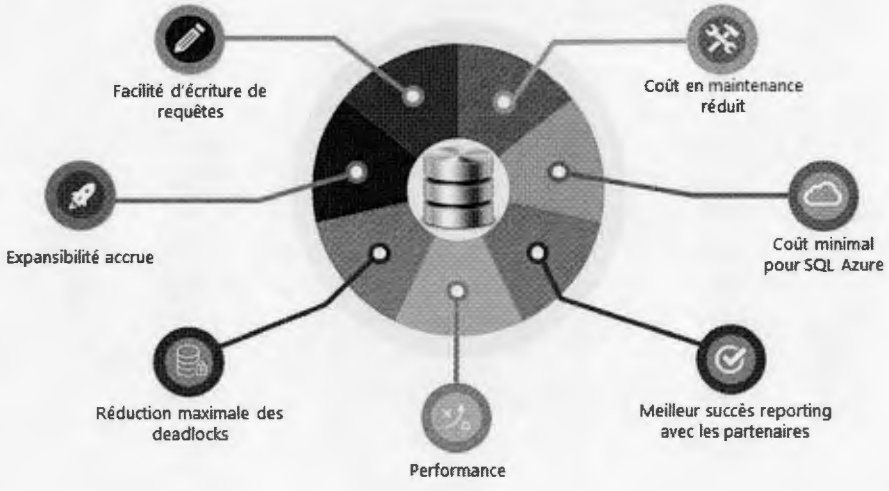
Corporatif (poursuite du projet)

- Migration de l'inventaire
- Adaptation des procédures stockées
- Adaptation aux modifications PIM v4
- Services de dictionnaire
- Intégration des tests avec la plateforme
- Adaptation à l'ODS
- Tests LoadImpact
- Formation & documentation aux développeurs

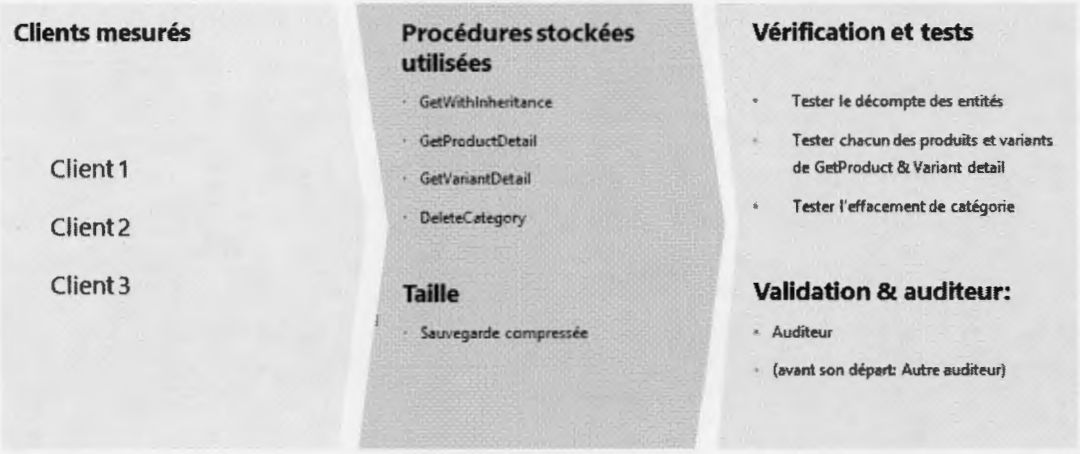
Mise en production

- Priorisation des clients v4
- Planification du downtime

Objectifs corporatifs additionnels



Mesures entre v3 et v4



Impacts sur l'applicatif

- Service de pivot des attributs
- Service de dictionnaire de données

Documents et suivis

	1 Présentation	2 Technique	3 Mesure	4 Suivi
Documentation	• Présentation du projet.pptx (Ce document) • Présentation des résultats.pptx	• ERD business.vsdx • ERD normalized.vsdx	• Mesures.xlsx	Gestion du projet.mpp Plans.docx
Emplacement	Partagés et disponibles en tout temps sur OneDrive 			

Questions ?

Références

- http://www.simplitfy.com/images/icon-performance.png
- https://jsreport.net/img/api.png
- http://www.iconhot.com/icon/png/database/512/database-1-1.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/c/c4/Ambox_blue_question.svg/900px-Ambox_blue_question.svg.png

ANNEXE VIII

Détail des tâches

	Intrants	Task Name	Extrants	Work	Predecessors
0		**Activités de la méthodologie**		**645 hrs**	
1		**Planification**		**93 hrs**	
2	Besoins d'affaires	**Définir le projet**	Énoncé des travaux	**8 hrs**	
3		Éliciter les exigences		8 hrs	
4	Énoncé des travaux	**Rechercher des solutions**	Solution retenue	**60 hrs**	2
5		Analyser les exigences		4 hrs	2
6		**Analyser des alternatives**		**56 hrs**	5
7		Reconceptualiser la base de données		8 hrs	5
8		Interroger les autres lignes d'affaires		4 hrs	5
9		S'informer dans la littérature		32 hrs	5
10		Regroupement des commerces		12 hrs	5
11	Solution retenue	**Planifier le projet**	Plan de projet	**23 hrs**	4
12		Documenter l'énoncé des travaux		1 hr	4
13		Documenter le plan de projet		1 hr	12
14		Documenter le plan de configuration		1 hr	13
15		Documenter les modalités de livraison		1 hr	14
16		Documenter les objectifs		1 hr	15
17		Documenter les critères de succès		1 hr	16
18		Documenter les ressources humaines et matérielles		1 hr	17
19		Documenter les risques		1 hr	18
20		Documenter la stratégie de contrôle de versions		1 hr	19
21		**Tâches**		**6 hrs**	20
22		Documenter l'énumération des tâches		2 hrs	20
23		Documenter les ressources par tâches		1 hr	22
24		Estimer les tâches et leurs durées		1 hr	23
25		Estimer le début et la fin de chaque tâche		1 hr	24
26		Calculer les efforts et coûts estimés		1 hr	25
27		Documenter les coûts estimés		1 hr	21
28		Lister les impacts en intégration		1 hr	21

	Intrants	Task Name	Extrants	Work	Predecessors
29		**Documenter le plan d'AQ**		**4 hrs**	**28**
30		Documenter le plan de mesures		2 hrs	28
31		Documenter le plan de tests		2 hrs	30
32		Documenter le plan de rapports de suivis		2 hrs	29
33	Plan de projet	**Présenter le plan**	Plan révisé / approuv	2 hrs	11
34		**Réalisation**		**520 hrs**	**1**
35	Plan de projet et solut	**Modéliser la preuve de concept**	Modélisation d'affair	**54 hrs**	**1**
36		Produire le ERD d'affaires		8 hrs	1
37		Valider le ERD d'affaires		6 hrs	36
38		Produire le ERD normalisé		40 hrs	37
39	Modélisations	**Créer la base de données**	Base de données	**72 hrs**	**35**
40		Produire la définition de la base de données		40 hrs	35
41		Produire le script de standardisation de la nomenclature des objets		8 hrs	40
42		Produire le script d'initialisation		8 hrs	41
43		Développer les outils d'aide aux développeurs (Vues)		8 hrs	42
44		Archivage des objets obsolètes de la conception précédente		8 hrs	43
45	Base de données	**Migrer les données**	Bases de données mi	**350 hrs**	**39**
46		Logging		8 hrs	39
47		Migrer les données périphériques de base		24 hrs	39
48		Migrer les listes de prix		24 hrs	47
49		Migrer les attributs		24 hrs	48
50		Migrer les catégories		24 hrs	49
51		Migrer les produits		32 hrs	50
52		Migrer les variants		40 hrs	51
53		Créer les origines		8 hrs	52
54		Migrer les relations		12 hrs	53
55		Gérer les exceptions de l'architecture précédente		32 hrs	54
56		**Tests d'intégration**		**122 hrs**	**55**
57		Test des vues		2 hrs	55
58		Test du décompte des entités		8 hrs	55
59		Test d'effacement en cascade		8 hrs	55

	Intrants	Task Name	Extrants	Work	Predecessors
60		**Test des données GetProductDetail**		**42 hrs**	**55**
61		Traduire le code en v4		2 hrs	55
62		Tester chaque produit		40 hrs	61
63		**Test des données GetVariantDetail**		**62 hrs**	**55**
64		Traduire le code en v4		2 hrs	55
65		Tester chaque variant		60 hrs	64
66		Auditer les mesures à venir		8 hrs	60,63
67	Bases de données actu	**Mesurer les écarts de CPU**	Résultats de mesures	**36 hrs**	**66**
68		**Mesurer la performance**		**36 hrs**	**66**
69		**GetWithInheritance**		**16 hrs**	**66**
70		Mesurer au global		8 hrs	66
71		Mesurer au dépendant		8 hrs	66
72		**GetProductDetail**		**6 hrs**	**66**
73		Mesurer au global		3 hrs	66
74		Mesurer au dépendant		3 hrs	66
75		**GetVariantDetail**		**6 hrs**	**66**
76		Mesurer au global		3 hrs	66
77		Mesurer au dépendant		3 hrs	66
78		**DeleteCategory**		**8 hrs**	**66**
79		Mesurer au global		4 hrs	66
80		Mesurer au dépendant		4 hrs	66
81		**Évaluation et contrôle**		**28 hrs**	**1**
82	Progression des travau	Mettre à jour le plan des tâches	Plan des tâches mis à	24 hrs	1
83	Demande de changement (DDC)	Analyser et évaluer la demande de DDC	DDC approuvée ou rejetée et plan	4 hrs	1
84		**Clôture du projet**		**4 hrs**	**34**
85	Résultats de mesures	Présenter les résultats des mesures	Continuité du projet ou non	4 hrs	34

Figure VIII-1 - Détail des tâches

ANNEXE IX

Plan de mesure

Chaque mesure de base a été appliquée sur chacune des procédures stockées visées pour chacune des bases de données, actuelle et nouvelle, des trois clients ciblés afin de mesurer les différences. Rappel des quatre procédures stockées mesurées :

- *ItemPriceListWithInheritance*
- *GetProductDetail*
- *GetVariantDetail*
- *DeleteCategory*

Afin d'alléger la quantité de tableaux, chaque tableau (excepté celui de la taille des données) représentera la combinaison de chacune des quatre procédures stockées, la passation sur la base de données actuelle et la nouvelle, et ce, pour chacun des clients souhaités.

Mesures de base

Tableau IX-1 - Nombre d'unités de CPU par procédure stockée et par client

<table>
<tr><td colspan="5">Mesure de base</td></tr>
<tr><td># 1</td><td colspan="2">Mesure
Nombre d'unités d'utilisation du CPU</td><td>Applicabilité
4 SPs x 2 BDs x Nbr Clients</td><td>U de M
Unité « CPU » du Profiler©</td><td>Précision
1 Unité</td></tr>
<tr><td>Qui mesure
Moi</td><td>Source de la mesure
« CPU » du Profiler ©</td><td>Où persister le résultat
Chiffrier des mesures</td><td>Outil
Profiler ©</td><td colspan="2">Moment
Après les vérifications de l'audit selon le détail des tâches</td></tr>
<tr><td colspan="2">Procédure de collecte
Démarrer et configurer l'outil Profiler©

Exécuter dans SSMS la procédure stockée

Collecter les données</td><td colspan="4">Assurance qualité
S'assurer qu'une itération de boucle s'exécute et que l'outil enregistre les données correctement.

Commentaires</td></tr>
</table>

Tableau IX-2 - Temps d'exécution par procédure stockée et par client par le profiler

Mesure de base					
# 2	**Mesure** Temps d'exécution au niveau du SGBD		**Applicabilité** 4 SPs x 2 BDs x Nbr Clients	**U de M** Nanosecondes	**Précision** 1 Ns
Qui mesure Moi	**Source de la mesure** « Duration » du Profiler ©	**Où persister le résultat** Chiffrier des mesures	**Outil** Profiler ©	**Moment** Après les vérifications de l'audit selon le détail des tâches	
Procédure de collecte Démarrer et configurer l'outil Profiler© Exécuter dans SSMS la procédure stockée Collecter les données		**Assurance qualité** S'assurer qu'une itération de boucle s'exécute et que l'outil enregistre les données correctement. **Commentaires**			

Tableau IX-3 - Temps d'exécution « single-thread »

Mesure de base					
# 3	**Mesure** Temps d'exécution global « single-thread »		**Applicabilité** 4 SPs x 2 BDs x Nbr Clients	**U de M** Millisecondes	**Précision** 1 Ms
Qui mesure Moi	**Source de la mesure** SSMS©	**Où persister le résultat** Chiffrier des mesures	**Outil** SSMS©	**Moment** Après les vérifications de l'audit selon le détail des tâches	
Procédure de collecte Exécuter dans SSMS la procédure stockée Collecter les données		**Assurance qualité** S'assurer qu'une itération de boucle s'exécute et que l'outil enregistre les données correctement. **Commentaires**			

Tableau IX-4 - Temps d'exécution « multi-thread »

<table>
<tr><td colspan="6">Mesure de base</td></tr>
<tr><td># 4</td><td colspan="2">Mesure
Temps d'exécution « multi-thread », « Actual Seconds / Iteration »</td><td>Applicabilité
2 SPs x 2 BDs x Nbr Clients</td><td>U de M
Seconde</td><td>Précision
1,00 Sec</td></tr>
<tr><td>Qui mesure
Moi</td><td>Source de la mesure
SQLQueryStress</td><td>Où persister le résultat
Chiffrier des mesures</td><td>Outil
SQLQueryStress</td><td colspan="2">Moment
Après les vérifications de l'audit selon le détail des tâches</td></tr>
<tr><td colspan="3">Procédure de collecte
Configurer l'outil pour :
1 000 itérations
8 Threads

Collecter les données</td><td colspan="3">Assurance qualité
Exécuter une seule itération avec un seul thread puis à l'aide du Profiler© comparer si les résultats concordent entre les deux outils.

Commentaires
L'outil ne permet pas les « User defined table types » donc uniquement « GetProductDetail » et « GetVariantDetail » seront testés. « DeleteCategory » sera aussi laissé de côté puisque ce n'est pas une opération courante et aussi par son temps excessif pour la base de données actuelle.</td></tr>
</table>

Tableau IX-5 - Taille compressée des données

<table>
<tr><td colspan="6">Mesure de base</td></tr>
<tr><td># 5</td><td colspan="2">Mesure
Taille compressée des données</td><td>Applicabilité
2 BDs x Nbr Clients</td><td>U de M
Mégaoctets</td><td>Précision
1,00 Mo</td></tr>
<tr><td>Qui mesure
Moi</td><td>Source de la mesure
Taille du fichier des sauvegardes</td><td>Où persister le résultat
Chiffrier des mesures</td><td>Outil
Windows© file explorer</td><td colspan="2">Moment
Après les vérifications de l'audit selon le détail des tâches</td></tr>
<tr><td colspan="3">Procédure de collecte
Effectuer une sauvegarde compressée après compaction des données (fillfactor = 100).

Collecter les données</td><td colspan="3">Assurance qualité
S'assurer de conserver dans les bases de données que les entités transposées de la nouvelle version.

Commentaires</td></tr>
</table>

Tableau IX-6 - Nombre d'unités de « Logical Reads » par procédure stockée et par client

Mesure de base					
# 6	**Mesure** Nombre de « Logical Reads »		**Applicabilité** 4 SPs x 2 BDs x Nbr Clients	**U de M** Unité « Logical Reads » du Profiler©	**Précision** 1 Unité
Qui mesure Moi	**Source de la mesure** « Logical Reads » du Profiler ©	**Où persister le résultat** Chiffrier du DBA	**Outil** Profiler ©	**Moment** Après les vérifications de l'audit selon le détail des tâches	
Procédure de collecte Démarrer et configurer l'outil Profiler© Exécuter dans SSMS la procédure stockée Collecter les données		**Assurance qualité** S'assurer qu'une itération de boucle s'exécute et que l'outil enregistre les données correctement. **Commentaires**			

Tableau IX-7 - Nombre d'unités de « Writes » par procédure stockée et par client

Mesure de base					
# 7	**Mesure** Nombre de « Writes »		**Applicabilité** 4 SPs x 2 BDs x Nbr Clients	**U de M** Unité « Writes » du Profiler ©	**Précision** 1 Unité
Qui mesure Moi	**Source de la mesure** « Writes » du Profiler ©	**Où persister le résultat** Chiffrier du DBA	**Outil** Profiler ©	**Moment** Après les vérifications de l'audit selon le détail des tâches	
Procédure de collecte Démarrer et configurer l'outil Profiler© Exécuter dans SSMS la procédure stockée Collecter les données		**Assurance qualité** S'assurer qu'une itération de boucle s'exécute et que l'outil enregistre les données correctement. **Commentaires**			

Tableau IX-8 - Nombre d'unités de mémoire vive par procédure stockée et par client

<table>
<tr><td colspan="5">Mesure de base</td></tr>
<tr><td># 8</td><td colspan="2">Mesure
Quantité de mémoire vive utilisée dans la cache SQL</td><td>Applicabilité
4 SPs x 2 BDs x Nbr Clients</td><td>U de M
Mégaoctets</td><td>Précision
1,00 Mo</td></tr>
<tr><td>Qui mesure
Moi</td><td>Source de la mesure
DMVs</td><td>Où persister le résultat
Chiffrier du DBA</td><td>Outil
DMVs</td><td colspan="2">Moment
Après les vérifications de l'audit selon le détail des tâches</td></tr>
<tr><td colspan="3">Procédure de collecte
Vider la cache mémoire de SQL

Utiliser les DMVs pour calculer la mémoire vive utilisée.

Collecter les données</td><td colspan="3">Assurance qualité
S'assurer que la cache SQL a bien été vidée.

Commentaires
À effectuer avec une cache à froid et une déjà initialisée dans SQL

Un seul jeu de valeur par procédure stockée sera utilisé pour reproduire exactement la même taille des données pour la nouvelle base de référence en v4.</td></tr>
</table>

Mesures dérivées

Tableau IX-9 - Pourcentage de différence des procédures transactionnelles pour l'utilisation CPU

Mesure dérivée					
	Mesure		**Formule**	**U de M**	**Précision**
# 1	Pourcentage de différence entre v3 et v4 pour l'utilisation CPU, le temps d'exécution « single et multi-thread » pour GetProductDetail, GetVariantDetail et ItemPriceListWithInheritance		(Somme Mesure v3 - Somme Mesure v4) / Somme Mesure v4 * 100	Pourcentage	1,00 %
Responsable	**Consommateur**	**Où persister le résultat**	**Présentation des résultats**	**Fréquence**	
Moi	Direction	Chiffrier des mesures	Amélioration ou dégradation en %	Une fois toutes les mesures de base collectées	
Procédure d'analyse		**Actions possibles selon les résultats**			
Positif = amélioration v4 Négatif = dégradation par rapport à v3		À évaluer et déterminer avec la direction.			

Tableau IX-10 - Pourcentage de différence de la procédure non transactionnelle pour l'utilisation CPU

Mesure dérivée					
	Mesure		**Formule**	**U de M**	**Précision**
# 2	Pourcentage de différence entre v3 et v4 pour l'utilisation CPU, le temps d'exécution « single-thread » pour DeleteCategory		(Somme Mesure v3 - Somme Mesure v4) / Somme Mesure v4 * 100	Pourcentage	1,00 %
Responsable	**Consommateur**	**Où persister le résultat**	**Présentation des résultats**	**Fréquence**	
Moi	Direction	Chiffrier des mesures	Amélioration ou dégradation en %	Une fois toutes les mesures de base collectées	
Procédure d'analyse		**Actions possibles selon les résultats**			
Positif = amélioration v4 Négatif = dégradation par rapport à v3		À évaluer et déterminer avec la direction.			

Tableau IX-11 - Pourcentage de différence des autres mesures

Mesure dérivée					
	Mesure		**Formule**	**U de M**	**Précision**
# 3	Pourcentage de différence entre v3 et v4 pour chacune des mesures de base restantes		(Somme Mesure v3 - Somme Mesure v4) / Somme Mesure v4 * 100	Pourcentage	1,00 %
Responsable	**Consommateur**	**Où persister le résultat**	**Présentation des résultats**	**Fréquence**	
Moi	Moi	Chiffrier du DBA	Amélioration ou dégradation en %	Une fois toutes les mesures de base collectées	
Procédure d'analyse		**Actions possibles selon les résultats**			
Positif = amélioration v4 Négatif = dégradation par rapport à v3		À prendre en considération lors de la poursuite du projet.			

ANNEXE X

Dégradation de la performance

Parmi les solutions non retenues, l'une d'elles était la consolidation de magasins, voire annexe VI « *Autres solutions envisagées* » pour les détails. Si cette approche, la consolidation, avait donné des résultats satisfaisants, beaucoup de temps aurait pu être sauvé puisque la majeure partie de la base de données aurait eu peu de modifications afin de s'y ajuster.

La consolidation de commerces testée sur la requête taxant le plus la base de données des produits a démontré qu'au lieu de réduire son empreinte CPU, il l'augmenta malgré le fait que le SQL dynamique était retiré. Le cas standard d'une requête, ayant la moyenne des entrées pour le commerce typique servant de référence lors de tests par les développeurs, a démontré une augmentation de 29 % de la charge CPU. Dans le pire des cas, la charge CPU a atteint 2 383 % d'augmentation. (Pour plus de détails, voir le fichier externe « Mesures.xlsx » sous le volet « Consolidation »). Donc, même lorsque le SQL dynamique avait été retiré, une modélisation inadéquate jumelée avec une augmentation de la taille du jeu de données dans des tables consolidées a démontré que cette solution accaparait davantage de ressource CPU, ce qui allait à l'encontre de l'objectif premier de la réduction de celui-ci.

ANNEXE XI

Journal de migration

Le journal de migration liste chaque étape de la migration des données en plus d'y indiquer le temps d'exécution respectif ainsi que cumulatif pour chaque élément. Des tabulations existent afin de démarquer des regroupements autant au niveau des entités que du temps de migration dudit regroupement. Dans l'exemple ci-joint, il n'y a qu'un *scope* et le nom du client a été retiré. L'intérêt d'utiliser les lignes « *Start* » était pour le débogage. Si une erreur survenait, la ligne « *End* » de la commande en erreur n'était pas atteinte et donc pas affichée. Donc, pour obtenir exactement la bonne commande, il ne suffisait que de copier l'erreur et chercher le texte dans le script de migration pour obtenir immédiatement la commande qui provoquait une erreur. Cette façon de faire a été implémentée, car l'outil SSMS© n'est pas en mesure de tracer correctement les lignes de code lorsqu'il y a du SQL dynamique imbriqué.

```
Discovered customer: UndisclosedCustomer
Local time: 2019-06-13 11:36:14
------------------------------------------------------------------------

Debugging: disabling IsSystemAttribute check constraint
Debugging: Migration UTC Time: 2019-06-13 15:36:14

Warning: Foreign key 'FK_Product_VariantPriceList_ScopeID_ProductID_VariantID_Product_VariantProduct' on table 'VariantPriceList'
referencing table 'VariantProduct' was disabled as a result of disabling the index
'UX_Product_VariantProduct_ScopeID_ProductID_VariantID'.

Elapsed time for tables out of scope loop: 1093 ms

Start: Scope migration 1 / 2 : Global (Global)
  Start: PriceList
    Start: PriceList validation
    End: PriceList validation 33 ms
    Start: Extracting pricelist author info from v3 and persist in v4 if not found
    End: Extracting pricelist author info from v3 and persist in v4 if not found: 47 ms
    Start: *_PRICELIST -> PriceList.PriceList
```

```
  End: *_PRICELIST -> PriceList.PriceList: 110 ms
  Start: *_PRICELIST_LOCALIZE -> PriceList.PriceListLocalize
  End: *_PRICELIST_LOCALIZE -> PriceList.PriceListLocalize: 46 ms
End: PriceList: 236 ms
Start: Category, Product, Variant Item migration
  Start: *_ITEM WHERE ItemDiscriminator = 'Category' -> Product.Category
    Start: Extracting category info and creating attribute unicity from v3 current scope
    End: Extracting category info and creating attribute unicity from v3 current scope: 29 ms
    Start: Extracting current scope new unique categories which has not been extracted yet from any processed scope
    End: Extracting current scope new unique categories which has not been extracted yet from any processed scope: 33 ms
    Start: Inserting new category origin
    End: Inserting new category origin: 13 ms
    Start: Extracting category author info from v3 and persist in v4 if not found
    End: Extracting category author info from v3 and persist in v4 if not found: 30 ms
    Start: Persisting in v4 current v3 scope new unique categories with at least a name or a different attribute value
    End: Persisting in v4 current v3 scope new unique categories with at least a name or a different attribute value: 47 ms
    Start: Keeping cumulative v3 & v4 category id link
    End: Keeping cumulative v3 & v4 category id link: 16 ms
    Start: Persisting in v4 category custom attributes values from v3
    End: Persisting in v4 category custom attributes values from v3: 63 ms
    Start: Persisting in v4 localized category attributes values from v3
    End: Persisting in v4 localized category attributes values from v3: 46 ms
    Start: Persisting current scope link between category and scope -> Product.CategoryScope
    End: Persisting current scope link between category and scope -> Product.CategoryScope: 33 ms
  End: *_ITEM WHERE ItemDiscriminator = 'Category' -> Product.Category: 313 ms
  Start: *_ITEM WHERE ItemDiscriminator = 'Product' -> Product.Product
    Start: Extracting product info and creating attribute unicity from v3 current scope
    End: Extracting product info and creating attribute unicity from v3 current scope: 90 ms
    Start: Extracting current scope new unique products which has not been extracted yet from any processed scope
    End: Extracting current scope new unique products which has not been extracted yet from any processed scope: 96 ms
    Start: Inserting new product origin
    End: Inserting new product origin: 13 ms
    Start: Extracting product author info from v3 and persist in v4 if not found
    End: Extracting product author info from v3 and persist in v4 if not found: 33 ms
    Start: Persisting in v4 current v3 scope new unique products with at least a name or a different attribute value
    End: Persisting in v4 current v3 scope new unique products with at least a name or a different attribute value: 157 ms
    Start: Keeping cumulative v3 & v4 product id link
    End: Keeping cumulative v3 & v4 product id link: 13 ms
    Start: Persisting in v4 product custom attributes values from v3
    End: Persisting in v4 product custom attributes values from v3: 63 ms
    Start: Persisting in v4 localized product attributes values from v3
    End: Persisting in v4 localized product attributes values from v3: 46 ms
    Start: Persisting current scope link between product and scope -> Product.ProductScope
```

```
End: Persisting current scope link between product and scope -> Product.ProductScope: 63 ms
Start: *_ITEM WHERE ItemDiscriminator = 'Variant' -> Product.Variant
  Start: Extracting variant info and creating attribute unicity from v3 current scope
  End: Extracting variant info and creating attribute unicity from v3 current scope: 76 ms
  Start: Extracting current scope new unique variants which has not been extracted yet from any processed scope
  End: Extracting current scope new unique variants which has not been extracted yet from any processed scope: 80 ms
  Start: Inserting new variant origin
  End: Inserting new variant origin: 0 ms
  Start: Extracting variant author info from v3 and persist in v4 if not found
  End: Extracting variant author info from v3 and persist in v4 if not found: 30 ms
  Start: Persisting in v4 current v3 scope new unique variants with at least a name or a different attribute value
  End: Persisting in v4 current v3 scope new unique variants with at least a name or a different attribute value: 127 ms
  Start: Keeping cumulative v3 & v4 variant id link
  End: Keeping cumulative v3 & v4 variant id link: 30 ms
  Start: Persisting in v4 variant custom attributes values from v3
  End: Persisting in v4 variant custom attributes values from v3: 63 ms
  Start: Persisting in v4 localized variant attributes values from v3
  End: Persisting in v4 localized variant attributes values from v3: 47 ms
  Start: Linking current scope product to current variant
    Start: Populating temp table with new variant
    End: Populating temp table with new variant: 30 ms
    Start: Persisting link between product and new variants -> VariantProduct
    End: Persisting link between product and new variants -> VariantProduct: 63 ms
  End: Linking current scope product to current variant: 93 ms
End: *_ITEM WHERE ItemDiscriminator = 'Variant' -> Product.Variant: 546 ms
End: *_ITEM WHERE ItemDiscriminator = 'Product' -> Product.Product: 1123 ms
Start: *_ITEM_PRICELIST -> Product.ProductPriceList
  Start: Extracting Item PriceList author info from v3 and persist in v4 if not found
  End: Extracting Item PriceList author info from v3 and persist in v4 if not found: 17 ms
  Start: Extracting Item PriceList from v3 and persist in v4
  End: Extracting Item PriceList from v3 and persist in v4: 13 ms
End: *_ITEM_PRICELIST -> Product.ProductPriceList: 30 ms
Start: *_ITEM_ASSOCIATION
  Start: -> Product.ProductCategory
  End: -> Product.ProductCategory: 47 ms
  Start: -> Product.CategoryHierarchy
  End: -> Product.CategoryHierarchy: 33 ms
End: *_ITEM_ASSOCIATION: 80 ms
Start: Extracting overall relation IDs without duplicates
End: Extracting overall relation IDs without duplicates: 17 ms
Start: Generating overall unique relation item list
End: Generating overall unique relation item list: 13 ms
Start: *_ITEM_RELATION -> Product.CategoryRelation
```

```
  Start: Persisting new unique Relation group
  End: Persisting new unique Relation group: 30 ms
  Start: Category relation to category
    Start: Initial extract CategoryRelationCategory relation
    End: Initial extract CategoryRelationCategory relation: 33 ms
    Start: Product -> CategoryRelationCategory
    End: Product -> CategoryRelationCategory: 29 ms
  End: Category relation to category: 63 ms
  Start: Category relation to product
    Start: Initial extract CategoryRelationProduct relation
    End: Initial extract CategoryRelationProduct relation: 33 ms
    Start: Product -> Product.CategoryRelationProduct
    End: Product -> Product.CategoryRelationProduct: 46 ms
  End: Category relation to product: 79 ms
  Start: Category relation to variant
    Start: Initial extract CategoryRelationVariant relation
    End: Initial extract CategoryRelationVariant relation: 46 ms
    Start: Product -> Product.CategoryRelationVariant
    End: Product -> Product.CategoryRelationVariant: 47 ms
  End: Category relation to variant: 93 ms
End: *_ITEM_RELATION -> Product.CategoryRelation: 267 ms
Start: *_ITEM_RELATION -> Product.ProductRelation
  Start: Product relation to category
    Start: Initial extract ProductRelationCategory relation
    End: Initial extract ProductRelationCategory relation: 30 ms
    Start: Product -> ProductRelationCategory
    End: Product -> ProductRelationCategory: 30 ms
  End: Product relation to category: 60 ms
  Start: Product relation to product
    Start: Initial extract ProductRelationProduct relation
    End: Initial extract ProductRelationProduct relation: 33 ms
    Start: Product -> Product.ProductRelationProduct
    End: Product -> Product.ProductRelationProduct: 63 ms
  End: Product relation to product: 96 ms
  Start: Product relation to variant
    Start: Initial extract ProductRelationVariant relation
    End: Initial extract ProductRelationVariant relation: 29 ms
    Start: Product -> Product.ProductRelationVariant
    End: Product -> Product.ProductRelationVariant: 93 ms
  End: Product relation to variant: 123 ms
End: *_ITEM_RELATION -> Product.ProductRelation: 280 ms
Start: *_ITEM_RELATION -> Product.VariantRelation
  Start: Variant relation to category
```

Start: Initial extract VariantRelationCategory relation
End: Initial extract VariantRelationCategory relation: 33 ms
Start: Variant -> VariantRelationCategory
End: Variant -> VariantRelationCategory: 13 ms
End: Variant relation to category: 63 ms
Start: Variant relation to product
Start: Initial extract VariantRelationProduct relation
End: Initial extract VariantRelationProduct relation: 30 ms
Start: Variant -> Product.VariantRelationProduct
End: Variant -> Product.VariantRelationProduct: 47 ms
End: Variant relation to product: 77 ms
Start: Variant relation to variant
Start: Initial extract VariantRelationVariant relation
End: Initial extract VariantRelationVariant relation: 50 ms
Start: Variant -> Product.VariantRelationVariant
End: Variant -> Product.VariantRelationVariant: 90 ms
End: Variant relation to variant: 140 ms
End: *_ITEM_RELATION -> Product.VariantRelation: 280 ms
Start: *_ITEM_PRICELIST -> Product.VariantPriceList
End: *_ITEM_PRICELIST -> Product.VariantPriceList: 33 ms
End: Category, Product, Variant Item migration: 2436 ms
End: Scope migration: Global : 2673 ms

ANNEXE XII

Données brutes des mesures

Les chiffriers des données brutes des mesures délivrées à la direction. Certains noms de *scope* ont été anonymisés.

La colonne « *Temps multithread* » s'est effectuée que sur « *GetProductDetail* » et « *GetVariantDetail* » car l'outil *SQLStressQuery* de l'annexe XV « *Outil simulant le « multithreading »* » ne supporte pas les « *User defined table types* ». Conformément au plan de mesure, « *DeleteCategory* » n'a pas été mesuré en « *multi-thread* » puisque ce n'est pas une opération transactionnelle courante et que son temps de mesure est excessif pour la version actuelle de la base de données.

Client 1

	B	D	E	F	G	H	I	J	K
1	Procédure stockée	Version	Scope	Nbr Produits	Nbr Itérations	Utilisation CPU	Temps multi-thread	Temps Profiler	Temps single-thread (ms)
20	GetProductDetail	v3	Global	N/A	1000	22060	0.4844	22764288	22337
21		v4				4140	0.0133	4189849	4262
22	Différence entre v3 et v4					434%	3241%	431%	424%
23									
24	GetProductDetail	v3	N/A	N/A	1000	33834	0.7615	34136838	34283
25		v4				3953	0.0164	3973634	4026
26	Différence entre v3 et v4					757%	4543%	761%	752%
27									
29									
30	GetVariantDetail	v3	Global	N/A	1000	43563	0.672	44179707	44248
31		v4				10766	0.2426	10939151	11002
32	Différence entre v3 et v4					305%	177%	304%	302%
33									
34	GetVariantDetail	v3	N/A	N/A	1000	63250	0.7546	64229223	64103
35		v4				14049	0.2785	14216820	14277
36	Différence entre v3 et v4					350%	173%	350%	349%
37									
39									
40	GetItemPricelistWithInheritance	v3	Global	1	1000	7188	N/A	7180966	7187
41		v4				1075		1137116	2086
42	Différence entre v3 et v4					561%		525%	245%
43									
44	GetItemPricelistWithInheritance	v3	N/A	1	1000	19234	N/A	19465823	19527
45		v4				1301		1244513	2171
46	Différence entre v3 et v4					1378%		1463%	799%
47									
48									
49	GetItemPricelistWithInheritance	v3	Global	26	1000	9655	N/A	9591071	9634
50		v4				1552		1719126	2652
51	Différence entre v3 et v4					509%		456%	263%
52									
53	GetItemPricelistWithInheritance	v3	N/A	26	1000	32516	N/A	32731528	32808
54		v4				2701		2577971	3694
55	Différence entre v3 et v4					1104%		1170%	788%
56									
57									
58	GetItemPricelistWithInheritance	v3	Global	Max	1000	31297	N/A	13449458	13520
59		v4				5623		5669272	6711
60	Différence entre v3 et v4					136%		137%	101%
61									
62	GetItemPricelistWithInheritance	v3	N/A	Max	1000	38016	N/A	10461171	18553
63		v4				10395		10521325	11654
64	Différence entre v3 et v4					266%		266%	231%

Figure XII-1 - Mesures transactionnelles brutes et pourcentages pour le client #1

	B	D	E	F	G	H	I	J	K
1	Procédure stockée	Version	Scope	Nbr Produits	Nbr Itérations	Utilisation CPU	Temps multi-thread	Temps Profiler	Temps single-thread (ms)
9									
10	DeleteCategory	v3	Global	N/A	10	10999	N/A	63418643	63423
11		v4				4062		10964728	10965
12	Différence entre v3 et v4					885%		661%	661%
13									
14	DeleteCategory	v3	N/A	N/A	10	162438	N/A	21997747205	21998304
15		v4				4062		11382923	11386
16	Différence entre v3 et v4					39868%		193153%	193105%
17									

Figure XII-2 - Mesures non transactionnelles brutes et pourcentages pour le client #1

Client 2

	B	D	E	F	G	H	I	J	K
1	Procédure stockée	Version	Scope	Nbr Produits	Nbr Itérations	Utilisation CPU	Temps multi-thread	Temps Profiler	Temps single-thread (ms)
20	GetProductDetail	v3	Global	N/A	1000	22503	0.1155	22844957	22913
21		v4				3828	0.0094	3905015	3971
22	Différence entre v3 et v4					488%	1341%	485%	477%
23									
24	GetProductDetail	v3	N/A	N/A	1000	31909	0.2283	32205806	32275
25		v4				3613	0.0098	3789402	3838
26	Différence entre v3 et v4					784%	2332%	750%	741%
27									
29									
30	GetVariantDetail	v3	Global	N/A	1000	32376	0.1541	32706973	32789
31		v4				8155	0.0109	8251368	8321
32	Différence entre v3 et v4					293%	1350%	296%	294%
33									
34	GetVariantDetail	v3	N/A	N/A	1000	100422	0.2681	101078680	101162
35		v4				15388	0.013	15658713	15731
36	Différence entre v3 et v4					553%	1964%	546%	543%
37									
39									
40	GetItemPricelistWithInheritance	v3	Global	1	1000	7514	N/A	7743099	7828
41		v4				1078		1002834	1946
42	Différence entre v3 et v4					597%		672%	302%
43									
44	GetItemPricelistWithInheritance	v3	N/A	1	1000	28827	N/A	27027127	27086
45		v4				1050		1071019	2043
46	Différence entre v3 et v4					2455%		2423%	1226%
47									
48									
49	GetItemPricelistWithInheritance	v3	Global	2	1000	7844	N/A	7976339	8038
50		v4				1032		1006683	1943
51	Différence entre v3 et v4					660%		692%	314%
52									
53	GetItemPricelistWithInheritance	v3	N/A	2	1000	20456	N/A	20682857	20799
54		v4				1061		1122806	2065
55	Différence entre v3 et v4					1828%		1742%	905%
56									
57									
58	GetItemPricelistWithInheritance	v3	Global	Max	1000	502313	N/A	10263447	10329
59		v4				181563		1018702	1953
60	Différence entre v3 et v4					177%		908%	428%
61									
62	GetItemPricelistWithInheritance	v3	N/A	Max	1000	21172	N/A	21327169	21359
63		v4				1253		1269004	2249
64	Différence entre v3 et v4					1590%		1581%	851%

Figure XII-3 - Mesures transactionnelles brutes et pourcentages pour le client #2

	B	D	E	F	G	H	I	J	K
1	Procédure stockée	Version	Scope	Nbr Produits	Nbr Itérations	Utilisation CPU	Temps multi-thread	Temps Profiler	Temps single-thread (ms)
9									
10	DeleteCategory	v3	Global	N/A	10	1375	N/A	1746660	1746
11		v4				515		543841	547
12	Différence entre v3 et v4					167%		221%	219%
13									
14	DeleteCategory	v3	N/A	N/A	10	3297	N/A	3342199	3345
15		v4				484		488390	484
16	Différence entre v3 et v4					581%		584%	591%
17									

Figure XII-4 - Mesures non transactionnelles brutes et pourcentages pour le client #2

Client 3

	B	D	E	F	G	H	I	J	K
1	Procédure stockée	Version	Scope	Nbr Produits	Nbr Itérations	Utilisation CPU	Temps multi-thread	Temps Profiler	Temps single-thread (ms)
20	GetProductDetail	v3	Global	N/A	1000	19467	0.1771	19746483	19817
21		v4				3687	0.0098	3774821	3821
22	Différence entre v3 et v4					428%	1707%	423%	419%
23									
24	GetProductDetail	v3	N/A	N/A	1000	34077	0.2558	34518450	34592
25		v4				4062	0.011	4103963	4152
26	Différence entre v3 et v4					739%	2225%	741%	733%
27									
29									
30	GetVariantDetail	v3	Global	N/A	1000	15641	0.2281	15950026	16016
31		v4				3230	0.01	3412934	3454
32	Différence entre v3 et v4					384%	2181%	367%	364%
33									
34	GetVariantDetail	v3	N/A	N/A	1000	13593	0.1684	13896256	13975
35		v4				3329	0.0092	3411824	3461
36	Différence entre v3 et v4					308%	1730%	307%	304%
37									
39									
40	GetItemPricelistWithInheritance	v3	Global	1	1000	7434	N/A	7840288	7707
41		v4				1004		935424	1909
42	Différence entre v3 et v4					640%		668%	304%
43									
44	GetItemPricelistWithInheritance	v3	N/A	1	1000	22748	N/A	22944427	23012
45		v4				1078		1111733	2056
46	Différence entre v3 et v4					2010%		1964%	1019%
47									
48									
49	GetItemPricelistWithInheritance	v3	Global	22	1000	8001	N/A	8164612	8226
50		v4				1079		1043715	1996
51	Différence entre v3 et v4					642%		682%	312%
52									
53	GetItemPricelistWithInheritance	v3	N/A	22	1000	22908	N/A	23214406	23277
54		v4				1315		1412754	2352
55	Différence entre v3 et v4					1642%		1543%	890%
56									
57									
58	GetItemPricelistWithInheritance	v3	Global	Max	1000	9779	N/A	9941855	9130
59		v4				1436		1384640	2399
60	Différence entre v3 et v4					511%		553%	283%
61									
62	GetItemPricelistWithInheritance	v3	N/A	Max	1000	33814	N/A	34127341	34285
63		v4				24970		25099361	26300
64	Différence entre v3 et v4					35%		36%	30%

Figure XII-5 - Mesures transactionnelles brutes et pourcentages pour le client #3

	B	D	E	F	G	H	I	J	K
1	Procédure stockée	Version	Scope	Nbr Produits	Nbr Itérations	Utilisation CPU	Temps multi-thread	Temps Profiler	Temps single-thread (ms)
9									
10	DeleteCategory	v3	Global	N/A	10	493500	N/A	502914564	503823
11		v4				71249		71709156	71767
12	Différence entre v3 et v4					593%		601%	601%
13									
14	DeleteCategory	v3	N/A	N/A	10	385125	N/A	307705781	307716
15		v4				969		1156502	1158
16	Différence entre v3 et v4					31389%		26507%	26473%
17									

Figure XII-6 - Mesures non transactionnelles brutes et pourcentages pour le client #3

Comparaison de la taille compressée

	A	B	C	D	E
1	Client	v3	v4	Pourcentage de compression	Facteur de compression
2					
3		Mégaoctets			
4	Client 1	22930	11350	102%	2,03
5	Client 2	721	171	322%	4,22
6	Client 3	7788	1535	407%	5,07

Figure XII-7 - Mesures brutes et pourcentage de la taille compressée des bases de données des produits

ANNEXE XIII

Données agrégées

Procédures stockées transactionnelles

Les résultats transactionnels font référence aux trois procédures stockées « *GetProductDetail* », « *GetVariantDetail* » et « *ItemPriceListWithInheritance* ».

	Améliorations transactionnelles		
	CPU Réduction de la charge	Temps d'exécution Amélioration single-thread	Temps d'exécution Amélioration multi-thread
Client 1	408%	408%	380%
Client 2	255%	645%	1757%
Client 3	313%	314%	1974%

Figure XIII-1 - Mesures agrégées des résultats transactionnels

Résultats non transactionnels

Ceci fait référence à la procédure stockée ayant initié le projet, le « *DeleteCategory* ». La portion « *Multi-thread* » n'est pas incluse puisque cette opération est non transactionnelle et rarissime. Lorsqu'elle est utilisée, c'est par un utilisateur qui l'exécute une à la fois et ne supporte pas la concurrence. Autrement, elle générerait pratiquement que des verrous mortels. Si cette opération venait qu'à devoir supporter massivement la concurrence (ce qui n'est aucunement dans les plans d'affaires), alors son fonctionnement devra être revu sous un autre angle.

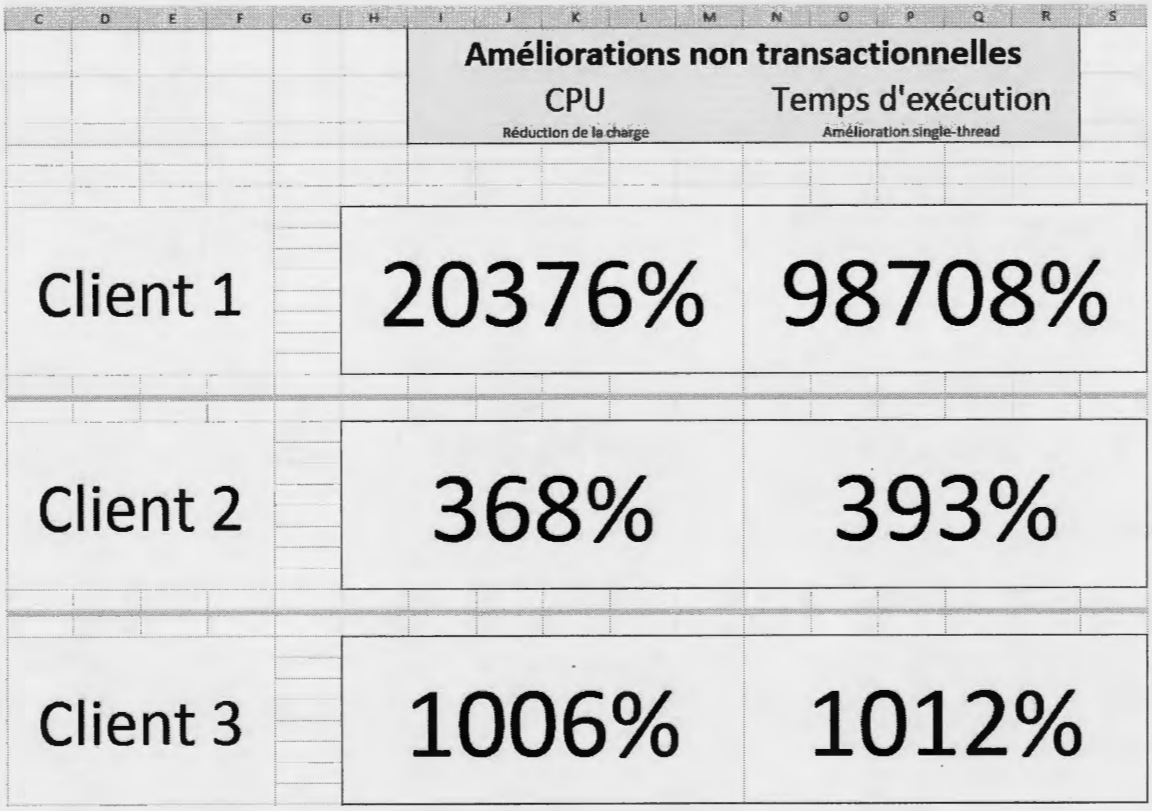


Figure XIII-2 - Mesures agrégées des résultats non transactionnels

ANNEXE XIV

Confirmation de l'approbation du projet par le directeur

Salut,

Tout est selon ce qui a été discuté. Je n'ai donc aucun commentaire additionnel.

[redacted]

From: Luc-Andre Jolivet <lucandre.jolivet >
Sent: December 21, 2018 11:32 AM
To:
Subject: Petit service svp!

Salut [redacted]

Je peux te demander un service svp?
Je sais que c'est un peu dernière minute avant les fêtes, je me suis empressé de modifier mon document suite à notre discussion d'hier.
Ça m'aiderait énormément pour pouvoir y travailler pendant les fêtes si tu pouvais le lire avant de quitter aujourd'hui svp.
Approuver ou sinon corriger ce qui est attendu par le côté « business » des fonctionnalités de la BD des produits en v4.

Au besoin, tu peux rajouter des commentaires dans des bulles dans Word, ce n'est pas le document final au propre mais une copie.

La formulation n'est pas définitive (et la correction des fautes d'orthographe) car je dois aussi valider avec ma directrice (et [redacted] pour d'autres sections), mais pour ce qui est attendu des fonctionnalités d'affaires, ça c'est 100% toi!

Merci

Luc-Andre Jolivet
Architecte de données

Figure XIV-1 - Courriel complet de l'approbation provenant du directeur du commerce numérique

ANNEXE XV

Outil simulant le « multithreading »

SqlQueryStress

https://github.com/ErikEJ/SqlQueryStress

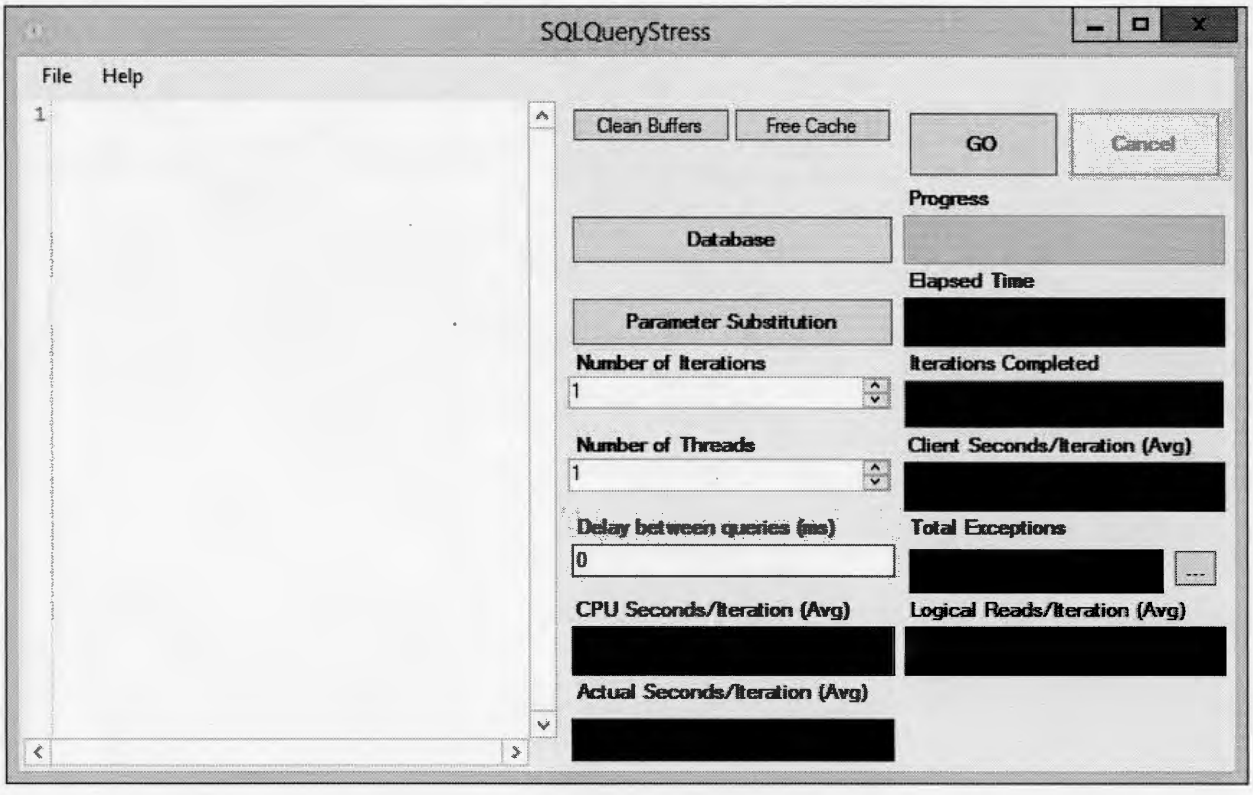


Figure XV-1 - SQLQueryStress outil simulant le "multithreading"

ANNEXE XVI

Commentaires de l'auditeur

Les quatre commentaires laissés par l'auditeur mais n'entravant en rien la poursuite des mesures. La confirmation de poursuite est à l'annexe XVII « *Confirmation de l'approbation de l'auditeur* ».

1. Contrairement à ce que laisse sous-entendre l'outil simulant le « multithreading », il n'utilisait pas tous les cœurs du CPU côté applicatif. Par contre, nous avions constaté que SQL était bien utilisé en parallèle, ce qui était le but recherché. Donc, tant qu'un seul cœur n'était pas surchargé, il était possible d'invoquer de multiples requêtes à SQL, en parallèle. L'auditeur a convenu que ce n'était pas un point bloquant quant à la fiabilité des résultats pour ce qui était des requêtes parallèles. Plus d'informations sur cet outil sont à l'annexe XV « *Outil simulant le « multithreading »* ».

2. L'état initial de la BD dans son intégration future avec la plateforme devra être adapté pour contrer un cas particulier si le projet se poursuit. Toutefois, il a convenu que ce n'était pas non plus un point bloquant pour effectuer les mesures.

3. Une table de la nouvelle base de données avait quelques données additionnelles non nécessaires, issues de la migration des données. L'auditeur a aussi convenu que ceci n'allait pas avoir d'impact sur les mesures et a donc acquiescé pour la poursuite de celles-ci.

4. Dans le fichier des mesures, il avait recommandé d'utiliser des couleurs pour bien faire ressortir les mesures favorables et défavorables. J'ai implémenté cette recommandation.

ANNEXE XVII

Confirmation de l'approbation de l'auditeur

From:
Sent: September 5, 2019 1:25 PM
To: Luc-Andre Jolivet <lucandre.jolivet
Subject: RE: Audit de mercredi dernier

Désolé du délai.

Développeur logiciel principal

From: Luc-Andre Jolivet <lucandre.jolivet
Sent: Friday, August 30, 2019 10:00 AM
To:
Subject: Audit de mercredi dernier

Bon matin

Voici les quelques questions suite à notre rencontre de mercredi le 28 août 2019 concernant la base des données de produits v4 dont je t'avais parlé :

1. Ai-je ton aval afin de poursuivre avec les mesures?
   Oui
2. Est-ce que les procédures stockées utilisées sont représentatives des points faibles à améliorer ou de l'utilisation courante de la base de données des produits actuelle?
   Oui
3. Est-ce que les outils, procédures, façons de faire et cas de tests sont suffisants afin d'obtenir et d'assurer des résultats valides et fiables?
   Oui
4. Est-ce que toutes les exigences d'affaires (autre que la performance CPU) ont été passées en revue et sont bien libellées pour la traçabilité?
   Oui

Facultatif: un commentaire que tu aimerais partager?

J'ai aimé structure de code utilisé par Luc-André pour le script de migration.

Merci de ton implication.

Luc-Andre Jolivet
Architecte de données

Figure XVII-1 - Courriel complet de l'approbation provenant de l'auditeur

ANNEXE XVIII

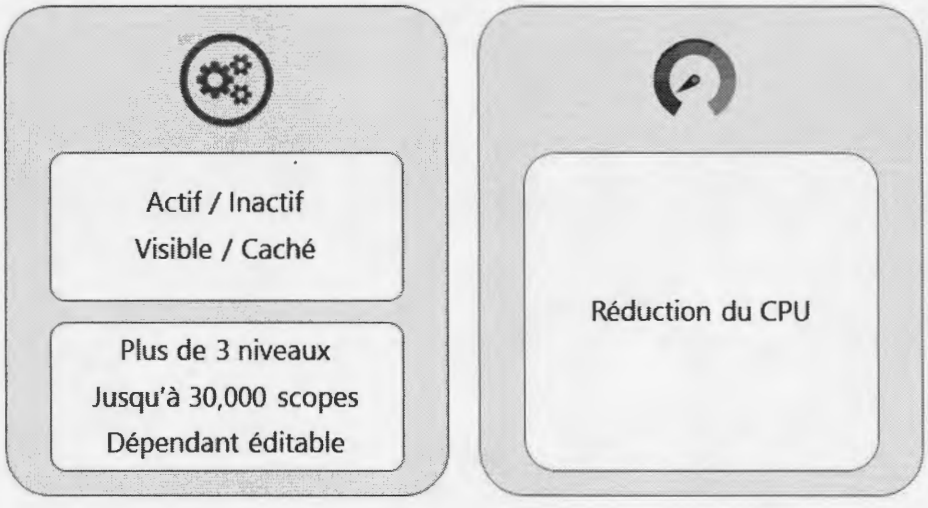
Présentation PowerPoint des résultats

Les noms des clients et individus ont volontairement été retirés aux fins de confidentialité. De plus, les éléments de structure de code ont été légèrement brouillés afin de percevoir le jeu de couleur servant à la traçabilité des fonctionnalités demandées par le directeur du commerce numérique.

Aperçu de la présentation

- Récapitulatif des objectifs initiaux
- Revue des fonctionnalités implémentées
- Récapitulatif des mesures de performance
- Résultats des mesures
- Récapitulatif des incidences positives additionnelles
- Discussion pour la suite

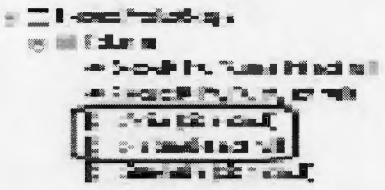
Exigences initiales



Traçabilité des exigences

Champs « IsActive » & « IsHidden » disponibles
(traçabilité rouge)

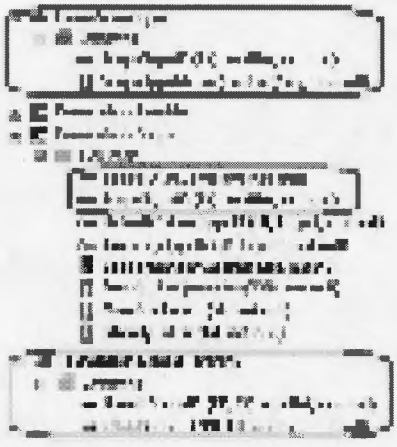
Par produit



Par variant

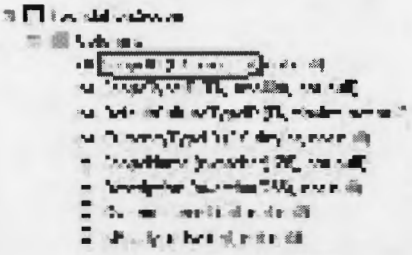
Traçabilité des exigences

Une hiérarchie de plus de trois niveaux
(traçabilité verte)



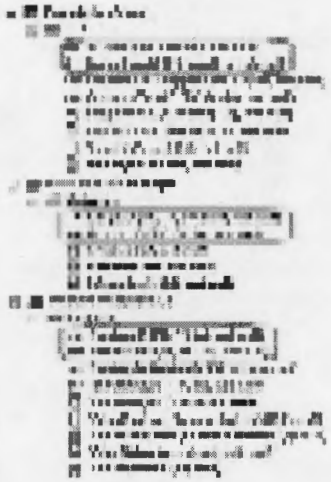
Traçabilité des exigences

30 000 scopes (traçabilité mauve)



Traçabilité des exigences

Modification des attributs de produit
peu importe le scope
(traçabilité orange)



Mesures entre v3 et v4

Clients mesurés

Client 1

Client 2

Client 3

Procédures stockées utilisées

- GetPriceListWithInheritance
- GetProductDetail
- GetVariantDetail
- DeleteCategory

Taille

- Sauvegarde compressée

Vérification et tests

- Tester le décompte des entités
- Tester chacun des produits et variants de GetProduct & Variant detail
- Tester l'effacement de catégorie

Validation & auditeur:

- Développeur sénior

Améliorations transactionnelles

	0 Client	1 Réduction CPU	2 Temps d'exécution "Single-Thread"	3 Temps d'exécution "Multi-Thread"
Get Product / Variant Detail + ItemPriceListWithInheritance	Client 1	408 %	408 %	380 %
	Client 2	255 %	645 %	1757 %
	Client 3	313 %	314 %	1974 %

Améliorations non transactionnelles

Delete Category

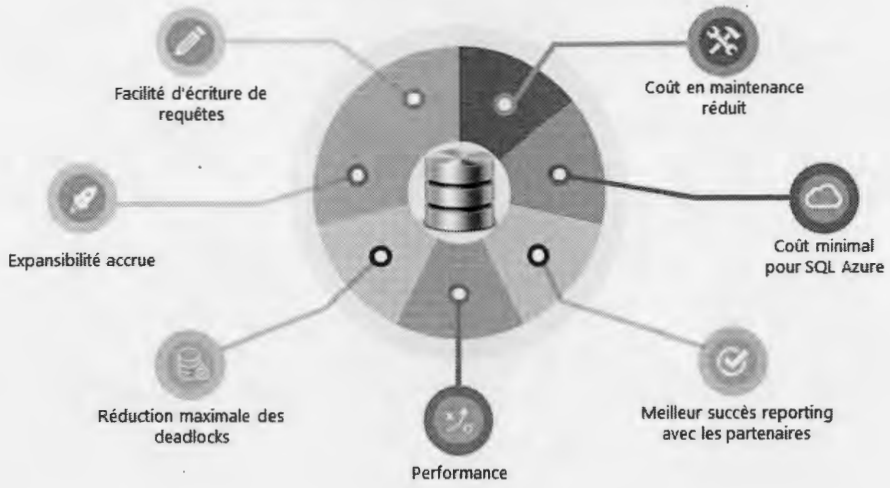
0 Client	1 Réduction CPU	2 Temps d'exécution "Single-Thread"
Client 1	20376 %	98708 %
Client 2	368 %	393 %
Client 3	1006 %	1012 %

Améliorations de la taille des sauvegardes

Réduction de la taille

0 Client	1 Taille compressée
Client 1	102 %
Client 2	322 %
Client 3	407 %

Incidences positives additionnelles



12

Poursuite du projet ?

13

Références

- http://www.simplitfy.com/images/icon-performance.png
- https://jsreport.net/img/api.png
- http://www.iconhot.com/icon/png/database/512/database-1-1.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/c/c4/Ambox_blue_question.svg/900px-Ambox_blue_question.svg.png

ANNEXE XIX

Résultat de la conformité avec ISO 29110

Les captures d'écran suivantes proviennent du même chiffrier que celui fournit pour le cours MGL 7560 par M. Normand Séguin pour le profil basique.

Nom du processus	% Exécutée	% Point vers les 67 tâches	Nom de l'activité	Oui	Non	Partiel	# Total de tâches
Gestion de Projet							
	97%	26%	PM1-Planification du projet	14	0	1	15
	92%	10%	PM2-Exécution du Plan de projet	5	0	1	6
	83%	4%	PM3-Évaluation et contrôle du projet	2	0	1	3
	100%	4%	PM4-Clôture du projet	2	0	0	2
Moyenne du processus	93%			23	0	3	26
Implémentation logicielle							
	100%	4%	SI1-Initiation de l'implantation du logiciel	2	0	0	2
	100%	7%	SI2-Analyse des exigences du logiciel	4	0	0	4
	100%	14%	SI3-Architecture et conception détaillée du logiciel	8	0	0	8
	100%	13%	SI4-Réalisation du logiciel	7	0	0	7
	100%	11%	SI5-Test et intégration du logiciel	6	0	0	6
	100%	5%	SI6-Livraison du produit	3	0	0	3
Moyenne du processus	100%			30	0	0	30
Tâches non exécutées		3%					
Moyenne total du processus	97%						

Figure XIX-1 - ISO 29110 - Exécution des tâches

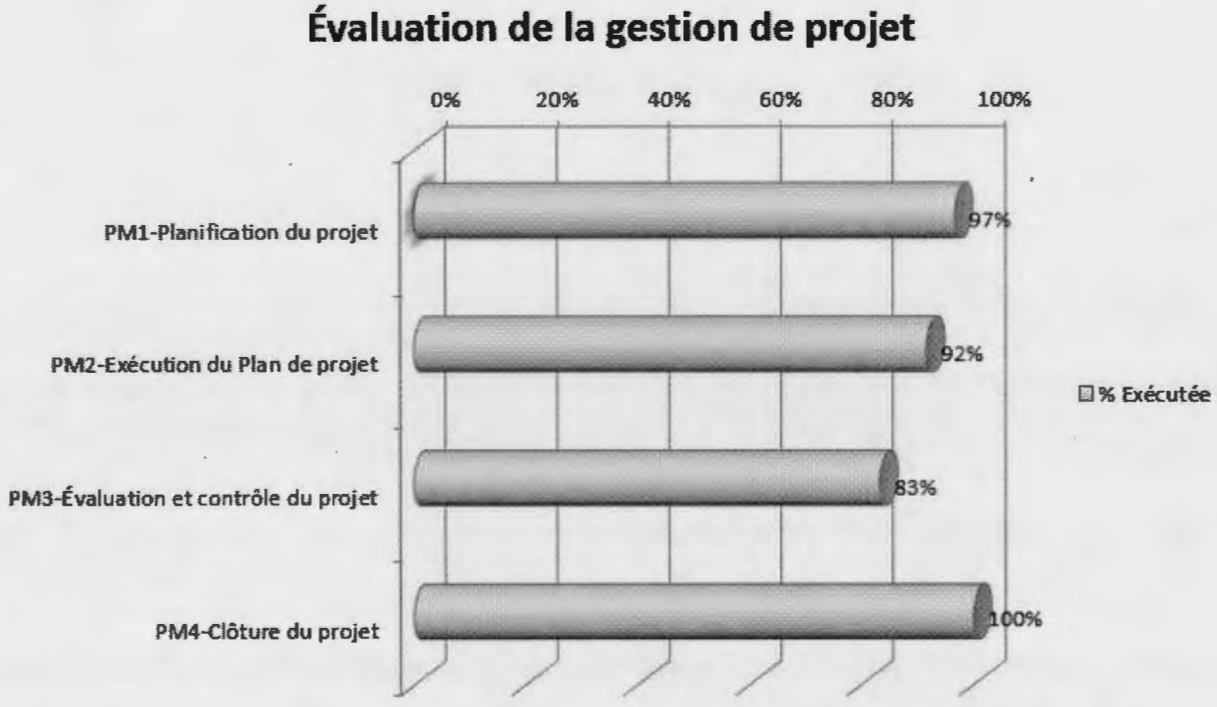


Figure XIX-2 - ISO 29110 - Évaluation de la gestion de projet

Il est à noter que pour GP1, GP2 et GP3, la raison de leur dépréciation est due au fait que le projet ne tient pas compte des dates dans le calendrier. Autrement, toutes les tâches auraient été exécutées selon ce qu'ISO 29110 mandate. Le détail des explications se retrouve dans la sous-section « *3.1.3 – Planifier le projet* ».

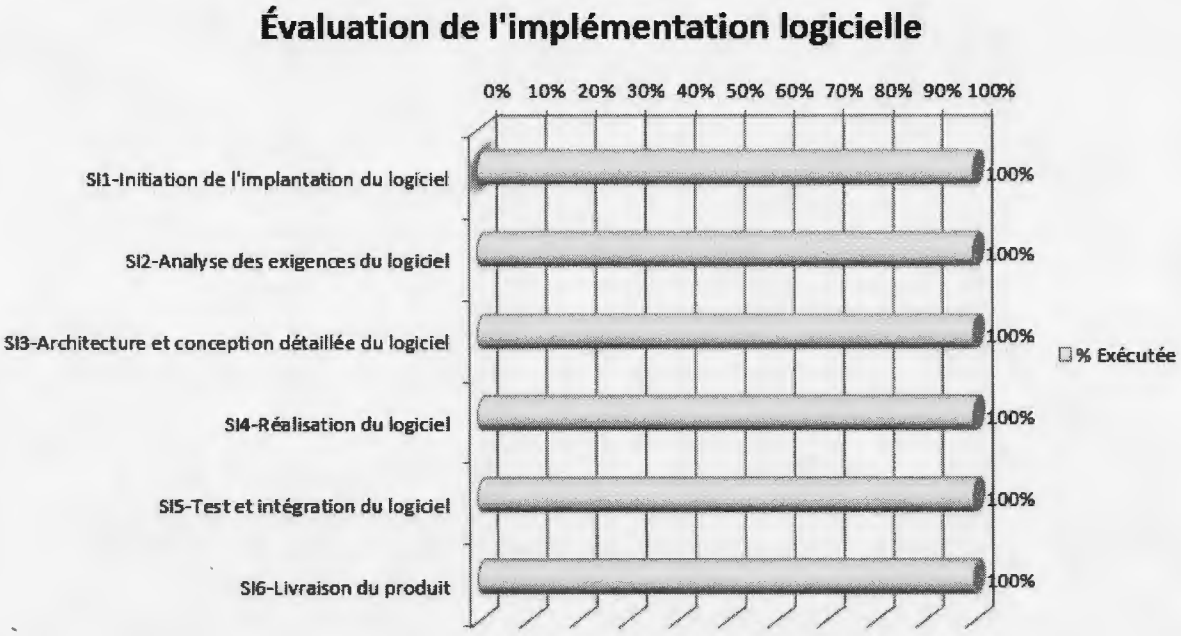


Figure XIX-3 - ISO 29110 - Évaluation de l'implémentation logicielle

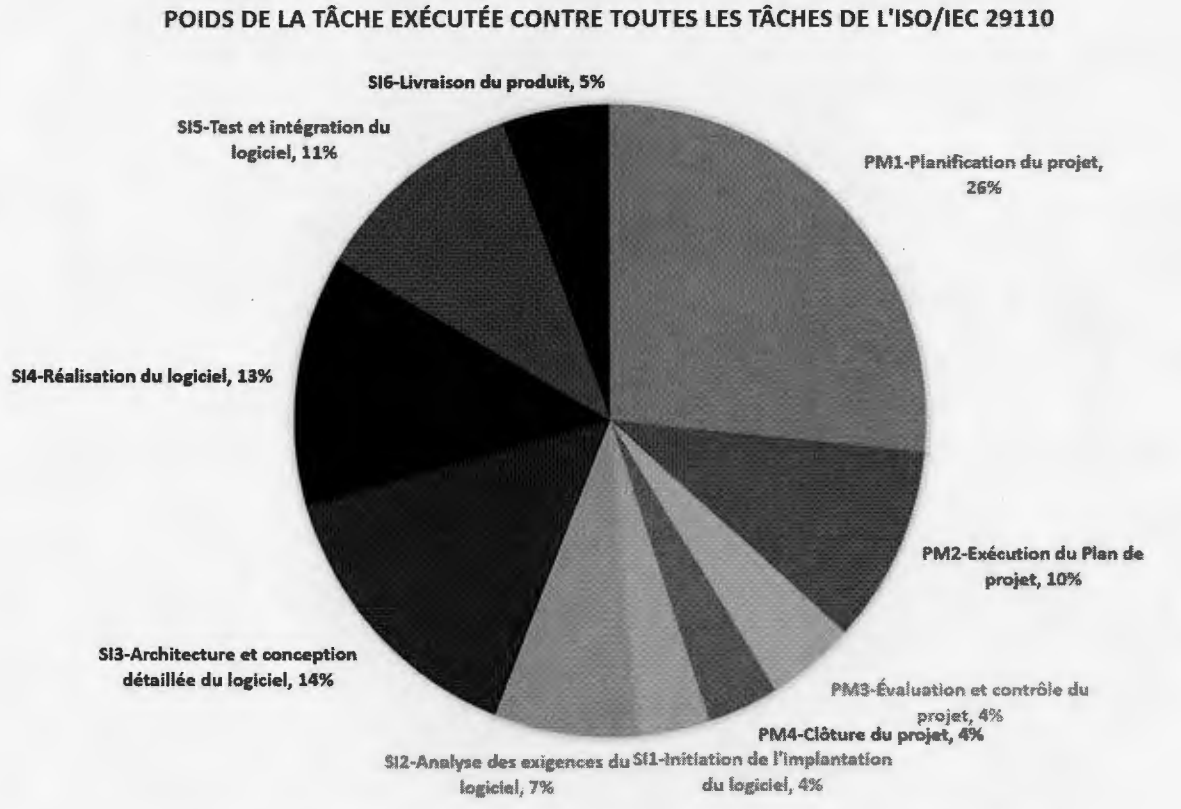


Figure XIX-4 - ISO 29110 - Répartition du poids des tâches ISO 29110

ANNEXE XX

Évolution de la nouvelle base de données des produits

La ligne d'affaires s'oriente avec l'approche Agile, plus précisément la méthode Scrum. L'intérêt d'aborder l'agilité est de sonder si la littérature offre une quelconque information de précaution après la livraison de la base de données afin qu'elle ne se retrouve pas dans le même état qu'au départ du projet.

Pour débuter, le manifeste indique clairement que pour accentuer l'agilité, il est préconisé d'effectuer les changements lorsque nécessaire, et ce, de la meilleure manière qu'il soit :

« *Continuous attention to technical excellence and good design enhances agility* » (Beck et al., 2001), le Manifesto Agile et reprise par IEEE 730 (IEEE Computer Society, 2012) p.95

De plus, il est supposé que les meilleures architectures et conceptions émergent d'équipes qui peuvent s'autoorganiser.

« *The best architectures, requirements, and designs emerge from self-organizing teams* » (Beck et al., 2001) , le Manifesto Agile et reprise par IEEE 730 (IEEE Computer Society, 2012) p.95

Pour une compagnie qui souhaite s'aligner avec l'agilité, et donc les deux principes ci-devant, elle doit composer avec ce qui suit. La littérature propre au développement des bases de données stipule qu'il n'est pas souhaitable de laisser aux développeurs seuls le soin de décider des évolutions de modélisations de bases de données sans en avoir l'expérience de ce qui peut nuire à la performance (Schraml, 2014). De plus, les développeurs, en matière de SQL, produisent les mêmes erreurs encore et encore (Karwin, 2010).

« *software developers make the same mistakes over and over again* » (Karwin, 2010) p.14

« *Despite SQL's widespread use in enterprise software, there are still pitfalls in using it. Many application developers don't understand SQL well and, as a result, have problems defining effective queries and commands.* » (Fowler, 2002) p.33

Quant à la méthode Scrum elle-même, l'une de ses caractéristiques est d'encourager les équipes multidisciplinaires (Boisvert, Trudel, 2011). La littérature va dans le même sens que le manifeste Agile et recommande que du personnel ayant les qualifications nécessaires, puisse se joindre aux équipes de développements afin de ne pas compromettre l'agilité (Shmeltzer, 2018). Dans le cas des compétences transversales ou partagées, la méthodologie Kanban est un outil pouvant œuvrer à faciliter l'adoption d'équipe multidisciplinaire lorsque ses membres sont peu disponibles (Boisvert, Trudel, 2011), par exemple, un DBA. Il n'est pas toujours possible en entreprise que ces compétences soient dédiées à un unique projet. Lorsqu'il y a lancement de nouveaux projets ou fonctionnalités, une recommandation est d'incorporer le plus tôt possible ces ressources partagées pour qu'elles puissent déterminer leur contribution, et à quel stade (en contrepartie d'attendre au moment où l'équipe a besoin d'eux) afin de continuer à s'aligner avec les principes Agiles (Boisvert, Trudel, 2011).

Le monde des bases de données possède ses propres caractéristiques lorsque vient le temps du développement (Shmeltzer, 2018). Dans un contexte d'intégration en continu et de versionnage de code, les bases de données ne sont pas aussi matures que pour du développement applicatif du fait qu'elles doivent maintenir des états cohérents de l'information en plus de devoir effectuer les transitions requises, ce qui rend le remaniement de code plus complexe en bases de données que pour du code applicatif (Shmeltzer, 2018).

Là où des observations divergentes émergent, c'est quant aux outils disponibles aux DBA versus ceux pour les développeurs. Les outils disponibles pour le remaniement de code sont peu adaptés, rendent les opérations ardues et il est difficile d'automatiser des tâches (Vial, 2015). Une adaptation sera nécessaire afin d'ajuster les gestionnaires de sources qui sont un problème beaucoup plus complexe que pour les développeurs (Shmeltzer, 2018). L'automatisation de processus pour déterminer quel script est appliqué à quel endroit, le maintien de l'état des données et l'impossibilité de détruire et recréer des objets en SQL sont des scénarios à envisager pour bien arrimer Agile avec la gestion du code des bases de

données (Shmeltzer, 2018). Une des raisons qui gêne ou entrave le remaniement de code provient justement d'outils inadéquats pour effectuer le travail (Tempero et al., 2017).

« *Le refactoring du modèle de données et de persistance d'un système est plus complexe que le refactoring de sa logique métier.* » Guillemette, François-Xavier, Enseignant Maîtrise Génie Logiciel, MGL7361, Automne 2015.

« *Les mesures de productivité de tâches équivalentes indiquent que tout remaniement de code impliquant la base de données prend plus de temps que pour du code uniquement applicatif.* » P.Sciascia, Directeur de recherche et développement, Posera HDX, 2012.

Finalement, pour plus de détail et d'exemples, un livre complet est consacré pour arrimer les techniques de bases de données avec l'agilité (Ambler, 2003).

APPENDICES

A. Norme ISO 29110

Les informations listées ci-après sont extraites de la norme ISO 29110 telle quelle (Organisation internationale de Normalisation, 2012).

Activités du processus de gestion de projet

PM.1 - Planification du projet

PM.1.1 - Réviser et accepter le plan de projet.

PM.1.2 - Définir, avec le client, les directives de livraison pour chacun des livrables déterminés dans l'énoncé des travaux.

PM.1.3 - Déterminer les tâches précises à effectuer afin de fournir les Livrables ainsi que les Composants logiciels indiqués dans l'énoncé des travaux. Intégrer les Tâches dans le processus de SI ainsi que la vérification, la validation et les révisions des Tâches qui incombent au client et à l'équipe de travail pour assurer la qualité des produits. Déterminer les Tâches visant l'application des Directives de livraison. Documenter les Tâches.

PM.1.4 - Établir la durée estimée pour la réalisation de chacune des tâches.

PM.1.5 - Déterminer et documenter les ressources humaines et matérielles nécessaires ainsi que l'équipement et les outils requis et les normes à appliquer ainsi que la formation dont ont besoin les membres de l'équipe de travail pour réaliser le projet. Intégrer au calendrier les dates auxquelles ces Ressources et cette formation sont requises.

PM.1.6 - Établir la Composition de l'équipe de travail et en assigner les rôles et les responsabilités en fonction des Ressources.

PM.1.7 - Fixer les dates prévues de commencement et de fin pour chacune des Tâches afin de créer le Calendrier des tâches du projet en tenant compte des Ressources qui y sont affectées, de la séquence des Tâches et de leur dépendance.

PM.1.8 - Calculer et documenter les Coûts et les efforts estimés.

PM.1.9 - Déterminer et documenter les risques qui peuvent avoir une incidence sur le projet.

PM.1.10 - Documenter la Stratégie de contrôle des versions dans le Plan de projet.

PM.1.11 - Produire le Plan de projet en y intégrant les éléments précédemment définis et documentés.

PM.1.12 - Inclure la Description du produit, la Portée, les Objectifs et les Livrables au Plan de projet.

PM.1.13 - Vérifier le Plan de projet et en obtenir l'approbation. S'assurer que tous les éléments du Plan de projet sont viables et cohérents. Les résultats sont documentés dans les Résultats de la vérification et des corrections sont apportées jusqu'à ce que le document reçoive l'approbation du gestionnaire de projet.

PM.1.14 - Réviser et accepter le Plan de projet. Le client examine et accepte le Plan de projet en s'assurant que les éléments qui y sont énoncés correspondent à ceux de l'Énoncé des travaux.

PM.1.15 - Établir un Dépôt d'information du projet en utilisant la Stratégie de contrôle des versions.

PM.2 - Exécution du plan de projet

PM.2.1 - Surveiller l'exécution du Plan de projet et consigner les données réelles dans le Rapport d'avancement.

PM.2.2 - Analyser et évaluer la Demande de changement dans la perspective de l'incidence sur le coût, le calendrier et les exigences techniques.

La Demande de changement peut provenir de l'externe (du client) ou de l'interne (de l'équipe de travail). Si le changement, une fois accepté, n'a pas d'incidence sur les ententes avec le client, mettre le Plan de projet à jour.

La Demande de changement qui a une incidence sur ces ententes doit faire l'objet de négociations entre les deux parties (voir PM.2.4).

PM.2.3 - Tenir des réunions de revue avec l'équipe de travail, cerner les problèmes, examiner l'état des risques, consigner les ententes et en assurer le suivi jusqu'à la clôture du projet.

PM.2.4 - Tenir des réunions de revue avec le client, consigner les ententes et en assurer le suivi jusqu'à la clôture du projet.

La demande de changement, présentée par le client ou par l'équipe de travail, ayant une incidence sur le client, doit faire l'objet de négociations et être acceptée par les deux parties.

S'il y a lieu, mettre le Plan de projet à jour en fonction de la nouvelle entente avec le client.

PM.2.5 - Produire des copies de sécurité en fonction de la Stratégie de contrôle des versions.

PM.2.6 - S'il y a lieu, effectuer la récupération du Dépôt d'information du projet en utilisant les Copies de sécurité.

PM.3 - Évaluation et contrôle du projet

PM.3.1 - Évaluer l'avancement du projet par rapport au Plan de projet en comparant :

- les Tâches réelles par rapport aux Tâches planifiées
- les résultats réels par rapport aux Objectifs du projet
- les ressources réellement affectées au projet par rapport aux Ressources planifiées
- les coûts réels par rapport aux estimations budgétaires
- les écarts par rapport à l'échéancier établi
- les risques réels par rapport aux risques précédemment déterminés

PM.3.2 - Établir des mesures visant à corriger les écarts ou à résoudre les problèmes et à éviter les risques déterminés susceptibles de compromettre la réalisation du plan, s'il y a lieu, les consigner dans le Registre des corrections, puis en assurer le suivi jusqu'à la clôture du projet.

PM.3.3 - Repérer les changements apportés aux exigences et au Plan de projet afin de corriger les écarts importants ou résoudre les problèmes et éviter les risques susceptibles de compromettre la réalisation du plan, consigner le tout dans la Demande de changement, puis en assurer le suivi jusqu'à la clôture du projet.

PM.4 - Clôture du projet

PM.4.1 - Officialiser la clôture du projet selon les Directives de livraison établies dans le Plan de projet, offrir le soutien nécessaire à l'acceptation du produit et faire signer l'enregistrement de réception.

PM.4.2 - Mettre à jour le Dépôt d'information du projet.

Processus de mise en œuvre du logiciel

SI.1 - Initiation de la mise en œuvre du logiciel

SI.1.1 - Réviser le Plan de projet en vigueur avec les membres de l'équipe de travail afin d'arriver à une compréhension commune et d'obtenir leur engagement pour la réalisation du projet.

SI.1.2 - Mettre en place ou à jour l'environnement de mise en œuvre.

SI.2 - Analyse des exigences du logiciel

SI.2.1 - Assigner les Tâches aux membres de l'équipe de travail conformément à leurs rôles selon le plan de projet en vigueur.

SI.2.2 - Documenter ou mettre à jour la Spécification des exigences.

Repérer et consulter les sources d'information (client, utilisateurs, systèmes précédents, documents, etc.) afin d'obtenir de nouvelles exigences.

Analyser les exigences établies pour déterminer la Portée et la faisabilité.
Générer ou mettre à jour la Spécification des exigences.

SI.2.3 - Vérifier la Spécification des exigences et en obtenir l'approbation.
Vérifier l'exactitude et la testabilité de la Spécification des exigences ainsi que sa cohérence avec la Description du produit. De plus, vérifier les exigences afin de veiller à ce qu'elles soient complètes, sans ambiguïté et qu'elles ne se contredisent pas. Les résultats sont documentés dans les Résultats de la vérification et des corrections sont apportées jusqu'à ce

que le document soit approuvé par un analyste. Si des changements importants sont requis, initier une Demande de changement.

SI.2.4 - Valider la Spécification des exigences et en obtenir l'approbation.
Valider la Spécification des exigences afin de s'assurer qu'elle réponde aux besoins et aux attentes établis, dont la convivialité de l'interface utilisateur. Les résultats sont documentés dans les Résultats de la validation et des corrections sont apportées jusqu'à ce que le document reçoive l'approbation du client.

SI.2.5 - Documenter la version préliminaire de la *Documentation de l'utilisateur du logiciel ou mettre à jour le document existant, s'il y a lieu.
*(Facultatif)

SI.2.6 - Vérifier et faire approuver le Documentation de l'utilisateur du logiciel*, s'il y a lieu.
Vérifier la cohérence de la *Documentation de l'utilisateur du logiciel avec la Spécification des exigences. Les résultats sont documentés dans les Résultats de la vérification et les corrections sont apportées jusqu'à ce que le document soit approuvé par un analyste. Si des changements importants sont requis, entreprendre une Demande de changement.
*(Facultatif)

SI.2.7 - Intégrer la Spécification des exigences et la *Documentation de l'utilisateur du logiciel à la Configuration du logiciel aux éléments de référence.
*(Facultatif)

SI.3 - Architecture du logiciel et conception détaillée du logiciel

SI.3.1 - Assigner les Tâches aux membres de l'équipe de travail conformément à leurs rôles et selon le plan de projet en vigueur.

SI.3.2 - Comprendre la Spécification des exigences.

SI.3.3 - Documenter ou mettre à jour la Conception du logiciel.

Analyser la Spécification des exigences pour produire la conception de l'architecture, sa disposition dans les sous-systèmes et les Composants logiciels définissant les interfaces internes et externes. Décrire en détail l'apparence et le comportement de l'interface, en fonction de la Spécification des exigences, et de façon à prévoir les ressources nécessaires à sa mise en œuvre.

Fournir les détails des Composants logiciels et de leurs interfaces pour en permettre la construction de façon évidente.

Produire ou mettre à jour l'Enregistrement de la traçabilité.

SI.3.4 - Vérifier la Conception du logiciel et en obtenir l'approbation.

Vérifier l'exactitude de la documentation de la Conception du logiciel, sa faisabilité et sa cohérence avec la Spécification des exigences. S'assurer que l'Enregistrement de la traçabilité contient les liens adéquats entre les exigences et les éléments de la Conception du logiciel. Les résultats sont documentés dans les Résultats de la vérification et des corrections sont apportées jusqu'à ce que le document reçoive l'approbation du concepteur.

Si des changements importants sont requis, une Demande de changement doit être entreprise.

SI.3.5 - Établir ou mettre à jour des Cas et procédures de test pour les tests d'intégration fondés sur la Spécification des exigences et la Conception du logiciel.

Au besoin, le client doit fournir les données de test.

SI.3.6 - Vérifier les Cas et les procédures de test et en obtenir l'approbation.
Vérifier la cohérence entre la Spécification des exigences, la Conception du logiciel et les Cas et procédures de test. Les résultats sont documentés dans les Résultats de la vérification et

des corrections sont apportées jusqu'à ce que le document reçoive l'approbation de l'analyste.

SI.3.7 - Mettre à jour l'Enregistrement de la traçabilité en y incorporant les Cas et procédures de test.

SI.3.8 - Incorporer la Conception du logiciel et l'Enregistrement de la traçabilité à la Configuration du logiciel à titre d'éléments du Dépôt d'information du projet.
Incorporer les Cas et les procédures de test au Dépôt d'information du projet.

SI.4 - Construction du logiciel

SI.4.1 - Assigner les Tâches aux membres de l'équipe de travail conformément à leurs rôles et selon le Plan de projet en vigueur.

SI.4.2 - Comprendre la Conception du logiciel.

SI.4.3 - Construire ou mettre à jour les Composants logiciels conformément aux parties détaillées du document de la Conception du logiciel.

SI.4.4 - Concevoir ou mettre à jour des cas de test unitaire et les mettre en œuvre pour veiller à ce que les Composants logiciels mettent en œuvre la partie détaillée de la Conception du logiciel.

SI.4.5 - Corriger les défauts repérés jusqu'à ce que les tests unitaires soient accomplis avec succès (critères de sortie remplis).

SI.4.6 - Mettre à jour l'Enregistrement de la traçabilité en y ajoutant les composants construits ou modifiés du logiciel.

SI.4.7 - Incorporer les Composants logiciels et l'Enregistrement de la traçabilité à la Configuration du logiciel à titre d'éléments de référence.

SI.5 - Intégration et tests du logiciel

SI.5.1 - Assigner les Tâches aux membres de l'équipe de travail conformément à leurs rôles et selon le Plan de projet en vigueur.

SI.5.2 - Comprendre les Cas et procédures de test.
Mettre en place ou à jour l'environnement de test.

SI.5.3 - Intégrer le Logiciel en utilisant les Composants logiciels et mettre à jour les Cas et procédures de test d'intégration, s'il y a lieu.

SI.5.4 - Réaliser les tests en utilisant les Cas et procédures de test pour l'intégration et documenter les résultats dans le Rapport de test.

SI.5.5 - Corriger les défauts repérés et réaliser des tests de régression jusqu'à ce que les critères de sortie soient remplis.

SI.5.6 - Mettre à jour l'Enregistrement de la traçabilité, s'il y a lieu.

SI.5.7 - Documenter le *Guide d'opération du produit ou mettre à jour le guide en vigueur, s'il y a lieu.
*(Facultatif)

SI.5.8 - Réviser et faire approuver le *Guide d'opération du produit, s'il y a lieu (voir la tâche SI.5.7)
Vérifier la cohérence du Guide d'opération du produit avec le Logiciel. Les résultats sont documentés dans les Résultats de la vérification et des corrections sont apportées jusqu'à ce que le document reçoive l'approbation du concepteur.
*(Facultatif)

SI.5.9 - Documenter la *Documentation de l'utilisateur du logiciel ou mettre à jour le guide en vigueur, s'il y a lieu.
*(Facultatif)

SI.5.10 - Vérifier et obtenir l'approbation pour la *Documentation de l'utilisateur du logiciel, s'il y a lieu (voir la tâche SI.5.9).

Vérifier la cohérence de la *Documentation de l'utilisateur du logiciel avec le Logiciel. Les résultats sont documentés dans les Résultats de la vérification et des corrections sont apportées jusqu'à ce que le document reçoive l'approbation du client.
*(Facultatif)

SI.5.11 - Incorporer les Cas et procédures de test, le Logiciel, l'Enregistrement de la traçabilité, le Rapport de test, le *Guide d'opération du produit et la *Documentation de l'utilisateur du logiciel à la Configuration du logiciel, à titre d'éléments de référence.
*(Facultatif)

SI.6 - Livraison du produit

SI.6.1 - Assigner les Tâches aux membres de l'équipe de travail conformément à leurs rôles et selon le Plan de projet en vigueur.

SI.6.2 - Comprendre la Conception du logiciel.

SI.6.3 - Préparer la Documentation de la maintenance ou mettre à jour les documents en vigueur.

SI.6.4 - Vérifier la Documentation de la maintenance et la faire approuver.

Vérifier la correspondance de la Documentation de la maintenance avec la Configuration du logiciel. Les résultats sont documentés dans les Résultats de la vérification et des corrections sont apportées jusqu'à ce que les documents reçoivent l'approbation du chef technique.

SI.6.5 - Incorporer la Documentation de la maintenance à titre d'élément de référence de la Configuration du logiciel.

SI.6.6 - Livrer le produit conformément aux Directives de livraison.

B. Résumé de la mesure de qualité d'un diagramme d'entité-relations

Ce qui suit est un résumé et une traduction libre des facteurs de qualité d'un ERD. Deux catégories la circonscrivent, soit la qualité d'un ERD produit, l'ontologie et le comportement, le premier affectant le second.

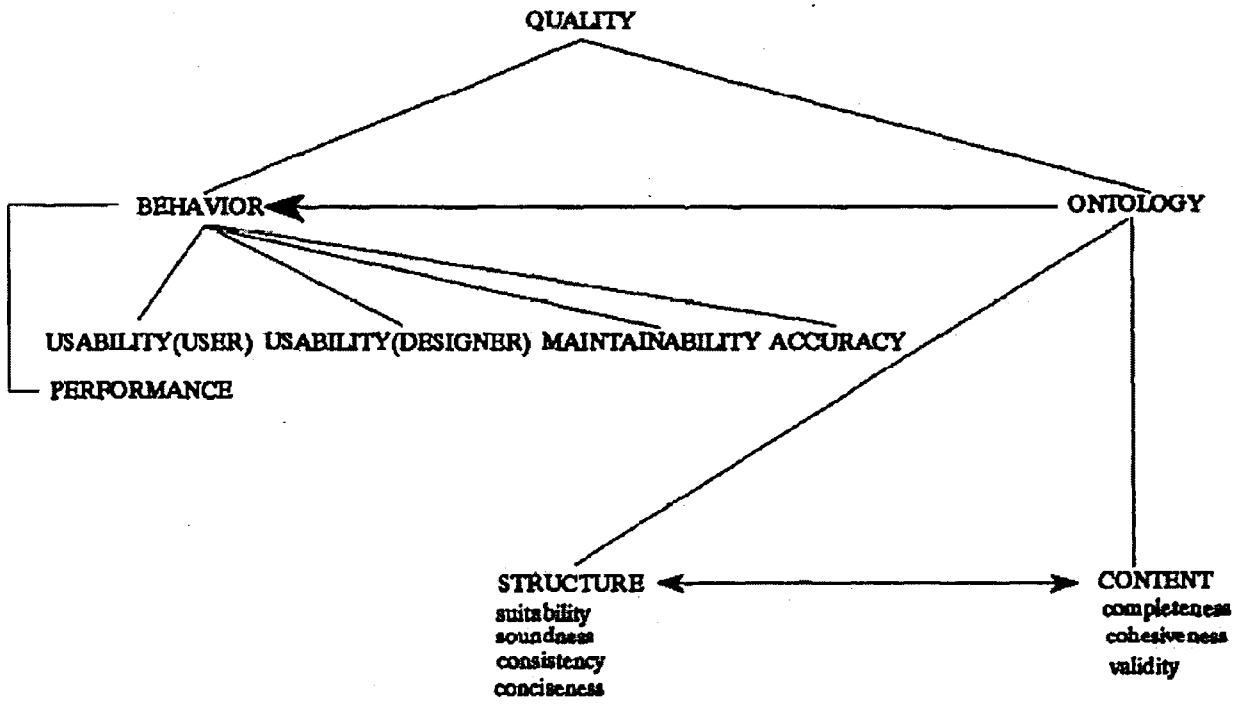


Figure B-1 - Critères de qualité et influence des facteurs de qualité composant un ERD

Tiré de: Information and Software Technology, Vol. 37(12), Evaluating the quality of entity relationship models, P. 3 (Kesh, 1995)

L'ontologie, dans son ensemble, représente les relations définies entre les entités ainsi que leurs cardinalités. Pour ce qui est du comportement, il se définit par l'état de ses relations (statique, soit non modifié lors de l'exécution de requêtes, l'inverse forcerait à user de SQL dynamique pour y arriver) ainsi que les fondements des critères d'affaires dictant les relations entre les entités pendant un laps de temps donné.

Ontologie

Les critères déterminés à travers ces deux sous-catégories souhaitent refléter le comportement du système modélisé.

Structure

La structure cible les entités et ses relations et se divise en quatre sous-catégories. « *Suitability* » souhaite indiquer si le modèle d'entité-relations reflète la problématique à résoudre. « *Soundness* » reflète l'adhérence à des principes de modélisation, ici, le choix de la sélection des entités ainsi que la cardinalité de leurs relations. « *Consistency* » indique si le modèle ne se contredit pas lui-même. Finalement « *Conciseness* » est défini par l'absence de relations redondantes. Un parallèle avec le code applicatif, ce qui est répétitif est un signe que la modélisation est incorrecte (Fowler, 2002). Donc, expliciter plusieurs fois la même information sous divers angles (ou de façon identique) n'est pas souhaité.

Contenu (Content)

Les critères de « *Content* » s'appliquent sur les attributs que les entités détiennent. « *Completness* » indique que chaque entité détient tous les attributs qui lui sont pertinents. « *Cohesiveness* » souhaite capturer la proximité du contexte des attributs « *closeness of the attributes* ». « *Validity* » représente deux facettes. La première facette représente si un attribut donné doit être détenu par l'entité en question. L'exemple donné est le suivant : « Une entité « Étudiant » possédant une liste de cours n'est pas un attribut valide puisque les valeurs des cours devraient être détenues par une entité « Cours » ». La seconde facette représente si les propriétés de l'attribut, type de données « *data type* » et sa longueur (si applicable), sont adéquats.

Comportement (Behavior)

L'article, d'où l'information entourant la qualité du ERD est tirée, scinde le volet comportement en deux sous-catégories avant de poursuivre avec les autres critères de qualité. Pour le présent projet, la description qui en est faite revêt d'une importance non négligeable. Ces deux volets non représentés dans le graphique sont décrits par le comportement « statique » et le comportement « dynamique ».

Le comportement statique indique que les relations dans la base de données demeurent dans le même état tandis que le comportement dynamique concerne les modifications apportées à la base de données *en conséquence de changements dans le domaine d'affaires.* Donc, les relations des tables gérées dynamiquement, telle que la base de données actuelle l'implémente, contreviennent à la définition d'un modèle de qualité.

La convivialité « utilisateur » (Usability User)

Cette caractéristique est définie par la praticité d'utilisation. Est-ce que les utilisateurs sont certains que les exigences ont été respectées.

La convivialité « concepteur » (Usability Designer)

La différence par rapport à la convivialité précédente, est la confiance d'un concepteur face au modèle afin de l'utiliser dans les étapes subséquentes de la modélisation. La raison mentionnée de cette dichotomie est définie par les points de vue différents du modèle présenté entre un utilisateur et un concepteur.

Maintenabilité

L'article définit la maintenabilité comme étant la facilité par laquelle un ERD peut être modifié, corrigé et amélioré. Le niveau de détail de maintenabilité n'est pas davantage décomposé pour rejoindre les sous-catégories (perfective, adaptative, corrective et la quatrième, non mentionnée dans l'article, celle préventive (Bourque, Fairley, 2014), (Trudel, 2017)).

Exactitude (Accuracy)

L'exactitude est définie comme étant un synonyme de fiabilité.

BIBLIOGRAPHIE

AMBLER, By Scott W et SADALAGE, Pramod J, 2006. *Refactoring Databases: Evolutionary Database Design*. S.l. : Addison Wesley Professional. ISBN 9780321293534.

AMBLER, Scott W, 2003. *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. S.l. : s.n. ISBN 0471202835.

APRIL, Alain et LAPORTE, Claude Yvon, 2011. *L'assurance qualité logicielle 1*. Paris : Hermes Science. ISBN 2746231476.

BECK, Kent, BEEDLE, Mike, BENNEKUM, Arie van, COCKBURN, Alistair, CUNNINGHAM, Ward, FOWLER, Martin, GRENNING, James, HIGHSMITH, Jim, HUNT, Andrew, JEFFRIES, Ron, KERN, Jon, MARICK, Brian, C. MARTIN, Robert, MELLOR, Steve, SCHWABER, Ken, SUTHERLAND, Jeff et THOMAS, Dave, 2001. Manifesto for Agile Software Development. In : [en ligne]. 2001. [Consulté le 8 janvier 2019]. Disponible à l'adresse : http://agilemanifesto.org/.

BOISVERT, Mathieu et TRUDEL, Sylvie, 2011. *Choisir l'agilité*. S.l. : Dunod. ISBN 978-2-10-055850-6.

BOURQUE, Pierre et FAIRLEY, Richard E. (éd.), 2014. *Guide to the Software Engineering Body of Knowledge, Version 3.0* [en ligne]. S.l. : IEEE Computer Society. ISBN 0-7695-2330-7. Disponible à l'adresse : www.swebok.org.

CARPENTER, Donald a, 2008. Clarifying Normalization. In : *Journal of Information Systems Education*. 2008. Vol. 19, n° 4, p. 379-383.

CHILTON, Michael a, 2006. Data Modeling Using Entity Relationship Diagrams: A Step-Wise Method. In : *Journal of Information Systems Education* [en ligne]. 2006. Vol. 17, n° 4, p. 385-394. Disponible à l'adresse : http://search.proquest.com/docview/200102759?accountid=28180%5Cnhttp://xt6nc6eu9q.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info:ofi/enc:UTF-8&rfr_id=info:sid/ProQ:abiglobal&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.jti.

CLEVE, Anthony, GOBERT, Maxime, MEURICE, Loup, MAES, Jerome et WEBER, Jens, 2015. Understanding database schema evolution: A case study. In : *Science of Computer Programming* [en ligne]. 2015. Vol. 97, n° P1, p. 113-121. DOI 10.1016/j.scico.2013.11.025. Disponible à l'adresse : http://dx.doi.org/10.1016/j.scico.2013.11.025.

COTELEA, Vitalie, 2012. Contributions to Logical Database Design. In : *Informatica Economica Journal* [en ligne]. 2012. Vol. 16, n° 1, p. p.14. Disponible à l'adresse : http://disco.uni-muenster.de/primo_library/libweb/action/display.do?tabs=detailsTab&ct=display&fn=search&doc=TN_doajb5b516729d2596d8f4fda0304760d226&indx=11&recIds=TN_doajb5b516729d2596d8f4fda0304760d226&recIdxs=0&elementId=0&renderMode=poppedOut&displayM.

DATE, C. J. et FAGIN, Ronald, 2002. Simple conditions for guaranteeing higher normal forms in relational databases. In : *ACM Transactions on Database Systems*. 2002. Vol. 17, n° 3, p. 465-476. DOI 10.1145/132271.132274.

DE LUCIA, Andrea, GRAVINO, Carmine, OLIVETO, Rocco et TORTORA, Genoveffa, 2010. An experimental comparison of ER and UML class diagrams for data modeling. In : *Empirical Software Engineering*. 2010. Vol. 15, n° 5, p. 455-492. DOI 10.1007/s10664-009-9127-7.

DEFRANCO, Joanna F. et LAPLANTE, Philip A., 2017. Review and analysis of software development team communication research. In : *IEEE Transactions on Professional Communication*. 2017. Vol. 60, n° 2, p. 165-182. DOI 10.1109/TPC.2017.2656626.

DEHAGHANI, Sayed Mehdi Hejazi et HAJRAHIMI, Nafiseh, 2013. Which factors affect software projects maintenance cost more? In : *Acta Informatica Medica*. 2013. Vol. 21, n° 1, p. 63-66. DOI 10.5455/aim.2012.21.63-66.

DIEDERICH, Jim et MILTON, Jack, 2002. New methods and fast algorithms for database normalization. In : *ACM Transactions on Database Systems*. 2002. Vol. 13, n° 3, p. 339-365. DOI 10.1145/44498.44499.

EVANS, Eric, 2003. *Domain Driven Design: Tackling Complexity in the Heart of Business Software*. 1 edition. S.l. : Addison-Wesley Professional. ISBN 0-321-12521-5.

FAGIN, Ronald, 2002. A normal form for relational databases that is based on domains and keys. In : *ACM Transactions on Database Systems*. 2002. Vol. 6, n° 3, p. 387-415. DOI 10.1145/319587.319592.

FOWLER, Martin, 2002. *Patterns of Enterprise Application Architecture*. S.l. : Addison-Wesley Professional; 1 edition (Nov. 5 2002). ISBN 0321127420.

GALVAN, Sergio, MORA, Manuel, O'CONNOR, Rory V., ACOSTA, Francisco et ALVAREZ, Francisco, 2015. A Compliance Analysis of Agile Methodologies with the ISO/IEC 29110 Project Management Process. In : *Procedia Computer Science* [en ligne]. 2015. Vol. 64, p. 188-195. DOI 10.1016/j.procs.2015.08.480. Disponible à l'adresse : http://dx.doi.org/10.1016/j.procs.2015.08.480.

GLASS, R.L., 2001. Frequently forgotten fundamental facts about software engineering. In : *IEEE Software* [en ligne]. mai 2001. Vol. 18, n° 3, p. 112-111. DOI 10.1109/MS.2001.922739. Disponible à l'adresse : http://ieeexplore.ieee.org/document/922739/.

GREEN, Rebecca, 1996. The design of a relational database for large-scale bibliographic retrieval. In : *Information Technology and Libraries* [en ligne]. 1996. Vol. 15, n° 4, p. 207+. Disponible à l'adresse : http://go.galegroup.com/ps/i.do?id=GALE%7CA19133383&v=2.1&u=berkeley_main&it=r&p=AONE&sw=w&asid=d129a311e7e114b9a39ef97d4c442925%5Cnhttp://go.galegroup.com/ps/retrieve.do?sgHitCountType=None&sort=DA-SORT&inPS=true&prodId=AONE&userGroupName=berkeley_main&t.

HINKLE, Matthew M, 2007. Software Quality, Metrics, Process Improvement, and CMMI: An Interview with Dick Fairley. In : *IT Professional* [en ligne]. mai 2007. Vol. 9, n° 3, p. 47-51. DOI 10.1109/MITP.2007.57. Disponible à l'adresse : http://ieeexplore.ieee.org/document/4216109/.

IEEE COMPUTER SOCIETY, 2012. *IEEE P730/D8 Draft Standard for Software Quality Assurance Processes*. S.l. : s.n. ISBN 978-1-903274-31-6.

ISO/IEC/IEEE 31320-2:2012(E), 2012. *INTERNATIONAL STANDARD ISO / IEC / IEEE Information technology — Modeling*. 1st. S.l. : s.n. ISBN 0738103403.

JONSSON, Dan, 1992. Formal syntax and semantics of a reconstructed relational database system. In : *ACM SIGMOD Record* [en ligne]. 1 mars 1992. Vol. 21, n° 1, p. 84-89. DOI 10.1145/130868.130879. Disponible à l'adresse : http://portal.acm.org/citation.cfm?doid=130868.130879.

KARWIN, Bill, 2010. *SQL Antipatterns: Avoiding the Pitfalls of Database Programming* [en ligne]. 1 edition. S.l. : Pragmatic Bookshelf. ISBN 1934356557. Disponible à l'adresse : http://it-ebooks.info/book/70/%5Cnpapers3://publication/uuid/7F34FB0E-A06F-4E46-8E59-2F2CDBCDB5AE.

KESH, Someswar, 1995. Evaluating the quality of entity relationship models. In : *Information and Software Technology*. 1995. Vol. 37, n° 12, p. 681-689. DOI 10.1016/0950-5849(96)81745-9.

KÖHLER, Henning et LINK, Sebastian, 2018. SQL schema design: foundations, normal forms, and normalization. In : *Information Systems*. 2018. Vol. 76, p. 88-113. DOI 10.1016/j.is.2018.04.001.

KRAUT, Robert E. et STREETER, Lynn A., 1995. Coordination in software development. In : *Communications of the ACM* [en ligne]. 1995. Vol. 38, n° 3, p. 69-81. DOI 10.1145/203330.203345. Disponible à l'adresse : http://portal.acm.org/citation.cfm?doid=203330.203345.

LAPORTE, Claude Y., MUNOZ, Mirna, MEJIA MIRANDA, Jezreel et OCONNOR, Rory V., 2017. Applying Software Engineering Standards in Very Small Entities: From Startups to Grownups. In : *IEEE Software*. 2017. Vol. 35, n° 1, p. 99-103. DOI 10.1109/MS.2017.4541041.

LAPORTE, Claude Y. et O'CONNOR, Rory V., 2016. Systems and Software Engineering Standards for Very Small Entities: Accomplishments and Overview. In : *Computer* [en ligne]. août 2016. Vol. 49, n° 8, p. 84-87. DOI 10.1109/MC.2016.242. Disponible à l'adresse : http://ieeexplore.ieee.org/document/7543423/.

LEE, Heeseok, 1995. Justifying database normalization: a cost/benefit model. In : *Information Processing and Management*. 1995. Vol. 31, n° 1, p. 59-67. DOI 10.1016/0306-4573(95)80006-F.

LEE, Taehee, LEE, Ig Hoon, LEE, Suekyung, LEE, Sang Goo, KIM, Dongkyu, CHUN, Jonghoon, LEE, Hyunja et SHIM, Junho, 2006. Building an operational product ontology system. In : *Electronic Commerce Research and Applications*. 2006. Vol. 5, n° 1, p. 16-28. DOI 10.1016/j.elerap.2005.08.005.

MARTIN, Robert C, FEATHERS, Michael C, OTTINGER, Timothy R, GRENNING, James W et DEAN, Kevin, 2009. *Coder Proprement*. S.l. : Pearson. ISBN 9782744041044.

MCMURDO, George, 1982. Database file normalization as an information science related activity. In : *Journal of Information Science* [en ligne]. 5 janvier 1982. Vol. 4, n° 1, p. 9-17. DOI 10.1177/016555158200400103. Disponible à l'adresse : http://journals.sagepub.com/doi/10.1177/016555158200400103.

MESQUIDA, Antoni-Lluís et MAS, Antonia, 2014. A project management improvement

program according to ISO/IEC 29110 and PMBOK ®. In : *Journal of Software: Evolution and Process* [en ligne]. septembre 2014. Vol. 26, n° 9, p. 846-854. DOI 10.1002/smr.1665. Disponible à l'adresse : http://doi.wiley.com/10.1002/smr.1665.

OMIECINSKI, Edward R., 1990. *A Parallel Algorithm for Relational Database Normalization.* 1990. S.l. : s.n.

ORGANISATION INTERNATIONALE DE NORMALISATION, 2012. Ingénierie du logiciel--profils de cycle de vie pour très petits organismes (TPO). Partie 5-1-1, Guide de gestion et d'ingénierie : groupe de profil générique : Profil d'entrée. In : *Guide de gestion et d'ingénierie : groupe de profil générique : Profil d'entrée.* 2012. Vol. 2011.

PANCHENKO, B. E., 2012a. Framework design of a domain-key schema of a relational database. In : *Cybernetics and Systems Analysis.* 2012. Vol. 48, n° 3, p. 469-478. DOI 10.1007/s10559-012-9426-7.

PANCHENKO, B. E., 2012b. Investigating a domain-key scheme of a relational database. In : *Cybernetics and Systems Analysis* [en ligne]. 28 novembre 2012. Vol. 48, n° 6, p. 943-955. DOI 10.1007/s10559-012-9476-x. Disponible à l'adresse : http://link.springer.com/10.1007/s10559-012-9476-x.

PAREDAENS, Jan, BRA, Paul, GYSSENS, Marc et GUCHT, Dirk, 1989. *The Structure of the Relational Database Model* [en ligne]. Berlin, Heidelberg : Springer Berlin Heidelberg. ISBN 978-3-642-69958-0. Disponible à l'adresse : http://link.springer.com/10.1007/978-3-642-69956-6.

REDA, A., 2003. Extracting the extended entity-relationship model from a legacy relational database. In : *Information Systems.* 2003. Vol. 28, n° 6, p. 597-618.

ROBERT, Ballenger, 2007. An eCommerce Development Case : Your Company ' s eCommerce Web Site. In : *Journal of Information Systems Education.* 2007. Vol. Vol. 18(4), n° Winter, p. 409.

ROLLAG, Keith, PARISE, Salvatore et CROSS, Rob, 2005. Getting New Hires Up to Speed Quickly. In : *MIT Sloan Management Review.* 2005. Vol. 46, n° 2, p. 35.

SCHRAML, Todd, 2014. There Are Many Tools in the Toolbox-Use More Than One. In : *Database Trends and Applications; Chatham* [en ligne]. 2014. Vol. 28, n° May, p. 1. Disponible à l'adresse : https://search-proquest-com.proxy.bibliotheques.uqam.ca/docview/1528493123?accountid=14719.

SHMELTZER, A Shay, 2018. Agile Development and the Modern Database. In : *Database Trends and Applications.* 2018. Vol. 32, n° September, p. 36-37.

SPRINGSTEEL, F.N., 2003. *Entity-relationship logical design of database systems: relational normal forms and extended regular ERDs.* 2003. S.l. : s.n. ISBN 0818608420.

TAKEUCHI, Motoko, KOHTAKE, Naohiko, SHIRASAKA, Seiko, KOISHI, Yumi et SHIOYA, Kazunori, 2014. Report on an assessment experience based on ISO/IEC 29110. In : *Journal of Software: Evolution and Process* [en ligne]. mars 2014. Vol. 26, n° 3, p. 306-312. DOI 10.1002/smr.1591. Disponible à l'adresse : http://doi.wiley.com/10.1002/smr.1591.

TEMPERO, Ewan, GORSCHEK, Tony et ANGELIS, Lefteris, 2017. Barriers to refactoring. In : *Communications of the ACM* [en ligne]. 2017. Vol. 60, n° 10, p. 54-61.

DOI 10.1145/3131873. Disponible à l'adresse : http://dl.acm.org/citation.cfm?doid=3144574.3131873.

TEOREY, Toby J., YANG, Dongqing et FRY, James P., 2002. A logical design methodology for relational databases using the extended entity-relationship model. In : *ACM Computing Surveys*. 2002. Vol. 18, n° 2, p. 197-222. DOI 10.1145/7474.7475.

THOMPSON, Cheryl Bagley et SWARD, Katherine, 2005. Modeling and teaching techniques for conceptual and logical relational database design. In : *Journal of Medical Systems*. 2005. Vol. 29, n° 5, p. 513-525. DOI 10.1007/s10916-005-6108-3.

TRUDEL, Sylvie, 2017. *Mesure d ' attributs externes du produit*. 2017. S.l. : s.n.

TRUDEL, Sylvie, LAVOIE, Jean Marc, PARÉ, Marie Claude et SURYN, Witold, 2006. PEM: The small company-dedicated software process quality evaluation method combining CMMISMand ISO/IEC 14598. In : *Software Quality Journal*. 2006. Vol. 14, n° 1, p. 7-23. DOI 10.1007/s11219-006-5997-8.

VIAL, Gregory, 2015. Database Refactoring: Lessons from the Trenches. In : *IEEE Software*. 2015. Vol. 32, n° 6, p. 71-79. DOI 10.1109/MS.2015.131.

WANG, Ting J. (T.J.), DU, Hui et LEHMANN, Constance M., 2016. Accounting For The Benefits Of Database Normalization. In : *American Journal of Business Education (AJBE)*. 2016. Vol. 3, n° 1. DOI 10.19030/ajbe.v3i1.371.

WIEGERS, Karl et BEATTY, Joy, 2013. *Software Requirements, 3rd Edition* [en ligne]. S.l. : s.n. ISBN 978-0-7356-7966-5. Disponible à l'adresse : https://www.microsoftpressstore.com/store/software-requirements-9780735679665.

YEH, Dowming, LI, Yuwen et CHU, William, 2008. Extracting entity-relationship diagram from a table-based legacy database. In : *Journal of Systems and Software*. 2008. Vol. 81, n° 5, p. 764-771. DOI 10.1016/j.jss.2007.07.005.