Jonathan Clavet-Grenier

16 070 207

Marc Dupuis

08 387 435

Base de données transactionnelle

Modélisation de bases de données

Rapport

Dans le cadre du cours IGE487

Université de Sherbrooke

2019-11-21

Table des matières

[Introduction 3](#_Toc23071120)

[Problème 4](#_Toc23071121)

[Hypothèse 5](#_Toc23071122)

[Schéma conceptuel 6](#_Toc23071123)

[Analyse argumentée pour le schéma conceptuel 7](#_Toc23071124)

[Schéma Logique 8](#_Toc23071125)

[Analyse argumentée pour le schéma logique 9](#_Toc23071126)

[Conclusion 10](#_Toc23071127)

# Introduction

Dans le cadre du cours d’IGE487, il nous fut impartie la tâche de développer une architecture de base de données afin de satisfaire les besoins d’une entreprise fictives en termes de gestion de développement logiciel. À ces fins, nous avons reçu carte blanche afin d’élaborer le prototype d’une solution en trois temps; d’abord de développer une base de données de type opérationnelle, puis d’une base de données dite analytique et pour finir le tout, d’une base de données de type historicisée.

Le présent rapport présente les résultats des différents prototypes de base de données (opérationnelle, analytique, et éventuellement historicisé). Afin de mieux cerner l’enjeu se cachant derrière le désir de la compagnie de développer un tel logiciel, nous présenterons brièvement la problématique accompagnée de nos hypothèses vis-à-vis de celle-ci. S’en suivra une présentation de deux diagrammes de conceptions; un diagramme conceptuel et un diagramme logique.

# Problème

Le fond du problème, tel qu’identifié par la compagnie, est que les logiciels actuels sont trop restrictifs dans leur manière d’aborder la gestion de projet. En prenant comme exemple le logiciel de gestion de projet ProjetQtor, on peut remarquer qu’il y a au moins trois constats apparents qui peuvent être soulevés.

Premièrement, le logiciel est considérablement complexe dans sa manière d’aborder la problématique; beaucoup trop d’outils sont mis à la disposition du gestionnaire ce qui occasionne une confusion profonde pour le gestionnaire et ce qui fait en sorte que le logiciel n’est pas utilisé à son plein potentiel.

Deuxièmement, le logiciel est beaucoup trop restrictif. Avec comme objectif d’encourager de saines habitudes de gestion de projet, l’équipe de développement du logiciel ont obligé la saisie de plusieurs champs, ont restreint la modification de certains champs selon l’état de certains objets et ont même restreint le « flow » de travail en empêchant de revenir à des états de projet et de tâches à priori. Par conséquent, la gestion de projet est très difficile, voire impossible à réaliser et se résume à : « Apprends à vivre avec tes erreurs »

Troisièmement, certains aspects analytiques sont complexes à mettre en place dans les logiciels existants. Il devrait être possible de faciliter la vie de tous les jours des analystes et des gestionnaires de l’entreprise.

# Hypothèse

À ce stade, nos hypothèses sont les suivantes :

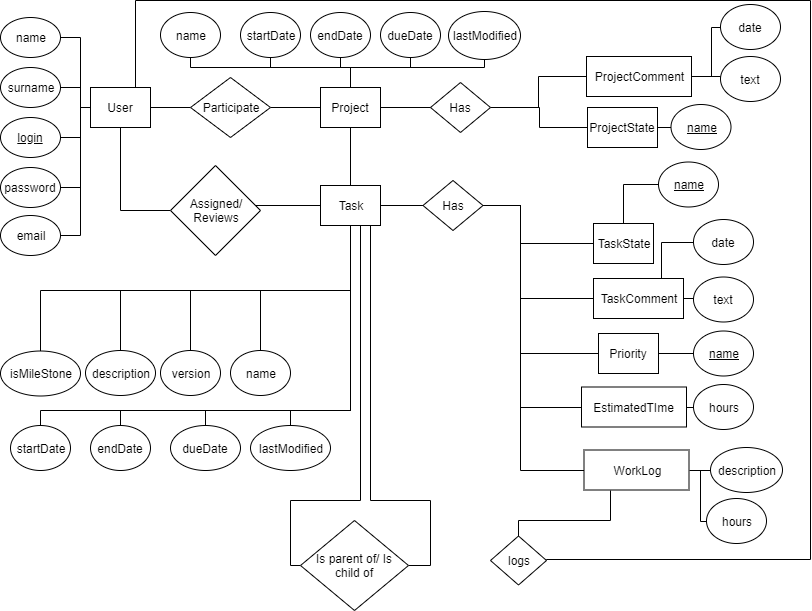
**HY.01** Il est possible de réaliser un logiciel de gestion de projet simple qui nous conviendra.

**HY.02** D’un raisonnement naïf, il est peu probable d’élaborer un système de gestion de projet convenant à toutes les méthodes de développement logiciel.

**HY.03** Un encadrement de gestion de projet plus flexible mènera à une meilleure gestion de projet.

**HY.04** Des outils analytique clés en main amélioreront la productivité des gestionnaires.

# Schéma conceptuel



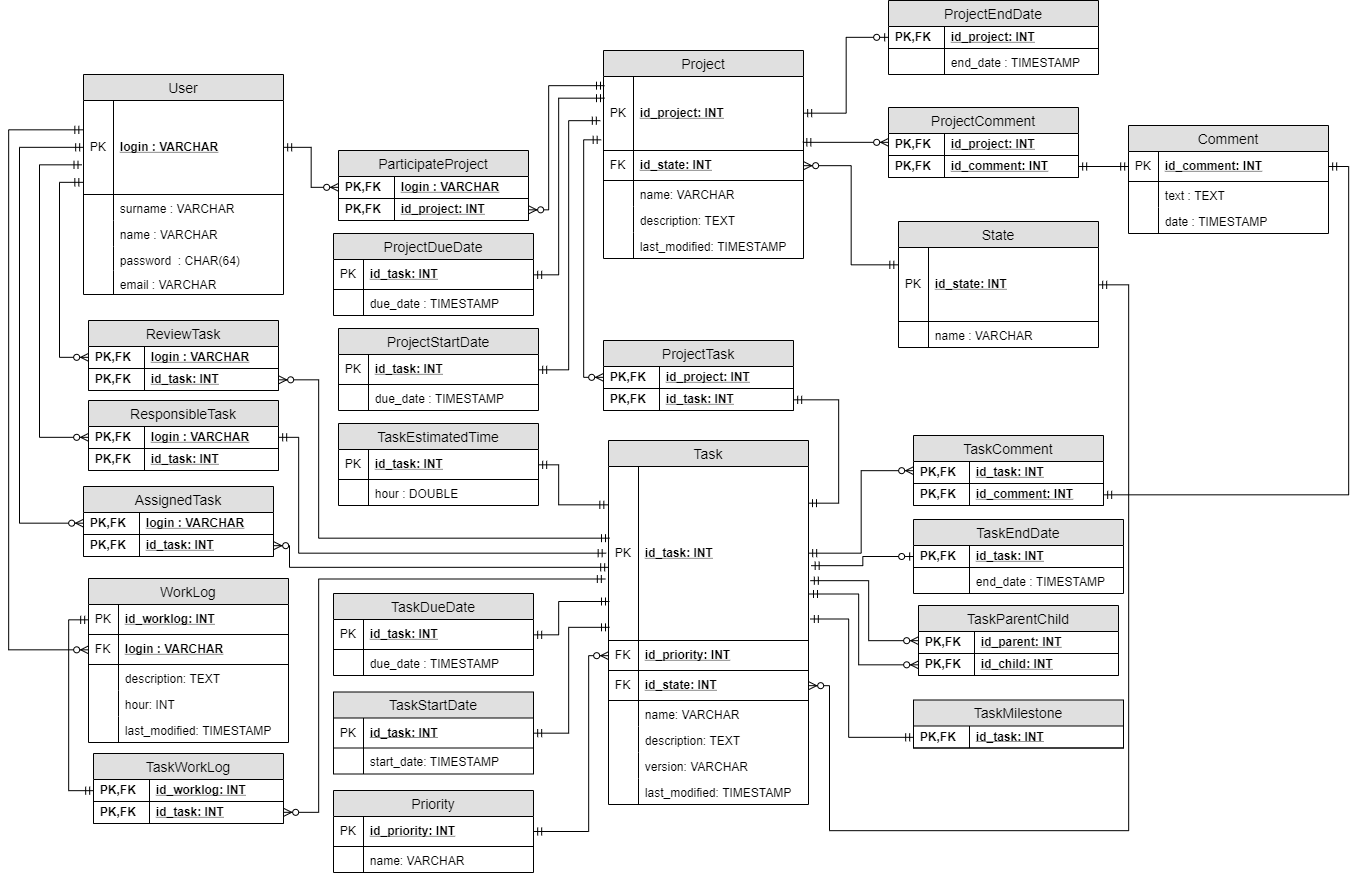
# Analyse argumentée pour le schéma conceptuel

L’une des premières décisions que nous dûmes prendre fut la représentation d’un jalon. Le premier constat qui nous apparût fut que la gestion d’un projet s’apparente très approximativement d’une gestion de système de fichier; un projet est en quelque sorte un dossier racine, une tâche est ou bien un dossier, ou bien un fichier et de même pour un jalon.

Ainsi, nous avons fais le choix de conception suivant :

1. Un projet est une relation distincte d’une tâche; nous ne désirons pas permettre l’existence de sous-projet.
2. Un jalon et une tâche sont tous deux représentés dans la même relation puisqu’ils sont conceptuellement pratiquement identiques. À cette étape, nous n’avons pas encore décidé de comment nous allons traiter l’attribut (dans une table ou en attribut).
3. Une tâche peut avoir des tâches parentes et implicitement, des tâches enfants.
4. Nous mettons de côté la gestion des rôles pour les usagers à une implémentation subséquente de notre prototype.

# Schéma Logique



# Analyse argumentée pour le schéma logique

Après l’élaboration du schéma conceptuel, des détails de conception nous restaient à définir. Le plus gros morceau fût la conception d’une tâche. Nous dûmes faire certains choix afin de respecter notre problématique :

1. Afin de conserver une forme normale tout en interdisant toute forme de champs « null », nous avons éclaté la relation tâche en plusieurs relations; chacune de ces relations nous permettront donc de laisser au gestionnaire le choix de remplir un champ donné. Par exemple, si un gestionnaire désire écrire une tâche pour son « backlog », il n’aura pas besoin de mettre une date de début à sa tâche.
2. Projet et Tâche partage la relation d’État ainsi qu’une relation de Commentaire.
3. Lors d’une modification d’une des sous-tables de tâche et de projet, le champ « last\_modified » doit être mis à jour pour la tâche correspondante.
4. Une seule personne peut être responsable d’une tâche, c’est le responsable de la tâche.
5. Plusieurs personnes peuvent être assignées à une tâche ou réviser une tâche.
6. Une personne qui n’est pas assignée à une tâche peut quand même entrer du temps sur une tâche. Particulièrement utile si par exemple, un programmeur demande de l’aide à un deuxième programmeur et que l’aide prodiguée par le deuxième programmeur est considérable.

# Schéma logique pour la base de données analytique

# Argumentation du schéma pour la base de données analytique

Nous nous sommes dotés de deux « data mart » pour notre base de données analytique. Ce que nous souhaitons pouvoir analyser à partir de ces deux dépôts de données, ce sont deux concepts; la semaine de travail d’un employé donné en fonction du temps rentré dans ses tâches, et le temps enregistré par tâche selon un projet, un état de tâche et une priorité. Ceci permettrait notamment à un gestionnaire ou à un analyste d’analyser la progression de ses employés en lien à leur semaine de travail Ceci pourrait potentiellement leur permettre d’identifier des goulots d’étranglement, mais aussi de livrer une analyse plus poussée quant à l’état d’un projet selon les tâches.

# Conclusion

Notre logiciel étant un logiciel de gestion de projet en informatique a pour élément centrale la gestion des tâches au sein de projet. Ainsi, les choix que nous avons opérés sur ce concept sont particulièrement important. Nous sommes particulièrement contents du choix de conception que nous avons finis par adopter vis-à-vis les relations parents/enfants. Le choix naïf que nous avons pris initialement était passablement plus complexe et présentait une redondance; notre premier choix envisageait l’utilisation de deux tables comportant des colonnes « task\_id » et « parent\_id » ou « child\_id ».

En ce qui concerne le volet analytique, nous croyons avoir été en mesure de développer une base de données en schéma étoile intéressant. Nous avons notamment orienté notre réflexion envers les métriques qu’un gestionnaire pourrait souhaiter voire apparaître dans un panneau de contrôle de projet.