13. Exemple: Gestion de projets

Dans ce chapitre nous présentons un autre exemple illustrant comment procéder pour analyser un problème et concevoir la partie de la base de données qui y a trait.

L'exemple couvre la saisie et la facturation d'heures effectuées pour des projets clients, par exemple par un cabinet de conseil, un bureau d'ingénieurs ou une société de services informatiques.

Il est évidemment impossible de couvrir ici ce problème dans ses moindres détails. Le résultat n'est donc pas une solution à utiliser telle quelle. L'important est de comprendre la démarche et les raisonnements effectués. Il est très probable d'ailleurs que ce problème ne soit, dans la réalité, jamais traité de manière totalement isolée: il s'intégrera presque certainement à d'autres fonctions et systèmes. Pour cet exercice, nous agissons comme s'il s'agissait d'un ilot. De même, nous ne traitons pas la facturation proprement dite, mais nous limitons à la saisie et à l'élaboration des données qui serviront à cette facturation.

Donc: la société en question effectue des projets pour ses clients. Pour réaliser ces projets, elle engage ses collaborateurs qui doivent enregistrer par jour les heures qu'ils consacrent à ces projets. Ces enregistrements servent à la facturation. Exigence: automatiser dans la mesure du possible la facturation des heures en travaillant avec des tarifs prédéfinis.

Inventaire des données

L'énoncé ci-dessus nous fournit immédiatement les types d'objets principaux:

Client

Projet

Collaborateur

Travail (heures effectuées)

Tarif?

Nous repoussons à plus tard la discussion des tarifs pour la facturation. Parmi les types d'objets découverts, deux nous paraissent élémentaires au sens du chapitre 9: *client* et *collaborateur*. *Projet* est évidemment rattaché à *client* et *travail* rattaché à *projet* et à *collaborateur*. S'il ne s'agissait pas d'un ilot, il existerait probablement déjà des types d'objets *client* et *collaborateur* dans la base de données.

Que faut-il savoir du client? Une discussion avec les futurs utilisateurs nous fournit les éléments suivants: raison sociale (2 lignes), rue, numéro, code postal, localité, nom personne contact, langue, no téléphone, fax, e-mail.

Que faut-il savoir du collaborateur? Seulement son numéro d'employé, nom et prénom.

Pour les projets, nous enregistrons: no de projet, client, nom, état, date début, date fin.

Et pour *travail*: projet, collaborateur, date, nombre d'heures.

Premier modèle

Nous posons les types d'objets suivants:

Client (#-client, raison sociale 1, raison sociale 2, rue, numéro, code postal, localité, personne contact, langue, téléphone, fax, e-mail

Collaborateur (#-employé, nom, prénom

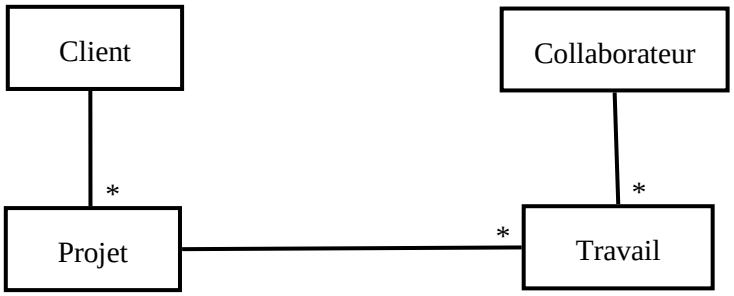
Projet (#-projet, *#-client*, nom projet, état, date début, date fin

Travail (#-travail, *#-projet*, *#-collaborateur*, date, heures

Nous avons évidemment appliqué l'appris des chapitres précédents:

- Nous avons assigné des clés primaires à chaque type d'objet. Le client n'avait pas de numéro, nous en créons un. Pour le collaborateur, nous utilisons le numéro existant d'employé. Pour le projet, nous reprenons le numéro de projet existant. Pour le travail, nous créons un nouveau numéro neutre qui sera incrémenté automatiquement par le système chaque fois qu'un collaborateur enregistre ses heures. Une alternative serait de choisir la concaténation *#-projet*, *#-collaborateur*, *date* comme clé primaire.
- Nous décrivons les dépendances (associations) au moyen de clé étrangères: *projet* à *client* en insérant *#-client* dans *projet*. *Travail* à *projet* et à *collaborateur* en insérant *#-projet* et *#-collaborateur* dans *travail*.

Le schéma se présente comme suit:



Un client peut posséder plusieurs projets, un projet n'appartient qu'à un client.

Un travail concerne un projet, plusieurs travaux sont exécutés pour un projet.

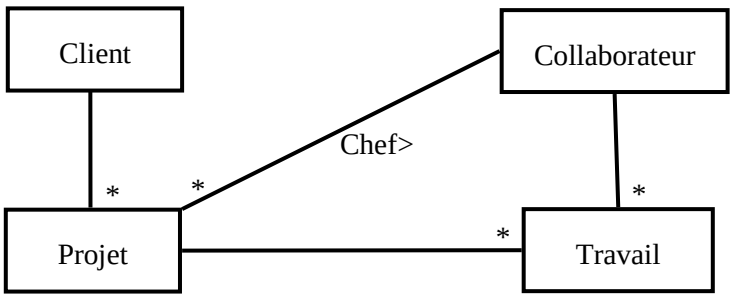
Un collaborateur exécute plusieurs travaux, un travail est exécuté par un collaborateur.

Étudiant le détail des définitions de types d'objets, *état* dans *projet* attire notre attention. Faut-il formaliser cet attribut, donc limiter sa valeur à une liste de valeurs prédéfinies (commandé, planifié, démarré, terminé, etc.). Si c'est le cas, nous créons une sous-table pour les valeurs de *état* dans la table des codes (voir chapitre 7) et nous remplaçons *état* par *#-état* dans la définition de *projet*.

Chef de projet

Lors de la présentation du modèle aux futurs utilisateurs, l'un d'entre eux remarque qu'il serait utile de pouvoir enregistrer le chef de chaque projet. Il s'agit évidemment d'une association entre *collaborateur* et *projet*. Chaque projet ne peut avoir qu'un seul chef, mais un collaborateur peut diriger plusieurs projets.

Projet (#-projet, *#-client*, nom projet, *#-état*, date début, date fin, *#-collaborateur chef projet*



Tarifs et facturation

Venons en finalement à la discussion des tarifs et de la facturation. La discussion avec les futurs utilisateurs à ce sujet apporte les éléments suivants:

- On aimerait enregistrer pour chaque projet un budget global et un taux horaire de facturation.
- Il est nécessaire de pouvoir spécifier pour chaque travail un code indiquant s'il est facturable ou non et un taux horaire de facturation, normalement celui du projet, mais qui peut être différent.

Nous complétons donc les définitions des types d'objets *projet* et *travail* comme suit:

Projet (#-projet, *#-client*, nom projet, *#-état*, date début, date fin, *#-collaborateur chef projet*, budget, taux facturation

Travail (#-travail, *#-projet*, *#-collaborateur*, date, heures, *#code facturable*, taux facturation

Formalisant *code facturable*, nous en avons, comme d'habitude, fait une sous-table de la table des codes et donc une clé étrangère dans *travail*.

Évidemment, le problème de la facturation peut être beaucoup plus complexe. On pourrait imaginer des taux liés à chaque collaborateur ou encore un taux liant le collaborateur au projet. Cette dernière solution entraînerait la création d'un nouveau type d'objet associatif *collaborateur – projet*. Nous laissons au lecteur comme exercice la spécification exacte de la solution dans ce cas.