

Table des matières

Constat.....	1
Le modèle universel.....	1
Les traitements universels.....	1
Génération automatique de formulaire.....	1
La récupération des données.....	1
La recherche de toute donnée.....	1
Les traitements spécifiques ou additionnels.....	1
Points divers, améliorations.....	1
Eclatement et regroupement.....	1
Typage physique.....	1
Images, Blobs.....	1
Performance.....	1
Création automatisée de fiches du méta modèle.....	2

Le modèle universel

Constat

L'informatique de gestion suppose une bonne organisation des données afin de mémoriser ces données au mieux, c'est à dire en un seul exemplaire par donnée élémentaire et avec une possibilité de retrouver celle-ci facilement et rapidement. Cet objectif est largement atteint grâce aux bases de données.

La bonne organisation des données consiste à créer dans la base des tables correspondants aux entités manipulées par les applications de gestion. Par exemple, table des clients, table des articles, table des commandes passées par les clients, table des lignes de commande faisant le lien entre commandes et articles commandés, etc ... Chacune des tables comprend des colonnes telles que Nom, Prénom, Adresse, ... dans le cas de la table clients. La construction des tables obéit à des règles strictes permettant d'identifier de façon unique toute les données d'une ligne par la connaissance d'un ou plusieurs champs clés. Pour chaque application informatique, on peut donc construire un modèle de données comprenant plusieurs tables reliées entre elles le plus souvent par des relations.

Dès que l'application évolue fortement, il est nécessaire de revoir ce modèle soit en ajoutant quelques colonnes supplémentaires aux tables existantes, soit en ajoutant de nouvelles tables et de nouvelles relations. L'art du concepteur de modèle de données réside dans le fait que son modèle résiste assez bien aux changements et nécessite que très peu de modifications du modèle.

Néanmoins, on comprend bien que le modèle de données est spécifique de l'application et qu'il n'est pas immuable dans toute la durée de vie de l'application.

A l'inverse, si nous pouvions imaginer un modèle de données universel, qui soit valable pour toute application, nous disposerions d'un modèle qui a de fortes chances d'être résistant au changement.

Le modèle universel

Ce modèle universel existe et il est le suivant :

Une entité (ou table) appelée **Objet** comprend les champs ou colonnes suivants :

Id clé unique technique identifiant parfaitement toute ligne de l'entité.
Id_pere clé étrangère référençant le père et/ou le type des occurrences de **Objet**
Donnee champ recevant toute donnée élémentaire **Objet**

Par exemple ; pour mémoriser dans cette table des noms et prénoms de personne ainsi que leur numéros de téléphone on pourrait avoir

Id	Id_pere	Donnee
1		<i>Racine des types</i>
2	1	<i>(Type) Nom</i>
3	1	<i>(Type) Prénom</i>
4	1	<i>(Type) téléphone</i>
5	2	Dupont
6	3	Christian
7	4	0320456789
8	2	Durant
9	3	Marcel
10	4	0430234567
...		... et ainsi de suite

Le modèle à une entité **Objet** proposé permet bien de ranger les données et d'identifier leur type. Mais ce modèle ne permet pas de savoir que Dupont, Christian et 0320456789 sont des données concernant la même personne !

Il faut adjoindre à notre modèle une seconde entité établissant les liens entre les données élémentaires ; il faut également faire apparaître un chapeau des Dupont, Christian et 0320456789 que l'on appellera *fiche contact* par exemple.

La **table *Objet*** s'enrichit donc du type *fiche* et de ses 2 occurrences comme ceci :

11	1	<i>Type de fiche</i>
12	11	Fiche de Dupont
13	11	Fiche de Durant
...		

Il faut ajouter une table **Relation** qui comprend les champs suivants :

Id clé unique technique identifiant parfaitement toute ligne de l'entité.
Origine clé de l'occurrence origine ou composé (ce sera la *fiche* dans notre exemple)
Destination clé de l'occurrence destination ou composant (nom, prénom, téléphone d'une personne). Cette deuxième table ressemblera donc à ceci :

Table Relation

Id	Origine	Destination
-----------	----------------	--------------------

1	<i>11</i>	2
2	<i>11</i>	3
3	<i>11</i>	4
4	12	5
5	12	6
6	12	7
7	13	8
8	13	9
9	13	10
...		... et ainsi de suite

Les 3 premières lignes définissent les relations entre les types, une façon de mémoriser le méta modèle correspondant à la fiche contact et ses données liées. Pour bien matérialiser la différence entre le méta modèle et le modèle, les données du méta modèle sont en italiques alors que les données du modèle sont en écriture droite normale.

L'ensemble table Objet et table Relation n'est rien d'autre que le modèle classique de la nomenclature de pièces en mécanique.

Reste à analyser deux choses :

- 1) peut on accéder facilement et intelligemment à toute donnée du modèle
- 2) ce modèle peut il enregistrer tout méta modèle tel qu'une fiche contact plus élaborée, un système de commandes comme imaginé plus haut, etc ...

1) **accès aux données** ; imaginons de récupérer toutes les données d'une fiche contact connaissant le n° de téléphone 0430234567. Pour ce faire :

Il faut rechercher 0430234567 dans la colonne Donnee de la table Objet ; on trouve la clé 10 et le type 4 (signifiant n° de téléphone).

On recherche ensuite 4 dans la colonne Origine de la table Relation ; on ne trouve rien, ce qui signifie que le numéro de téléphone est une donnée liée à une origine. Mieux, on peut imaginer que des recherches précédentes sur le méta modèle nous ont permis de savoir que 4 est une donnée liée.

On recherche donc 10 dans la colonne Destination de Relation ; on trouve que l'origine a pour clé 13. On recherche donc désormais 13 dans la colonne Origine de Relation et on trouve 3 occurrences de clés Id respectives 8, 9 et 10.

Pour chacune des clés, on recherche dans la table Objet :

2	Durant	(le 2 identifiant le type Nom)
3	Marcel	(le 3 identifiant le type Prénom)
4	0430234567	(le 4 identifiant le type Téléphone)

On est donc en mesure de répondre totalement à la question qu'on s'est posée.

Remarque importante : la structure de la table Objet permet d'y mémoriser toute arborescence. Par exemple si on définit une fiche contact qui comprend le nom (1fois), le prénom (2 fois), le téléphone (3 max), l'adresse (2 fois) elle même composée de ligne adresse (2 fois), code postal, ville on obtient le méta modèle suivant :

Table Objet

Id	Id_pere	Donnee	Répétition
1		<i>Racine des types</i>	

2	1	(Type) Fiche client	1
3	2	(Type) Nom	1
4	2	(Type) Prénom	2
5	2	(Type) Téléphone	3
6	2	(Type) Adresse	2
7	6	Type Ligne adresse	2
8	6	(Type) Code postal	1
9	6	(Type) Commune	1

Nous avons du ajouter une colonne à la table objet pour enregistrer le facteur de répétition. Pour prendre en compte le caractère obligatoire d'une donnée ou ses caractéristiques physiques (nombre, date, chaîne de caractères, ...) ainsi que sa taille mini et/ou maxi, nous avons besoin d'ajouter d'autres colonnes à la table Objet. Bien que ce ne soit pas développé dans cette présentation, ces compléments sont essentiels en pratique.

Table Relation

Id	Origine	Destination
1	2	3
2	2	4
3	2	5
4	2	6
5	6	7
6	6	8
7	6	9
...

De la même façon, les occurrences correspondants aux types s'inséreront dans la table Objet d'une part et leurs relations dans la table Relation.

Remarque importante : le couple (Id_pere, Donnee) de la table Objet est unique. Il ne peut donc pas exister deux occurrences du Nom Dupont. Mais rien n'empêche d'avoir plusieurs fiches comprenant le Nom Dupont car dans la table Relation, en colonne Destination figurera la clé du premier et seul Nom Dupont enregistré une seule fois dans Objet. De même on peut avoir un couple (2, Rose) et un autre couple (21, Rose) par exemple ; le premier est le Nom Rose, le deuxième est la couleur si 21 est l'Id du type Couleur.

2) Ce modèle peut il enregistrer tout autre méta modèle ?

Essai d'enregistrement du modèle de gestions des commandes dans ce modèle universel.

La table Client comprend les champs Nom, Prénom, Téléphone et Adresse. On voit que la table Client est équivalente à la fiche contact précédente.

La table Article comprend les champs Libellé, Prix, Quantité disponible ?

La table Commande comprend la référence au client et une Ligne commande (n fois) ainsi que le Montant total commandé.

La Ligne commande comprend la référence article, la quantité, le prix

Ce modèle génère donc le méta modèle suivant :

Le type Fiche déjà vu est renommé Fiche Client. Donc les champs Nom, Prénom Téléphone et Adresse sont déjà repris. On complète le méta modèle comme suit :

Table Objet (suite)

Id	Id_pere	Donnee	Répétition
...
10	1	(Type) Article	N fois
11	10	(Type) Libellé	1
12	10	(Type) Prix	1
13	10	(Type) Quantité disponible	1
14	1	(Type) Commande	N
15	14	Type référence Client	1
16	14	Type référence Ligne commande	N
17	1	(Type) Ligne commande	N
18	17	Type référence Commande	1
19	17	Type référence article	1
20	17	(Type) Quantité commandée	1
21	17	(Type) Prix payé pour cet article	1
22	14	(Type) Montant total commande	1

Les lignes de clés 15 et 16 ainsi que 18 et 19 sont grisées car sans doute inutiles.

On complète la table Relation comme suit :

Table Relation (suite)

Id	Origine	Destination
...
8	10 (type Article)	11(type Libellé)
9	10	12(type Prix)
10	10	13(type Qté Dispo)
	14(type Commande)	2(type Client)
	14	17(type Ligne commande)
	14	22(type total commandé)
	17(type Ligne commande)	2(type Commande)
	17	10(type Article)
	17	20(type Quantité)
	17	21(type Prix payé)

Les occurrences de ces types de données obéissent exactement au même schéma.

Nous venons donc de montrer que notre modèle universel accepte bien de mémoriser fidèlement le méta modèle de gestion des commandes ainsi que les données des occurrences Client, Commande, Ligne de commande et Article correspondantes.

Par contre nous n'avons pas montré comment s'effectueraient des traitements comme la décrémentation de la quantité disponible et le calcul du montant total de la commande ni même à quel endroit du modèle universel, nous pourrions mémoriser ces traitements ou méthodes.

Tel quel, le modèle universel sait stocker les informations ou les relire mais ne sait pas encore effectuer des traitements. De même, il serait souhaitable que les traitements les plus courants comme la création de fiches complètes et la recherche de toute information soient universelles.

Dernier exemple d'une comptabilité pour particulier. Une opération comptable se compose d'un Compte bancaire, d'une date, d'un montant, d'une imputation, d'une référence tiers, d'un type. L'imputation est par exemple, Alimentation, Habillement, Maison, ...

La référence est le commerçant ou l'employeur, ...

Le type est chèque, CB, prélèvement, versement, ...

Le modèle de données classique comprendrait les entités Operation, Compte, Imputation, Tiers, Type.

En modèle universel on obtient :

Table Objet

Id	Id_pere	Donnee	Répétition
1		<i>Racine des types</i>	
2	1	<i>(Type) Opération</i>	1
3	2	<i>(Type) Date</i>	1
4	2	<i>(Type) Montant</i>	1
5	1	<i>(Type) Compte</i>	1
6	5	<i>(Type) Banque</i>	1
7	5	<i>(Type) Numéro de compte</i>	1
8	1	<i>(Type) Imputation</i>	1
9	8	<i>(Type) Sous imputation</i>	1
10	9	<i>(Type) Libellé imputation</i>	1
11	1	<i>(Type) Tiers</i>	1
12	11	<i>(Type) Raison sociale tiers</i>	1
13	1	<i>(Type) mouvement</i>	1
14	13	<i>(Type) libellé mouvement</i>	1

Table Relation

Id	Origine	Destination
	2(<i>type Operation</i>)	3(<i>type date</i>)
	2	4(<i>type Montant</i>)
	2	5(<i>type Compte</i>)
	2	9(<i>type Imputation</i>)
	2	11(<i>type Tiers</i>)

	2	13(type Mouvement)
	5(type Compte)	6(type Banque)
	5	7(type N° compte)
	9	10
	11	12
	13	14

Reste à vérifier que l'on sait exploiter correctement et automatiquement les données du modèle universel sur les deux exemples précédents. En effet, non seulement le modèle universel doit contenir toutes sortes de données, mais il doit être écrit de telle manière que son code soit lui aussi universel. Ainsi, la seule description et l'enregistrement de tout méta modèle permettra immédiatement et sans recodage d'obtenir une application correspondante opérationnelle.

Les traitements universels

Puisque nous connaissons exactement la structure d'une fiche quelconque, il doit être possible de construire des traitements universels.

Ces traitements universels sont :

- La génération du formulaire de saisie adapté à la fiche
- La récupération des données saisies et la mise à jour automatique de la base universelle
- La recherche de toute donnée de la base.

La recherche de toute donnée dans la base permet également de retrouver les méta données ce qui peut surprendre l'utilisateur courant. Remarquons que la colonne répétition n'est utilisée que pour les méta données. En forçant 0 ou NUL dans cette colonne pour les occurrences, on dispose d'un moyen de cacher les métadonnées si nécessaire.

Génération automatique de formulaire

Donnée d'entrée : une méta fiche spécifiée par son Id ou par son nom qui doit être unique.

Sortie du générateur : un formulaire de saisie.

Table Objet (rappel)

Id	Id_pere	Donnee	Repetition
1		<i>Racine des types</i>	
2	1	<i>(Type) Fiche client</i>	1
3	2	<i>(Type) Nom</i>	1
4	2	<i>(Type) Prénom</i>	2
5	2	<i>(Type) Téléphone</i>	3
6	2	<i>(Type) Adresse</i>	2
7	6	<i>Type Ligne adresse</i>	2
8	6	<i>(Type) Code postal</i>	1

9	6	(Type) Commune	1

Analyse : La donnée d'entrée identifie une fiche et une seule. Soit Idfiche son Id. On recherche alors dans Objet, toutes les lignes qui ont Id_pere = IdFiche. Pour chacune de ces lignes on récupère donc dans le champ Donnee le label ou libellé du champ à saisir et dans le champ Repetition, le nombre de champs de saisie à afficher.

Par exemple label = Prénom qui sera suivi de 2 champs de saisie puisque 2 prénoms max attendus. Cas de l'adresse : L'adresse n'est pas une donnée élémentaire mais une regroupement des données élémentaires Ligne adresse, Code postal, Commune.

On peut déterminer qu'adresse est un composé parce qu'il existe des lignes Objet qui ont pour Id_pere, l'Id de Adresse. Et il est important, pour un composé comme Adresse de récupérer le facteur de répétition, ici 2. Cela signifie que le formulaire de saisie doit permettre la saisie de 2 blocs adresse, chacun de ces blocs comportant :

le label Ligne adresse suivi de 2 champs de saisie

le label Code postal suivi d'un champ de saisie

le label Commune suivi d'un champ de saisie.

En général, les champs de saisie doivent être nommés en vue de permettre leur récupération après soumission du formulaire. De même, il apparaît important d'associer à chaque champ de saisie du formulaire, l'Id du type de la donnée à saisir et sans doute l'Id_pere.

Compte tenu qu'il existe à priori des champs composé (comme Adresse), l'algorithme de génération du formulaire est sûrement récursif.

Algorithme proposé :

c = 0 ; # numéro de champ

afficheFichearbo(idFiche,c)

Recherche des fils de idFiche

Si existe au moins un fils

Pour chaque fils, récupère données de ce fils soit ;

Id, Id_pere, Repetition, Donnee

Si ce fils n'a pas de petit fils (cas Nom, Prénom, ...)

Affiche le label = Donnee

Exécute Repetition fois

Affichage du champ à saisir de nom : text.c, (text1, text2, ...)

Génère les champs pere.c = Id_pere et gpere.c = IdFiche en caché,

Incrémente c

fin Exécute Repetition fois

Si ce fils a t il des petits fils (cas Adresse)

Exécute Repetition fois

Génère un identifiant unique

Génère le champ contenant l'identifiant unique de nom text.c en caché

Génère les champs pere.c = Id_pere et gpere.c = en caché

afficheFichearbo(idAdresse,c) récursivité !

fin Exécute Repetition fois

Si aucun fils

Affiche erreur nom fiche

Terminé

La récupération des données

La récupération des données s'appuie sur le formulaire généré précédent dans lequel les champs saisis ont pour noms text0, text1, text2, ... , les types associés ont pour noms pere0, pere1, pere2, ... les grand pères associés ont pour noms gpere0, gpere1, gpere2, ...

Exemple de récupération possible :

text0 = Duchmol pere0 = 3 (type Nom) gpere0 = 2 (type Fiche contact/client)
text1 = Eddy pere1 = 4 (type Prénom) gpere1 = 2
text2 = Michel pere2 = 4 gpere2 = 2 (2ème prénom)
text3 = 0123456789 pere3 = 5 (type Téléphone) gpere2 = 3
text4 = 0623456789 pere4 = 5 gpere4 = 3 (2ème téléphone)
text5 = -non renseigné- pere5 = 5 gpere5 = 3 (pas de 3ème téléphone)
text6 = UUID pere6 = 6 (type Adresse) gpere6 = 2
text7 = 24 rue de la liberté pere7 = 7 (type Ligne adresse) gpere7 = 6 (type Adresse)
text8 = 75001 pere8 = 8 (type Code postal) gpere8 = 6
text9 = Paris pere9 = 9 (type Commune) gpere9 = 6
text10 = UUID pere10 = 6 gpere10 = 2 (2ème adresse n'existe pas)
text11 = -non renseigné- pere11 = 7 gpere11 = 6 (type Adresse)
text12 = -non renseigné- pere12 = 8 gpere11 = 6
text13 = -non renseigné- pere13 = 8 gpere11 = 6

Que faut il mémoriser dans Objet et dans Relation ?

D'abord une occurrence de Fiche contact dont le père est Fiche contact du méta modèle ; soit les champs Id_pere = 2, Donnee = UUID, Repetition = 0 ou NUL. Noter la clé Idf générée automatiquement par la base de donnée.

Ensuite créer une occurrence Objet Id_pere = pere0 = 3, Donnee = text0 = Duchmol, Repetition = 0 ou NUL. Noter la clé Idn générée

Il faut également lier cette occurrence Objet à sa fiche en créant une occurrence de Relation telle que :

Origine = Idf de l'occurrence fiche et Destination = Idn de l'occurrence nom.

Même traitement pour les prénoms, les numéros de téléphone, les adresses, ligne adresse, code postal et ville.

Si le champ de saisie n'est pas renseigné on ne mémorise rien, bien entendu. Dans le cas de l'adresse, il faut lire le textn suivant pour savoir si la ligne adresse a été renseigné ; astuce possible : analyser les données récupérées et supprimer les UUID des adresses non renseignées.

Attention, les ligne adresse, code postal et commune doivent être liés à Adresse et non à la fiche dans la table Relation.

Important : on ne peut pas créer de doublons (Id_pere, Donnee), donc il faut récupérer l'Id de l'original existant déjà et s'interdire de créer une nouvelle occurrence Objet si le cas arrive.

Algorithme proposé :

Analyser la liste des champs récupérés et mettre à -non renseigné- les UUID des Adresses n'ayant pas de ligne adresse.

MemoriseFiche

 Génère UUID

 Crée occurrence Objet avec Id_pere = gpere0 Donnee = UUID Repetition = 0 ou NUL

 Note Id de la fiche créée dans Idf

n = 0

memoriseFichearbo(Idf,n)

 Pour chaque triplet n de la liste de récupération

 Si textn est renseigné

 Si gperen = gpere0

 Recherche si occurrence Objet avec Id_pere = peren et Donnee = textn et Repetition = 0 ou NUL existe

Si existe, récupérer sa clé Idn
Si n'existe pas
Crée occurrence Objet avec Id_pere = peren, Donnee = textn, Repetition = 0 ou NUL
Note Id de l'occurrence créée dans Idn
Crée occurrence Relation avec Origine = Idf et Destination = Idn
Si gperen <> gpere0
memoriseFichearbo(Idn,n) #récursivité ! L'Id de l'adresse remplace l'Idf de la fiche
fin Pour chaque triplet n

La recherche de toute donnée

Cette recherche peut être soit typée soit non typée.

Recherche typée

Il faut générer un formulaire d'affichage et sélection qui propose la liste des méta fiches existantes. L'utilisateur sélectionne une fiche.

Il faut alors générer un formulaire de saisie qui propose la liste des types de données présentes dans la fiche. Ce formulaire est identique à celui vu plus haut sauf qu'il n'y a pas lieu de faire figurer les répétitions possibles.

L'utilisateur peut renseigner un ou plusieurs champs du formulaire. Il peut également utiliser des jokers ? Ou *

La recherche consiste alors à rechercher, pour chaque type et donnée renseigné, les occurrences Objet qui correspondent pour ce type.

Pour chaque type de donnée on crée une liste des clés des occurrence Objet trouvées. On en supprime les doublons éventuels.

On effectue ensuite l'intersection logique de ces listes pour obtenir la liste des clés retenues.

Recherche non typée

Dans le cas de la recherche non typée, on donne la possibilité à l'utilisateur de renseigner un seul champ avec ou sans jokers.

On recherche alors dans Objet toutes les occurrences qui correspondent sans tenir compte du type.

On élimine les doublons éventuels. On obtient la liste des clés retenues.

Affichage de la réponse (commun aux deux type de recherche)

Pour chacune de ces clés (de composants), on remonte jusqu'à sa fiche. On note la clé de la fiche dans liste fiche.

On élimine les doublons éventuels.

Pour chaque clé de fiche, on recherche les occurrences d'Objet qui en sont les composants et on affiche la réponse à l'utilisateur sous forme d'un tableau ou chaque ligne correspond à une fiche et chaque colonne à un type de données.

Remarque : la création d'une liste de contacts sous la forme universelle ci-dessus est très intéressante à plus d'un titre.

Il devient possible d'interroger sur tout type de donnée.

Les données étant uniques, on peut, par exemple associer à une adresse, ses coordonnées géodésiques ; il devient alors simple de trouver tous les contacts à proximité de.

Les traitements spécifiques ou additionnels

Dans le cas de l'application de gestion des commandes nous avons vu la nécessité de calculer le montant total de la facture ou encore de décrémenter la quantité d'articles en stock. Dans le cas de l'application de comptabilité particulier nous avons besoin d'afficher le bilan par compte, par imputation, etc ... Les traitements universels d'enregistrement de fiches, de modification de fiches (non analysé ici), de recherche ne suffisent pas. Mais il serait également dommage que les traitements spécifiques n'utilisent pas à bon escient le modèle universel.

Idée proposée :

Normaliser les traitements spécifiques un peu comme des méthodes de classes.

Etablir un catalogue de ces traitements spécifiques normalisés avec leur noms, paramètres d'appel et retours attendus.

Enregistrer quelque part dans le modèle universel, le nom, les paramètres d'appel et le retour attendu d'un traitement.

Catalogue de traitements/fonctions (amorce de catalogue)

Cumul de montant sur n occurrences d'une fiche

Semble applicable aux opérations comptables avec montant = montant de l'opération et critère de sélection = compte ou imputation donné.

Semble applicable aux lignes de commande avec montant = prix payé et critère de sélection = lignes d'une commande donnée

Produit de 2 champs numériques

Semble applicable au calcul du prix payé en fonction de la quantité commandée et du prix de l'article.

Incrément/décrément d'un champ

Semble applicable à la quantité disponible d'un article.

Affichage liste pour sélection

Semble applicable à de nombreux cas comme choix d'un article commande, choix d'un compte, d'une imputation ...

Problème

Comment paramétrer universellement ces traitements ou fonctions ?

Quel est l'évènement déclencheur du traitement ?

Cumul de montant de n occurrences d'une fiche (ou tête de liste)

Il s'agit d'une fonction

Entrées : Id donné d'un père (par exemple, Id de la tête de liste des opérations comptables)

Id d'un critère de sélection dans la liste.

Id du type de champ à cumuler

Exemple de contenu des tables Objet et Relation pour le cumul d'opérations comptables pour une imputation donnée

table Objet

Id	Id_pere	Donnee
100	1	(Type) Imputation
101	100	Alimentation
102	100	Transport
103	100	Salaire
...		

120	1	(type) Operation
121	120	UUID
122	120	UUID
123	120	UUID
...		
130	1	(type) Date
131	130	04/07/2015
132	130	04/08/2015
133	130	07/08/2015
...		
140	1	(type) Montant
141	140	+3780
142	140	-870
...

Table Relation

Id	Origine	Destination
	121(opération 1)	131(date)
	121	141(Montant de 3780)
	121	103(occ . Imputation Salaire)
	122(opération 2)	133(date)
	122	142(Montant de -870)
	122	101(occ. Imputation Alimentation)
	123(opération 3)	132(date)
	123	141(Montant de 3780)
	123	103(occ . Imputation Salaire)
...

Pour l'imputation salaire le cumul sera de 3780×2 .

Dans l'exemple ci-dessous pour Id donné = 120 on trouve les Id d'opérations 121, 122, 123

On trouve, respectivement pour chaque occurrence fils, l'Id du critère est 103 qui correspond au salaire et l'Id du montant à cumuler est 141

l'Id du critère est 101 qui correspond au salaire et l'Id du montant à cumuler est 142

l'Id du critère est 103 qui correspond au salaire et l'Id du montant à cumuler est 141

103 est l'Id du critère donné, donc des occurrences à garder

140 est l'Id du type Montant qui a pour fils 141 et 142 (Remarque il n'y a pas de 3ème occurrence montant car le montant des opérations 121 et 123 est le même (salaire inchangé))

On cumule donc ici, 2 fois le montant 3780 qui a pour clé 141.

Algorithme de traitement proposé :

Initialise cumul à 0

Lire dans Objet toutes les occurrence du type de champ à cumuler et créer une tableChamp de couples (Id, Donnee) → 141,3780 et 142,-870

Lire dans Objet les occurrences du type père donné → Liste Id : 121, 122, 123

Pour chaque occurrence trouvée lire occ. Relation telle que Origine = Id et Destination = Id critère → trouve 121, 103 et 123, 103

Pour chaque occ. Relation trouvée, note Origine dans tableOrigine → trouve 121 et 123

Pour chaque élément de tableOrigine effectue recherche dans Table Relation avec Origine = élément de tableOrigine et Destination= l'un des éléments Id de tableChamp ; si trouvé cumule élément Donnee correspondant de tableChamp.

Retourne ce cumul

Remarque : cet algorithme n'est pas optimal ; à améliorer ; peut être faut il créer des relations supplémentaires dans la table Relation pour accélérer le traitement, par exemple relations inverses Destination → Origine.

Affichage pour sélection

Il s'agit d'un formulaire générique

Entrée : Id d'un père (par exemple, Id de la tête de liste des imputations)

Affichage : Liste des imputations

Action utilisateur ; choix d'une occurrence de la liste

Traitement : retourne l'Id de l'occurrence choisie.

Algorithme proposé :

Dans Objet lire les occurrences telles que Id_pere = Id d'un père

Pour chaque occ. trouvée afficher dans un formulaire le champ Donnee et garnir champ caché Id de cette occurrence

Lorsque l'utilisateur sélectionne une occurrence, retourner l'Id caché correspondant.

Points divers, améliorations

Typage physique et autres caractéristiques méta

Le modèle universel tel que montré jusqu'ici, ne possède pas de typage physique des données. Le champ Donnee de l'entité Objet peut recevoir de la chaîne de caractères de longueur quelconque représentant des libellés, des dates, des nombres, etc ... Pour mieux guider la saisie, pour éviter des erreurs d'interprétation des données, mettre en place un typage des données est sans doute très utile. Ce typage pourrait consister dans l'ajout d'une colonne sur la table Objet. Mais cette solution reviendrait à ajouter une colonne totalement inutile pour les occurrences objets autre que celles du méta modèle.

Par rapport aux occurrences d'objet, les types d'objet ont de nombreuses propriétés qu'il faut mémoriser comme facteur de répétition, obligation de saisie, type physique de la donnée à mémoriser, etc ...

Il semble donc judicieux d'éclater la table Objet en deux parties :

- une partie réservée au méta modèle ayant les colonnes Id, Id_pere, Donnee, Repetition, Obligation, Typage, ...
- une partie réservée au modèle ayant les seules colonnes Id, Id_pere, Donnee.

Cette séparation simplifiera également la recherche dans les occurrences puisqu'on ne trouvera plus de mélange entre meta modèle et modèle dans la même table Objet. La table du méta modèle aura pour nom Objet_meta. La table des occurrences aura pour nom Objet.

Images, Blobs

Pour que le modèle universel puisse enregistrer des objets volumineux comme des images ou des blobs, il faut que le champ Donnee le permette ; il faut également éviter le gâchis de mémoire en base de données. Il faut donc une gestion optimale des champs longs genre blobs par la base de données.

Autre possibilité : mettre dans Donnee le chemin d'accès à un fichier image ou autre.

- semble plus économe d'espace mémoire
- mais fige l'implantation physique des fichiers correspondants.

La séparation en partie méta modèle et partie occurrences du modèle imaginée dans le paragraphe précédent permettra sans doute de bien stocker les blobs dans Objet.

Performance

La table Objet comportera à terme un très grand nombre d'occurrences puisque chaque objet unique constitue une occurrence de la table. Par contre on n'y trouvera aucun doublon. La table Relation comportera un très grand nombre d'occurrences puisque c'est elle qui mémorise les relations entre objets et évite tout doublon sur les objets. En termes de performance, il faut regarder au premier chef, la performance d'accès en lecture et recherche. En effet, lors de la création et enregistrement de données par un utilisateur, la machine dispose de beaucoup temps par rapport à la célérité de l'utilisateur.

Il est essentiel d'avoir un index accélérateur de recherche sur les champ Donnee et Id_pere (type de donnée).

Estimation du temps d'accès en lecture ;

Revoir le paragraphe Recherche de donnée. Cas de la recherche non typée. Quand l'utilisateur appuie sur le bouton Rechercher, il déclenche une recherche du motif de recherche sur la table Objet et sur le seul champ Donnee. La recherche est simple et peu être pré-compilée en base de données. Donc, à priori cette recherche est rapide même avec un index important.

La remontée sur la fiche à l'origine des données utilise Id_pere également indexé.

La visualisation des types composants de la fiche implique :

De lire les occurrences de Relation pour une Origine donnée (l'Id de la fiche)

De remonter sur chaque occurrence de Objet ayant pour clé Id, la Destination associée à cette Origine.

Pour une fiche comprenant 10 types de donnée, nous obtenons :

10 accès à Relation par Origine donnée, 10 accès à Objet par Id donné (clé primaire)

Soit un total de 21 accès en plus de la recherche initiale.

En supposant une durée moyenne d'accès de 10ms on obtient un temps de réponse légèrement supérieur à 0,02 seconde.

Gestion automatisée du méta modèle

Nous avons vu que le le modèle universel proposé permet de pendre en charge tous les modèles possibles.

Reste à automatiser la saisie d'un méta modèle. On commence par la création d'une simple fiche avec ses données.

Génération d'une fiche du méta modèle

Scénario :

L'utilisateur est invité à entrer un nom pour la nouvelle fiche. L'unicité de ce nom est contrôlée par rapport à celles qui existent déjà. Le facteur de répétition de la fiche initiale est initialisé à 1.

L'utilisateur peut alors saisir le nom d'une donnée nouvelle ou choisir un nom de donnée (ou fiche) existant déjà. Il doit préciser le facteur de répétition souhaité.

L'utilisateur voit apparaître un tableau avec une première ligne fiche initiale non indentée.

Chaque donnée ou fiche rattachée apparaît en dessous indentée et dans l'ordre de saisie.

Si une fiche est rattachée, ses données apparaissent en dessous elles mêmes indentées.

Traitement réalisé

Création d'une ligne Objet pour la méta fiche nouvelle avec Id_pere = Id des types, Donnee = nom de la fiche, Répétition = 1 et mémorisation de l'Id de cette fiche dans Idfiche.

Pour chaque donnée nouvelle créée, création d'une ligne Objet avec Id_pere = Id de la racine des types, Donnee = nom de la donnée, Repetition = facteur de répétition saisi ; mémoriser l'Id de la donnée créée. Si on choisit une donnée existant déjà, récupérer son Id.

De plus on crée la ligne Relation avec Origine = Idfiche et Destination = Id de la donnée créée ou choisie. Si l'hypothèse de la relation inverse est retenue, créer de même cette ligne dans la table Relation.

Algorithme proposé :

Modification d'une fiche du méta modèle

Analyse :

On peut changer le facteur de répétition des données d'une fiche ou encore ajouter une nouvelle donnée ou supprimer une donnée d'une fiche.

Quelles sont les incidences de ces modifications sur les données déjà enregistrées ?

Changement du facteur de répétition :

Lors de la saisie de données d'un type donné, un champ supplémentaire (ou plus peut apparaître) ; par exemple on décide d'enregistrer non plus deux mais trois prénoms au maximum. Lors de la recherche ou consultation, on suit les liens créés. Donc, les saisies déjà enregistrées apparaitront avec un ou 2 prénoms maxi, tandis que les saisies nouvelles apparaitront avec un à trois prénoms maxi.

On peut considérer que le facteur de répétition n'a pas d'incidence sur les données déjà enregistrées.

Ajout d'un type de donnée nouveau :

Lors de la saisie d'une fiche, un champ nouveau répété n fois selon son facteur de répétition apparaîtra. Par exemple ajout du mail sur la fiche contact.

Si ce champ nouveau est obligatoire, la fiche de saisie sera erronée pour les fiches déjà enregistrées.

Et il n'est pas imaginable de renseigner automatiquement un champ nouveau, inconnu jusqu'à

présent. Il faut donc procéder à la mise à jour de toutes les occurrences de fiches anciennes relative à la fiche du méta modèle modifiée.

Si ce champ est facultatif, pas de problème.

Suppression d'un type de donnée :

Lors de la saisie d'une fiche, un champ n'apparaîtra plus ; il ne sera plus à saisir. Par exemple, on suppose que la ligne adresse du bloc adresse ne sert à rien. Pour les nouvelles fiches saisies pas de problème. Pour la consultation, la recherche sur le critère ligne adresse ne sera plus possible, mais les données ligne adresse anciennes restent présentes dans le modèle universel bien que devenues inutiles. Lors d'une consultation, en suivant les liens de Relation inchangés, on pourra faire apparaître les données ligne adresse mais on sera incapables de préciser leur label de champ puisque celui-ci n'existe plus dans la méta fiche !

1) il faut modifier l'algorithme d'affichage des données afin qu'il ne plante pas s'il ne trouve pas le label

2) mieux il faut cacher (ne pas afficher) les données correspondantes

3) plus fort encore, en profiter pour supprimer celles-ci dans Objet et les liens correspondants dans Relation afin de faire le ménage au fur et à mesure des consultations.

4) associer à la suppression de type de donnée, un traitement de fond faisant ce ménage, sans doute en différé pour éviter l'attente de l'utilisateur.

Fusion de deux types de données :

Supposons que l'on décide de remplacer les deux types Nom et Prénom d'une fiche contact par un seul type PrénomNom. Pour chaque fiche contact, il faudra alors récupérer le nom et le prénom puis construire une chaîne prénom suivie du nom (ou l'inverse, et/ou avec séparateur entre les deux, ...); cette construction deviendra une occurrence du type PrénomNom liée correctement à la fiche par les liens adhoc dans Relation. Il faudra également supprimer toutes les occurrences de Nom et de Prénom dans Objet, devenues obsolètes. Même scénario au niveau de la méta fiche mais les choses sont plus simples parce qu'effectuées une seule fois mais au préalable puisqu'on a besoin de l'Id du nouveau type pour définir le père (Id_pere) des occurrences.

A priori, la fusion de types est donc un traitement de fond. L'administrateur du modèle universel prépare la fusion qui est mise en attente d'exécution la nuit par exemple.

Eclatement d'un type en deux sous types (ou plus) :

Par exemple, supposons que l'on décide de séparer la ligne adresse en 2 parties, le numéro et le nom de rue d'une part, les infos complémentaires bâtiment, étage, ... d'autre part. D'abord il faut déterminer les règles qui permettent de faire de découpage de données de façon automatique. Ensuite pour chaque occurrence de fiche, on récupère deux données élémentaires donc deux occurrences à créer dans Objet ayant pour père/type l'Id de chacun des nouveaux types créés. Il faut bien entendu créer les liens correspondants dans Relation. Et il faut au final supprimer les occurrences Objet de l'ancien type devenu obsolète et supprimer les liens de Relation correspondants sans objet.

A priori, l'éclatement de types est donc un traitement de fond. L'administrateur du modèle universel prépare l'éclatement qui est mis en attente d'exécution la nuit par exemple. En particulier, l'administrateur devrait disposer d'une option lançant l'éclatement à blanc (sans mises à jour) mais de nature à détecter les cas où la règle de découpage ne marche pas bien.

