

La persistance des objets

Philippe PRADOS

pp@philippe.prados.name



*Préservez l'environnement,
n'imprimez pas ce document*

TABLE DES MATIERES

| | |
|---|---|
| Différence entre le modèle relationnel et le modèle objet | 3 |
| Modèle relationnel | 3 |
| Modèle objet | 3 |
| Traduction du modèle objet en modèle relationnel..... | 4 |
| Traduction de l'identifiant | 4 |
| Traduction des agrégations..... | 4 |
| Traduction des relations | 5 |
| Traduction de l'héritage..... | 5 |
| Traduction du polymorphisme..... | 5 |
| Solutions réalistes..... | 6 |
| Ne pas utiliser le modèle objet | 6 |
| Utiliser une base de donnée objet..... | 6 |
| Utiliser un Framework de persistance..... | 6 |
| Conclusion | 6 |

Avant propos

Ce document se propose de regarder les différentes solutions pour sauver un modèle objet dans une base de données relationnel. Il explique les différences entre le modèle relationnel et le modèle objet et propose des solutions pour unifier ces deux mondes.

1. DIFFERENCE ENTRE LE MODELE RELATIONNEL ET LE MODELE OBJET

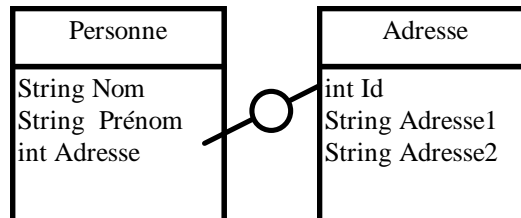
Le modèle objet est un modèle de donné beaucoup plus riche que le modèle relationnel. La manipulation de ce modèle est complètement différente de la manipulation d'un modèle objet.

1.1 Modèle relationnel

Le modèle relationnel permet de construire des tables d'enregistrement. Une table est constituée de colonne. Chacune possède un type qui la caractérise. Les types des colonnes sont finis (entier, date, caractères...) Il n'est pas possible de demander la création d'une colonne avec un type particulier n'étant pas connu de la base de donnée. Dans cette table, chaque ligne représente un enregistrement. Tous les enregistrements d'une table ont les mêmes colonnes. Deux lignes ayant strictement les mêmes valeurs dans chacune des colonnes ne sont pas différenciables.

Pour sélectionner des enregistrements, il faut indiquer des critères de sélection sur les valeurs des colonnes. Cela permet de sélectionner un sous-ensemble dans une table.

Le modèle relationnel permet également d'effectuer des relations entre les colonnes de plusieurs tables. La base de donnée sélectionne alors les enregistrements permettant une unification entre les valeurs des colonnes. Par exemple, prenons une table `Personne`. Celle-ci possède les colonnes nécessaires pour mémoriser le nom et le prénom d'un individu. Si nous autorisons plusieurs personnes à habiter à la même adresse (même famille), il est intéressant de factoriser les informations concernant l'adresse pour éviter de dupliquer ces informations dans plusieurs lignes de la table `Personne`. Nous créons une table pour mémoriser une adresse. Il faut ajouter une colonne dans la table `Personne` pour référencer les adresses. Il faut également ajouter un identifiant dans la table `Adresse`.



Cela permet de mettre en relation la table `Personne` et la table `Adresse` par l'intermédiaire de l'identifiant de l'adresse. Pour obtenir toutes les informations sur les `Personnes`, il faut utiliser une requête :

```
select * from Personne,Adresse
where Personne.Adresse=Adresse.Id
```

Les champs des deux tables sont retournés.

Pour utiliser le modèle relationnel, il faut associer plusieurs tables. Cela permet de retourner un ensemble d'enregistrement. On ne manipule que des ensembles. Si l'on demande la récupération d'un individu particulier, le système retournera un ensemble composé d'un seul enregistrement.

Le modèle relationnel est caractérisé par :

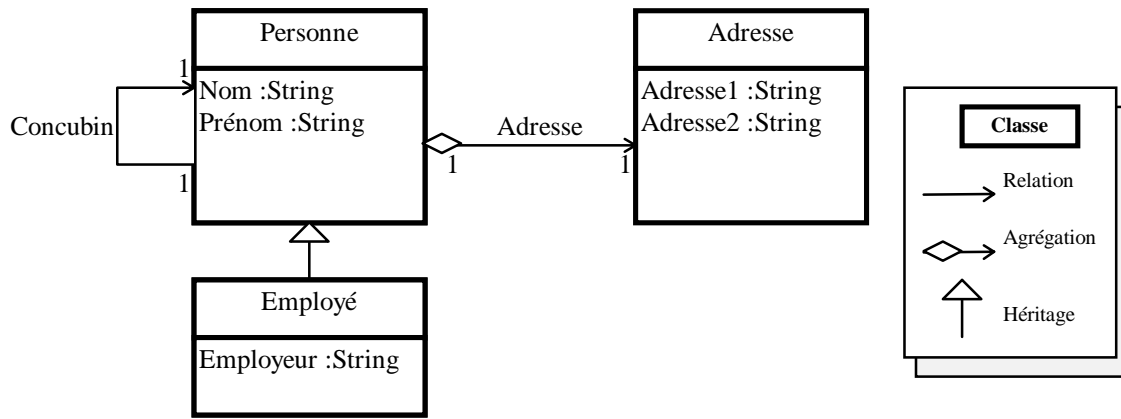
- Des ensembles d'enregistrement monomorphiques
- L'absence d'identification des enregistrements.
- Des requêtes ensemblistes retournant des sous-ensembles.

1.2 Modèle objet

Le modèle objet permet de construire des enregistrements (ou objet) en héritant des comportements et des attributs d'autres objets. On construit des classes. Celles-ci regroupent un ensemble d'attribut et de méthode. Une classe peut hériter d'une autre classe. Cela veut dire qu'elle bénéficie de toutes les méthodes et de tous les attributs de la classe héritée. Elle peut ainsi ajouter de nouvelles méthodes et de nouveaux attributs.

Les attributs d'un objet peuvent également être des objets. Une `Personne` peut posséder un attribut `Adresse`, lui-même composé de différents attributs. Cela s'appelle l'*agrégation*.

Prenons une classe `Personne`. Celle-ci possède différents attributs. Créons une classe `Employé` héritant de `Personne`.



Une *Personne* peut posséder une relation de concubinage avec une autre *Personne*.

Tous les traitements concernant les *Personnes* peuvent être appliqués aux *Employés*. Par contre, les traitements spécifiques aux employés ne peuvent pas s'appliquer aux personnes.

Il n'y a pas de notion de table (ensemble de tous les objets du même type) Deux objets ayant strictement les mêmes valeurs ne sont pas considérées comme identiques. Pour demander un traitement, il faut identifier précisément l'instance devant l'exécuter.

Pour utiliser le modèle objet, on navigue d'un objet à un autre. Connaissant un objet *Personne* particulier, on utilise un de ces services pour obtenir l'objet *Adresse* correspondant. De proche en proche, on parcourt tous les objets de l'application.

Le modèle objet se caractérise par :

- Un identifiant unique pour chaque objet
- Un héritage des objets
- Des objets polymorphiques
- Une agrégation d'objets
- Des relations entre les objets

2. TRADUCTION DU MODELE OBJET EN MODELE RELATIONNEL

Pour sauver un modèle objet dans un modèle relationnel, il faut utiliser des règles de traduction. Celles proposées ci-dessous ne sont pas forcément les plus efficaces. Elles permettent de présenter les difficultés de traductions. Pour plus d'information, reportez-vous à la présentation « La persistance des objets - Modèle objet et modèle relationnel », Ph. Prados, juillet 96.

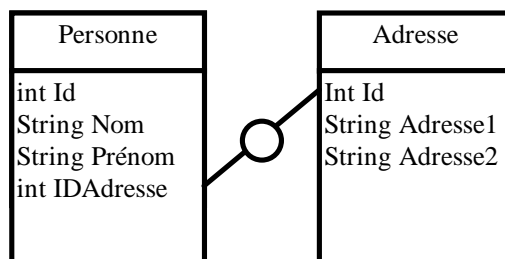
2.1 Traduction de l'identifiant

Comme nous l'avons vu précédemment, le modèle objet demande que chaque enregistrement soit identifié de façon unique. Il faut alors créer une colonne spéciale pour y placer une valeur unique qui caractérisera l'objet. Cet identifiant sera utilisé par la suite pour sélectionner les enregistrements lors de requêtes ensemblistes. Il faut alors un générateur d'identifiant. Certaines bases de données proposent un type particulier pour cela. Ce n'est pas suffisant. Il faut un identifiant unique pour toutes les tables. Il est alors possible d'ajouter à l'identifiant généré par la table, le nom de la classe de l'objet.

2.2 Traduction des agrégations

Un objet possède un ensemble d'attribut. Certains sont compatibles avec les types offerts par le modèle relationnel. Une colonne sera créée pour chaque attribut compatible (entier, flottant, chaîne de caractères, ...)

Pour les objets plus complexes, possédés par un autre objet, il va falloir utiliser l'identifiant de l'objet possédés. Une colonne sera créée pour chacun de ces objets. Celles-ci posséderont les identifiants des objets contenus.



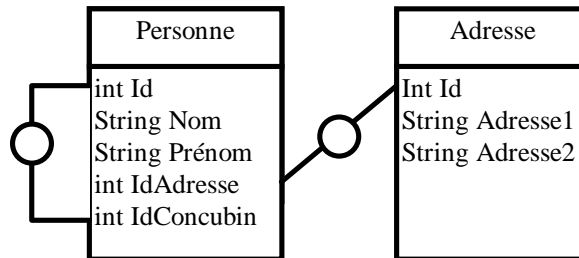
Pour lire un objet, il faut utiliser une requête ensembliste avec jointure. Cette requête retourne un ensemble composé de tous les attributs de l'objet.

```
Select * from Personne,Adresse
where Personne.Id=X
and Personne.IdAdresse=Adresse.Id
```

Ce type de requête devra être enrichi pour tous les objets agrégés. Sur ce concept de base, il faudra ajouter la prise en compte du polymorphisme (voir plus bas).

2.3 Traduction des relations

Pour traduire les relations, cela s'apparente à l'agrégation.

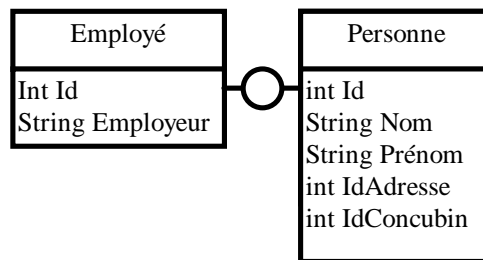


Pour sélectionner le concubin d'une personne, il faut utiliser une requête comme ceci :

```
select P1.* from Personne=P1,Personne=P2
where P1.IdConcubin=P2.Id
```

2.4 Traduction de l'héritage

Chaque instance étant unique, l'identification d'un **Employé** peut servir d'identification d'une **Personne**. La même valeur d'identifiant sera alors utilisée.



Cela permet de rédiger une requête ensembliste pour récupérer toutes les informations d'un employé.

```
Select * from Employé, Personne
where Employé.Id=Personne.Id
```

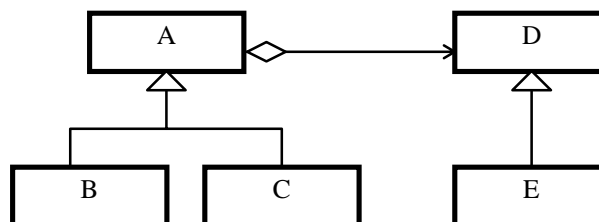
2.5 Traduction du polymorphisme

Si un objet possède une relation vers un objet **Personne**, il ne sait pas si cette personne est un **Employé** ou non. Il doit alors utiliser plusieurs requêtes ensemblistes pour toutes les combinaisons possibles afin de trouver le bon type de personne.

```
select * from Objet,Employé,Personne
where Objet.rela=Employé.Id
and Objet.rela=Personne.Id
select * from Objet,Personne
where Objet.rela=Personne.Id
```

Cela s'appelle un **select** polymorphe. Pour sélectionner un objet, il faut multiplier les requêtes pour toutes les branches de l'héritage afin de sélectionner tous les objets possibles. Si ces objets possèdent des agrégations, il faut alors procéder de mêmes pour l'arbre d'héritage des objets agrégés.

Pour l'arbre d'héritage suivant :



Une demande de tous les objets de type **A** se traduit comme ceci¹ :

```
select * from A,E,D
where A.Agr=E.Id and E.Id=D.Id
select * from A,D
where A.Agr=D.Id
select * from B,A,E,D
where A.Agr=E.Id and E.Id=D.Id
and B.Id=A.Id
select * from B,A,D
where A.Agr=D.Id
and B.Id=A.Id
select * from C,A,E,D
where A.Agr=E.Id and E.Id=D.Id
and C.Id=A.Id
select * from C,A,D
where A.Agr=D.Id
and C.Id=A.Id
```

Lorsque l'arbre d'héritage est plus important, cela devient extrêmement prohibitif d'exécuter ce type d'instruction. De plus, les risques d'erreurs sont innombrables. Une petite modification dans l'arbre d'héritage entraîne la correction de toutes ces requêtes.

3. SOLUTIONS REALISTES

Comme on peut le constater, le mariage entre le modèle objet et le modèle relationnel n'est pas simple. Il existe trois approches antagonistes pour offrir la persistance des objets.

3.1 Ne pas utiliser le modèle objet

Au vu de la complexité décrite ci-dessus, il semble raisonnable de simplifier le modèle de donnée. Il faut alors concevoir le modèle suivant une approche relationnelle classique, et rédiger des objets encapsulant le modèle relationnel. Ces objets ne bénéficieront pas de l'héritage. Ils n'offriront qu'une encapsulation des données. C'est le seul moyen raisonnable d'utiliser une base de données relationnelle pour sauver les données du modèle objet. Il ne faut plus utiliser de vrais objets. Le modèle objet pourra être utilisé pour tous ce qui n'a pas besoin d'être persistant.

3.2 Utiliser une base de donnée objet

Devant l'incompatibilité des deux modèles de données, des entreprises ont entrepris de rédiger des bases de données spécialisées dans le modèle objet. Celui-ci étant plus riche que le modèle relationnel, ils ont ajouté des interfaces permettant de consulter la base objet à l'aide d'outils relationnel classique. Des interfaces SQL permettent de consulter ces bases. Bien entendu, elles n'offrent pas les mêmes performances que les bases relationnelles classiques si on les utilise avec une approche ensembliste. Ils ne sont pas faits pour cela. Par contre, leurs performances lors d'une navigation entre objets sont tout à fait acceptables, et beaucoup plus rapide que l'approche ci-dessus pour le même usage. Les produits du marché sont aujourd'hui matures. Ils offrent des accès transactionnels, des sauvegardes en ligne, des capacités énormes, des accès multi base, etc.

3.3 Utiliser un Framework de persistance

Des outils du commerce se proposent de faciliter la traduction du modèle objet vers le modèle relationnel. Ces outils sont généralement très structurants. L'utilisateur doit utiliser des interfaces utilisateurs spécifiques pour décrire son modèle objet. Les sources correspondants sont alors générées. Certains imposent leur propre philosophie du modèle objet qui n'est pas toujours en conformité avec le langage utilisé.

4. CONCLUSION

Le modèle objet est un modèle beaucoup plus complexe que le modèle relationnel. Sauver des objets dans une base relationnelle c'est comme si, lorsque vous rentrez chez vous, vous démontez votre voiture pour la ranger dans votre garage. Le matin vous devez la remonter avant de l'utiliser.

Le problème de la persistance ne doit pas être négligé lors d'un développement avec un langage objet. Auparavant, le modèle relationnel était l'élément structurant des développements. La représentation interne des données dans le langage utilisé était différente de la représentation offerte par les APIs du modèle relationnel. Les traitements étant séparés des données, une simple traduction du modèle de donnée permettait l'utilisation de ce modèle. Les développements étaient très dépendants de l'utilisation de la base de donnée.

Le paradigme objet propose de réunir les données et les traitements s'y rattachant. Il est de ce fait, très structurant lui aussi. Il n'est pas possible de rédiger un programme en utilisant deux éléments structurant incompatible en eux. Les difficultés viennent de là. Il faut que l'un des deux modèles fasse des concessions.

Pour bénéficier des avantages du modèle objet, il faut sacrifier le modèle relationnel. Il est à noter que le modèle relationnel peut être encapsulé par un modèle objet. Il n'y a pas de perte de l'existant. Pour bénéficier des infrastructures existantes des bases de données relationnelles, il faut sacrifier le modèle objet.

¹ Cette syntaxe simplifiée ne tient pas compte de la détection des doublons.

INDEX

A

agrégation 4

H

héritage 5

I

identifiant 4

M

modèle
 objet 3
 relationnel 3

O

objet
 modèle 3

P

polymorphisme 5

R

relation 5
relationnel
 modèle 3

S

select polymorphe 5