**Le modèle CRUD ou CRUDE (Create Read Update Delete ou DElete)**

Le modèle CRUD est un modèle classique de la gestion de données. Il correspond à la mise en place des manoeuvres élémentaires et indispensables de tout développement informatique gérant des entités de données. Comme il n'existe pas un seul programme qui n'ait à gérer à un moment ou à un autre, des entités de données, on peut dire que cette compétence est FONDAMENTALE en informatique.

## Le modèle CRUD

1. Toute application est amenée à gérer des entités de données.   
  
2. Les entités de données sont des aggrégations d'information qui décrivent une entité de la réalité. Ces entités sont appelées structures (C), objets (C++ ou Java), enregistrements (SGBD).   
  
3. Toute structure n'est jamais gérée comme un objet unique. N'importe quelle application, une fois qu'elle gère un type d'entité, en gère en général plusieurs. Toute entité se gère donc par l'intermédiaire d'une collection.   
  
4. Une collection est un container qui gère plusieurs exemplaires différents d'une entité. Une collection ordonnée s'appelle une liste.   
  
5. Les quatre manoeuvres indispensables pour gérer une collection d'entités sont :

* créer une entité et l'ajouter à la collection. (C)
* afficher la collection exitante. (R)
* modifier une entité dans la collection. (U)
* supprimer une entité de la collection. (D)

Trois de ces manoeuvres sont indispensables pour pouvoir mettre la collection dans TOUS ses états possibles : collection vide, un élément, deux élements etc. Il s'agit de CR et D. U peut être obtenue par une combinaison de R -> D -> C, c'est-à-dire que *"j'extrais l'objet que je veux modifier, je le modifie à l'extérieur de la collection, je détruis son ancienne version et j'ajoute une nouvelle entité modifiée"*. Cependant, qu'elle soit simulée, ou qu'elle soit implémentée sans détruire l'entité dans la collection, l'effet final est le même : la collection contient le même nombre d'éléments qu'avant la modification, dont un a changé de contenu.

## Le modèle CRUD et le Web

Dans le contexte Internet, les quatre opérations du modèle CRUD ont une traduction sous forme d'interface utilisateur :

### Read

Traduit la lecture de la collection. Le résultat est un écran de liste, qui peut être affiché sous forme :

* d'une table d'enregistrements (un enregistrement correspond à une ligne ou à un [bloc](http://www.ethnoinformatique.fr/mod/glossary/showentry.php?courseid=87&concept=Bloc) de lignes d'afichage de la table. (si l'information à afficher est composée).
* d'une liste déroulante (si l'information à afficher est unique)

Exemple :   
  
Soit la gestion d'utilisateurs d'un système sous forme d'entités définies par : 

|  |
| --- |
| class User{  var $login;  var $password;  var $firstname;  var $lastname; var $description; var $role; varf $enabled; } |

La liste peut s'exprimer sous forme tabulée : 

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| vf | xxxxxx | Fremaux | Valery | Administrateur | ADMIN | Y |
| hdm | yyyyyy | de Milleville | Hervé | Responsable du département | MEMBER | Y |
| rc | zzzzzz | Chelouah | Rachid | Enseignant | MEMBER | Y |
| bg | wwww | Glonneau | Bernard | Enseignant | MEMBER | Y |

### Create

La création d'une instance de cette entité de donnée nécessite une saisie de données, usuellement effectuée via un formulaire de saisie. Il est classique de présenter un formulaire sous forme d'un tableau à deux colonnes [label,saisie]

|  |  |
| --- | --- |
| **Login** | <input type="TEXT" ... > |
| **Password** | <input type="PASSWORD" ... > |
| **Nom** | <input type="TEXT" ... > |
| **Prénom** | <input type="TEXT" ... > |
| **Description** | <textarea></textarea> |
| **Role** | <select ... >    <option></option> </select> |
| **Actif** | <input type="RADIO" ... value="Y" > Oui <input type="RADIO" ... value="N" > Non |

Toutes les valeurs de formulaire peuvent être fixées à des valeurs par défaut SAUF le champ de type PASSWORD pour des raisons de sécurité.

L'ensemble de ces champs de saisie sont placées dans un élément de formulaire :

|  |
| --- |
| <form name="user" method="get" action="users.php"> ... </form> |

### Update

La mise à jour de données nécessite également la saisie des mêmes données (dans la plupart des cas, parfois de données complémentaires). Dans ce cas, la quasi totalité du formulaire de saisie d'ajout d'instance est identique. Les seules différences sont :

* Il faut recharger tous les champs avec les données existantes.

Recharger les champs nécessite plusieurs techniques, en supposant les valeurs existantes dans un objet opportuniste $aUser.

* + Input TEXT :

|  |
| --- |
| <input type="TEXT" name="parmname" value="<?php echo $user->parmname ?>"> |

* + TEXTAREA :

|  |
| --- |
| <textarea name="parmname"><?php echo $user->parmname ?></textarea> |

* + SELECT

|  |
| --- |
| <select name="parmname"> <?php foreach($options as $opt) { ?>  <option value="<?php echo $opt->values ?>"   <?php if ($opt->values == $user->parmname) echo "selected" ?> >  ...  </option> <?php } ?> </select> |

* + input RADIO

|  |
| --- |
| <input type="RADIO" name="parmname" value="Y"   <?php if ($user->parmname == 'Y') echo "checked" ?> />  <input type="RADIO" name="parmname" value="N"  <?php if ($user->parmname == 'N') echo "checked" ?> /> |

* + input CHECKBOX

|  |
| --- |
| <input type="CHECKBOX" name="parmname" value="Y"   <?php if ($user->parmname == 'Y') echo "checked" ?> /> |

* Il faut rappeler l'identifiant unique de l'objet modifié.

|  |
| --- |
| <form name="user" method="get" action="users.php"> <input type="hidden" name="item" value="<?php echo $user->id ?>"> ... </form> |

### Delete

La suppression est un acte simple (quand l'entité est elle-même triviale). Elle peut être implantée dans la liste d'affichage, sous forme d'un commande de suppression attachée à chaque entité.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| vf | xxxxxx | Fremaux | Valery | Administrateur | ADMIN | Y | Supprimer |
| hdm | yyyyyy | de Milleville | Hervé | Responsable du département | MEMBER | Y | Supprimer |
| rc | zzzzzz | Chelouah | Rachid | Enseignant | MEMBER | Y | Supprimer |
| bg | wwww | Glonneau | Bernard | Enseignant | MEMBER | Y | Supprimer |

## L'intérêt de bien comprendre le modèle CRUDE

Ce modèle est tout d'abord universel et extrêmement courant, même en dehors du contexte des applications de gestion. La "gestion" d'une entité de données est une constante du [paradigme](http://www.ethnoinformatique.fr/mod/glossary/showentry.php?courseid=87&concept=paradigme) informatique en lui-même.

On se rend compte que présenter le modèle CRUDE de cette façon permet d'envisager la gestion de n'importe quelle entité simple sur le même modèle. Une fois une entité écrite, on peut même dégager de cette écriture des parties de code (patterns ou schémas) qui sont très peu dépendantes de la structure de l'entité. A quelques paramètres près, on peut même tenter de générer l'ensemble de code pour une autre table, si on sait décrire celle-ci et la façon dont les éléments de cette autre table viennent intervenir dans la structure du CRUDE générique.

Ceci amène à une conclusion méthodologique du métier d'ingénieur : En travaillant sur les "processes de production de l'objet" et non sur l'objet lui-même, la puissance de production d'un développement est fortement démultiplié, et la rentabilité d'un projet avec.