

P.-A. Sunier, Haute Ecole Arc, Neuchâtel

1	Introduction.....	1
2	Les technologies de l'information (TI)	2
3	L'atelier de génie logiciel Designer	2
4	Démarche méthodologique	3
5	Exigences & analyse	3
5.1	Modélisation des processus métier.....	4
5.2	Modélisation conceptuelle des données	4
5.3	Modélisation conceptuelle des traitements.....	5
6	Transformation des modèles conceptuels.....	6
7	Conception	8
7.1	Modélisation logique de données.....	8
7.2	Modélisation logique de traitements	9
7.3	Génération du code applicatif	10
8	Liens utiles	10

1 Introduction

Dans notre premier article « Notions élémentaires » nous avons présenté *Designer* dans l'optique d'une démarche méthodologique pilotée par la modélisation conceptuelle des données en analyse. Ensuite, la modélisation des traitements était effectuée, dans une vision RAD¹, c'est-à-dire directement en phase de conception à partir des tables du modèle relationnel de données obtenu à partir de la transformation du modèle conceptuel de données.

Dans cet article, nous présenterons une démarche d'ingénierie plus complète commençant par la modélisation du métier, continuant par l'analyse des données et des traitements du système d'information pour finir par la génération du code applicatif à partir de l'enrichissement réalisé en conception.

Remarque: Comme pour les précédents articles, la conception des traitements est effectuée dans l'optique de générer des applications Web PL/SQL ; toutefois, la démarche est rigoureusement identique pour la génération d'applications Visual Basic ou Forms/Reports.

Dans cet article, nous ne traiterons pas des problèmes spécifiques de génération de code ou d'exécution des applications Web PL/SQL ; si nécessaire, le lecteur se référera à nos anciens articles et plus particulièrement au premier « Notions élémentaires ».

¹ RAD : Rapid Application Development

2 Les technologies de l'information (TI)

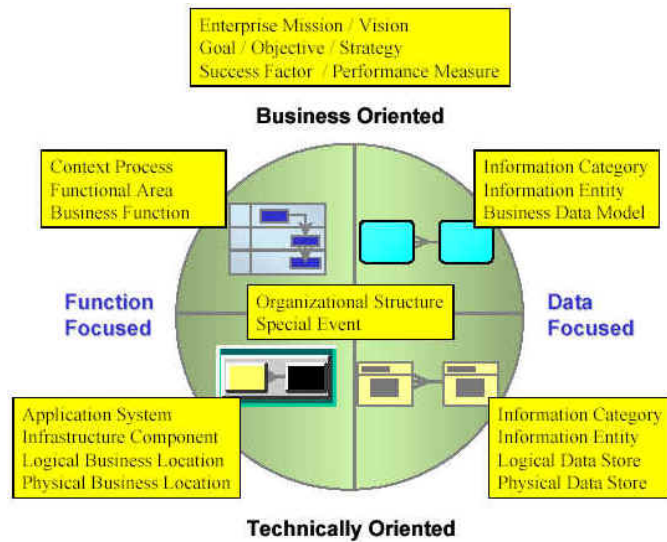


Figure 1 - Axes de modélisation

Designer est un atelier de génie logiciel² (AGL) mettant en œuvre les concepts prônés par les méthodes de développement de logiciels de gestion dites des technologies de l'information (TI).

Les technologies de l'information préconisent de :

- élaborer une stratégie applicable au système d'information dans laquelle s'inscrit tout développement logiciel ;
- avoir une vue globale de l'entreprise et de son système d'information ;
- considérer plusieurs niveaux d'abstraction lors de la modélisation ;
- mettre en évidence la dualité des données et des traitements ;
- utiliser des outils logiciels pour toutes les activités automatisables.

3 L'atelier de génie logiciel Designer

Designer est un outil de la catégorie des ateliers de génie logiciel intégrés ou « Integrated case » en anglais.

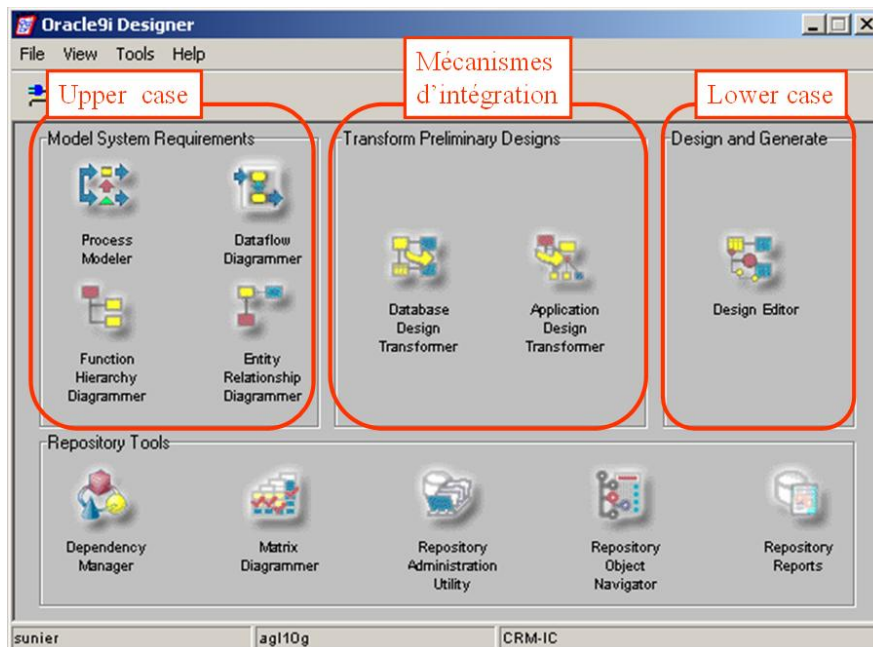


Figure 2 - Panneau de commande de *Designer*

Un atelier de génie logiciel est qualifié de « Upper case » lorsqu'il offre un support pour la modélisation à réaliser aux niveaux d'abstraction les plus élevés ; avec *Designer*, ce sont les modèles du métier et les modèles conceptuels de données et de flux de données.

Un atelier de génie logiciel est qualifié de « Lower case » lorsqu'il offre un support pour la modélisation à réaliser aux niveaux d'abstraction les moins élevés ; avec *Designer*, ce sont les modèles logiques de données et de traitements. Lorsqu'un atelier de génie logiciel couvre les besoins de modélisation aux divers niveaux d'abstraction et offre des mécanismes de

transformation automatique entre modèles de niveaux différents, il est qualifié d'intégré.

² Case, Computer Aided System Engineering en anglais; nous voyons que le terme anglais met en évidence l'aide apportée par l'outil logiciel.

4 Démarche méthodologique

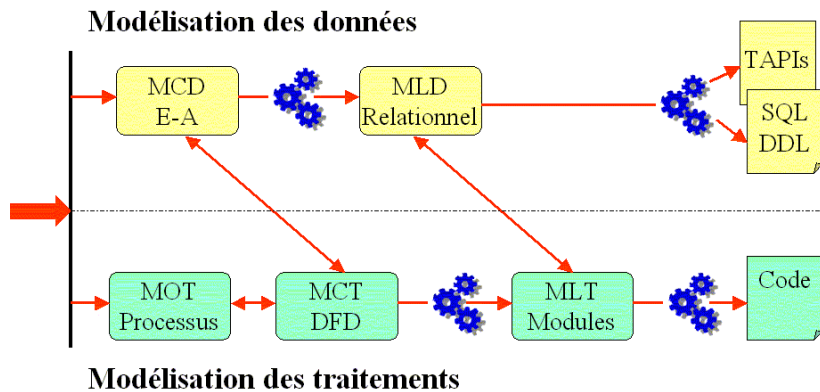


Figure 3 - Enchaînement d'activités

Conformément aux concepts méthodologiques des technologies de l'information (TI), nous préconisons une démarche d'ingénierie illustrée par la figure 3.

Succinctement, l'enchaînement d'activités est le suivant :

1. modélisation des processus métier (MOT); éventuellement modélisation conceptuelle des données (MCD);
2. modélisation conceptuelle des données (MCD) et modélisation conceptuelles des traitements (MCT); les

deux activités peuvent se faire dans n'importe quel ordre ou par allers et retours successifs ;

3. transformation des données (les entités deviennent des tables);
4. enrichissement du modèle logique de données (MLD) ;
5. transformation des traitements (les fonctions deviennent des modules);
6. enrichissement du modèle logique de traitement (MLT) ;
7. génération du code de création de la structure de données ;
8. génération du code du logiciel applicatif.

De manière générale, l'enchaînement d'activités est modulable selon les impératifs du projet ou de la méthode de développement. La contrainte principale à respecter a trait à la présence des données sur les traitements ; les traitements dépendants des données, il est impératif de transformer les données avant les traitements.

Par ailleurs, la transformation des données entre niveau conceptuel et niveau logique peut être itérative ; ce n'est pas le cas au niveau des traitements.

5 Exigences & analyse

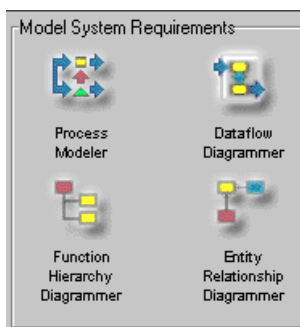


Figure 4 - Upper case

La partie « Upper case » de *Designer* permet de modéliser les exigences et de réaliser les modèles d'analyse en utilisant les composants suivants :

- (a) *Process Modeler* pour la modélisation des processus métier de l'entreprise vue dans son ensemble.
- (b) *Entity Relationship Diagrammer* pour la modélisation conceptuelle des données du système d'information.

(c) *Dataflow Diagrammer* pour la modélisation conceptuelle des traitements du système d'information.

Remarque : *Function Hierarchy Diagrammer* est un composant qui permet de visualiser la hiérarchie des fonctions.

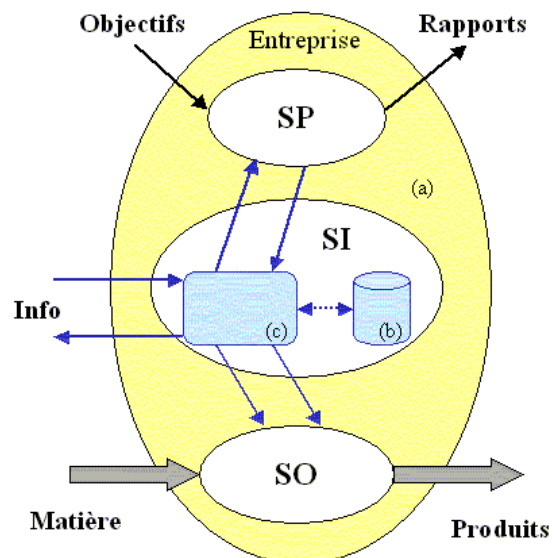


Figure 5 - Cibles de modélisation

5.1 Modélisation des processus métier

Le modèle des processus métier est organisé sous forme de travées horizontales qui représentent des unités d'organisation ou des rôles.

Le modèle de processus décompose un processus sous forme d'étapes ; chacune des étapes est, à son tour, un processus qui peut être décrit sous forme d'étapes et ainsi de suite, en créant autant de niveaux que nécessaire. Le modèle des processus métier est constitué de :

- unités d'organisation ou rôles ;
- étapes de processus attribuées à des unités d'organisation qui ont la responsabilité de leur exécution ;
- entrepôts de matière ou de données alimentés ou consommés par les étapes de processus ;
- événements déclencheurs qui initialisent un enchaînement d'étapes de processus ;
- événements de sortie émis par l'étape de processus terminale ;
- flux de matière ou de données à l'intérieur d'un processus.

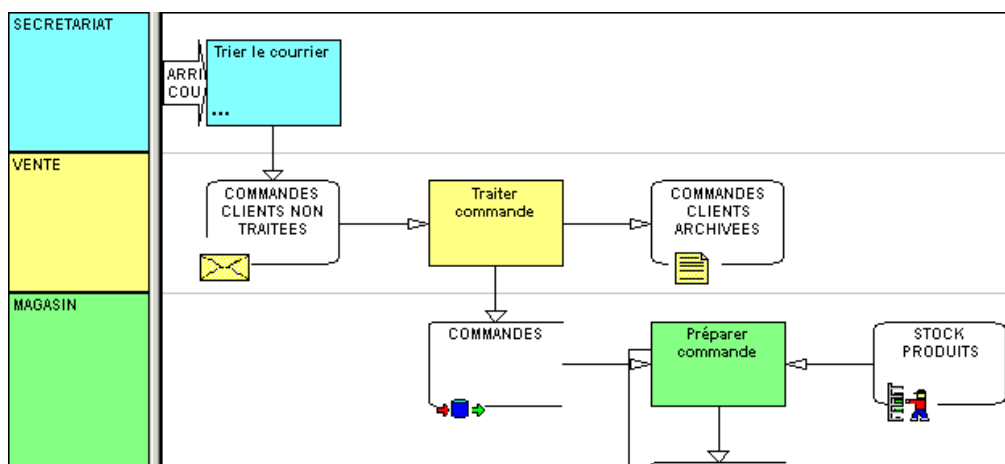
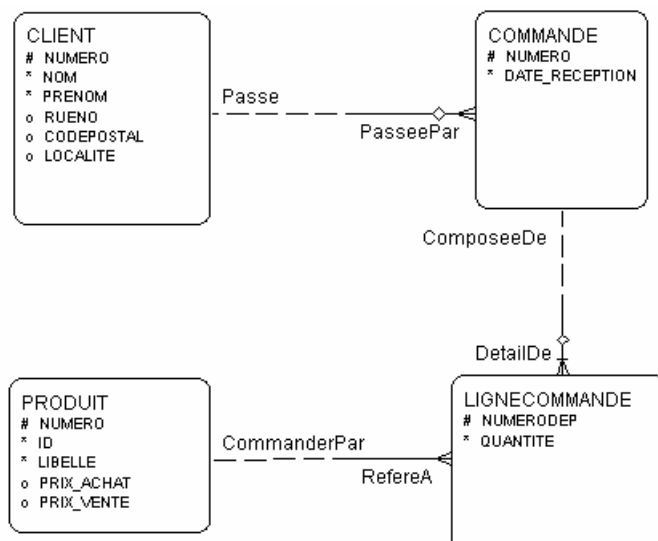


Figure 6 - Processus commercial d'une société de vente

La figure 6 représente le début des étapes du processus commercial d'une société de vente; nous voyons que le processus est déclenché par l'arrivée de courrier au secrétariat; la première étape est le tri du courrier ; les commandes de clients sont mises à disposition du service de vente qui va traiter les commandes.

5.2 Modélisation conceptuelle des données



Pour la modélisation conceptuelle des données sous forme d'entités et d'associations, nous renvoyons le lecteur à nos premiers articles, essentiellement les trois premiers.

La figure 7 représente les données que nous avons modélisées pour l'enregistrement des commandes reçues des clients.

Figure 7 -Structure conceptuelle de données relative aux commandes de clients

5.3 Modélisation conceptuelle des traitements

La modélisation conceptuelle des traitements s'effectue avec le formalisme proposé au milieu des années '70 par De Marco et connu sous le nom de « Diagramme de flux de données » communément abrégé DFD.

Le modèle de flux de données (DFD) est construit à partir d'une fonction dite frontière.

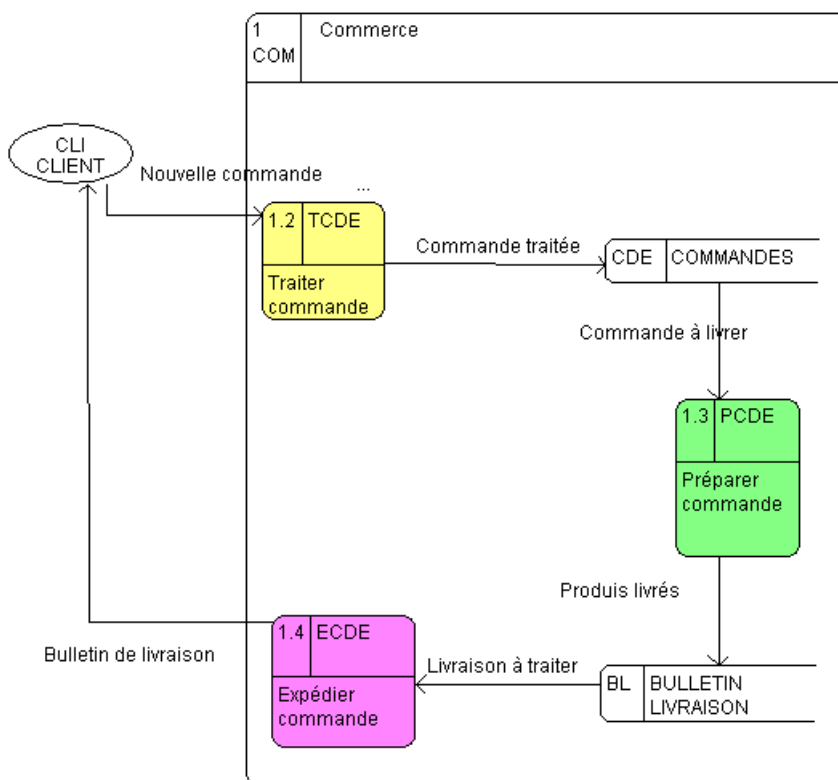
Tout comme pour la modélisation des processus, cette fonction frontière est décomposée en parties que l'on nomme des fonctions locales ; à leur tour, les fonctions locales peuvent devenir des fonctions frontières qui sont décomposées en fonctions locales de niveau inférieur et ceci à autant de niveaux que nécessaire.

Une fonction de DFD ou un processus de la modélisation de processus sont deux interfaces d'un seul et unique objet du référentiel nommé *Business function*. De ce fait, en passant de la modélisation du métier à la modélisation du système d'information, les processus incluant des traitements de données deviennent automatiquement des fonctions ; un objet *Business function* du référentiel présente simplement des propriétés différentes dans les deux modèles.

Le modèle de flux de données (DFD) est constitué de :

- acteurs externes qui représentent l'environnement du système d'information ; ce sont eux qui déclenchent l'exécution des fonctions de haut niveau par l'envoi de données et en reçoivent quittance sous forme de données;
- fonctions qui représentent l'acquisition, le stockage, le traitement proprement dit ou la restitution de données par le système d'information;
- entrepôts qui représentent les données stockées par le système d'information ;
- flux de données qui représentent les échanges d'information à l'intérieur du système d'information ou avec son environnement, c'est-à-dire : le système opérant, le système de pilotage ou l'environnement de l'entreprise.

La figure 5 symbolise cette représentation du système d'information ; (c) représente les fonctions, (b) les entrepôts et les flèches bleues correspondent aux échanges de données.



La figure 8 correspond à la partie du système d'information du processus métier de gestion commerciale présenté en figure 6.

La fonction frontière **COMMERCE** n'a pas été créée pour l'élaboration de ce modèle ; elle est issue de la modélisation précédente des processus et il en est de même pour les trois fonctions locales.

Il est à noter que le processus métier **Trier le courrier** n'apparaît pas car, nous ne le considérons pas comme faisant partie du système d'information.

Par contre nous avons créé une entité externe **CLIENT** qui par l'envoi d'une commande déclenche la fonction **COMMERCE**.

Figure 8 - DFD de la fonction COMMERCE

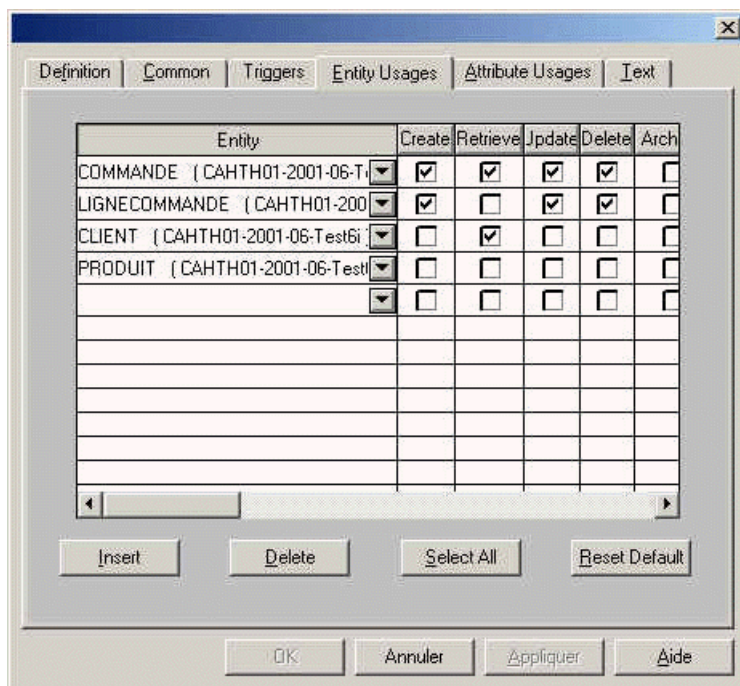


Figure 9 - Utilisation d'entités par la fonction ENREGCDE

Selon les concepts de modélisation des flux de données (DFD), la structure des entrepôts et des flux de données est définie à partir du modèle conceptuel de données.

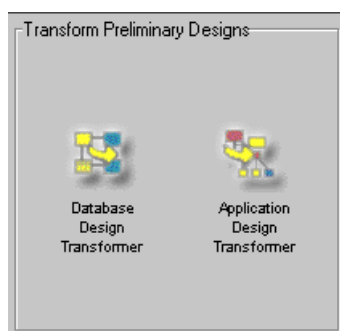
Toutefois, pour effectuer la transformation des fonctions en modules de niveau logique, *Designer* exige de spécifier l'utilisation des entités par les fonctions même si cette information pourrait être déduite des flux de données entrants et sortants des fonctions.

Pour chaque fonction qui devra être transformée en module applicatif, il y a lieu de procéder à cet enrichissement ; en plus de définir les entités utilisées et les opérations à leur appliquer, nous pouvons définir les utilisations d'attributs de chacune des entités.

La figure 9 montre les entités utilisées par la fonction **Enregistrer commande**; cette fonction est une fonction locale de la fonction **Traiter commande**.

L'organisation hiérarchique des fonctions est visible sur la figure 11.

6 Transformation des modèles conceptuels



La transformation des entités et associations du modèle conceptuel de données en tables et relations entre tables doit être réalisée préalablement à la transformation des fonctions en modules. En effet, l'utilisation des entités par les fonctions doit être transformée en utilisation de tables par les modules ; pour ce faire, *Application Design Transformer* doit trouver dans le référentiel les relations d'implémentation³ entre entités et tables créées par *Database Design Transformer*.

Figure 10 - Mécanismes d'intégration

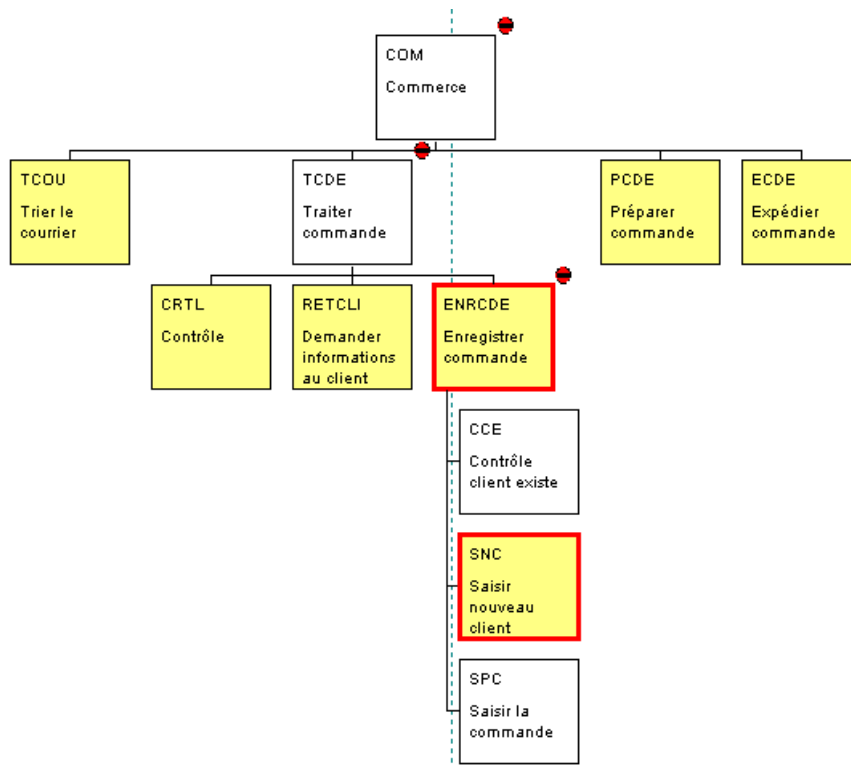
La transformation des données peut être itérative, c'est-à-dire qu'il est possible d'enrichir le modèle conceptuel et ensuite de le transformer alors que le modèle logique existe déjà. L'outil de transformation *Database Design Transformer* est tout à fait capable de reporter les changements sans altérer le modèle logique qui aurait lui-même été enrichi.

A l'inverse, l'outil de transformation *Application Design Transformer* crée un nouveau module pour implémenter une fonction ; cette transformation est unique. Le module créé est, en quelque sorte, un squelette qui doit être finalisé en conception.

³ Voir le premier article pour plus de détails sur ce lien de dépendance et le mécanisme de transformation des données.

Pour définir les fonctions d'analyse qui doivent être transformées en modules, nous conseillons d'utiliser le composant *Function Hierarchy Diagrammer*; ce composant permet de visualiser la hiérarchie des fonctions. De manière simplifiée, *Application Design Transformer* crée un module à partir d'une fonction si ;

- la fonction est définie comme élémentaire ;
- la fonction est une feuille de l'arborescence et qu'elle n'a pas d'ancêtre élémentaire.



La figure 11 montre la hiérarchie de la *Business Function COMMERCER* ; nous pouvons observer que le processus métier **Trier le courrier** apparaît car le diagrammeur montre les *Business Function* qu'elles soient utilisées en modélisation des processus et/ou en modélisation des flux de données. Pour permettre une représentation visuelle des fonctions élémentaires qui seront transformées en modules nous les avons encadrées en rouge. Remarque : Nous avons colorié en jaune les fonctions qui deviendront des modules candidats.

Figure 11 - Hiérarchie de la *Business Function COMMERCER*

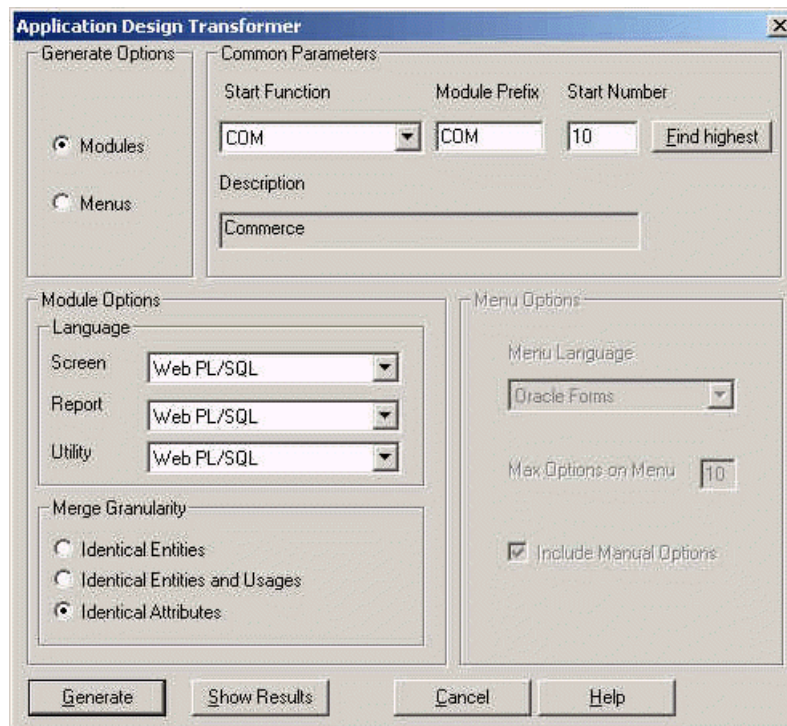


Figure 12 - Paramétrage de *Application Design Transformer*

Application Design Transformer crée des modules dits candidats. Après transformation, il nous appartient de définir parmi les modules candidats ceux qui seront effectivement pris en compte pour réaliser l'application ; pour notre exemple, il est évident que le module candidat qui implémente le processus métier **Trier le courrier** ne sera pas retenu.

Après notre acceptation des modules candidats, nous pouvons utiliser une deuxième fois le composant de transformation pour créer le menu de l'application ; la structure du menu sera réalisée à partir des unités d'organisation du modèle de processus. Chaque unité d'organisation deviendra une entrée principale du menu et il y aura autant d'entrées principales qu'il y a d'unités mises en œuvre pour réaliser la fonction racine du menu.

7 Conception



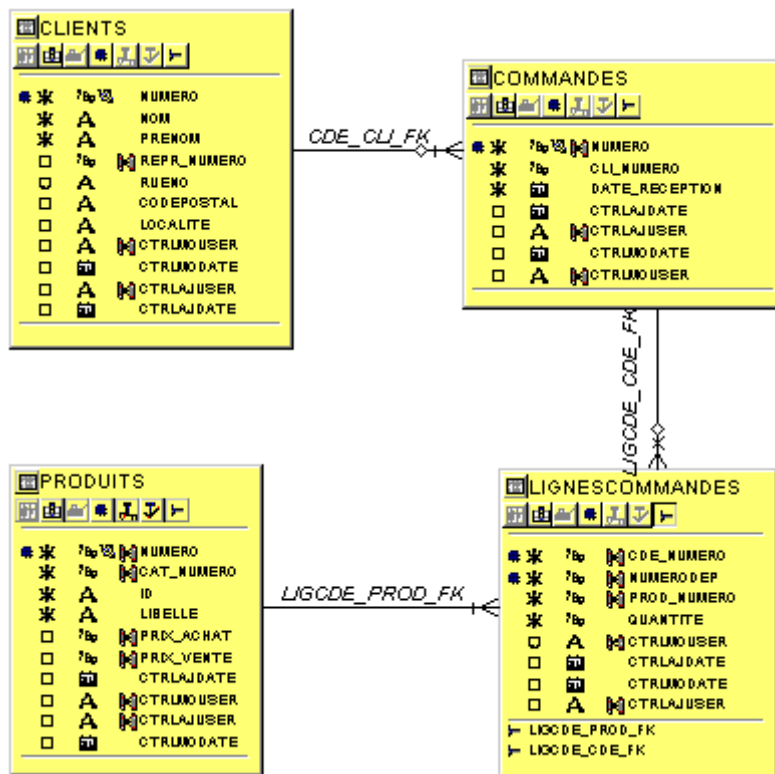
La partie « Lower case » de *Designer* permet de réaliser les modèles de conception et de générer le code.

Le composant *Design Editor* est constitué de deux parties essentielles :

- la modélisation et la génération de code des données, Server Model ;
- la modélisation et la génération de code des traitements, les modules.

Figure 13 - Lower case

7.1 Modélisation logique de données



Pour la modélisation logique des données sous forme de tables et de relations, nous renvoyons le lecteur à nos premiers articles, essentiellement les trois premiers.

La figure14 représente le modèle relationnel pour l'enregistrement des commandes reçues des clients. Les tables et relations ont été créées par le composant de transformation *Database Design Transformer*.

Figure 14 - Structure logique de données relative des commandes

7.2 Modélisation logique de traitements

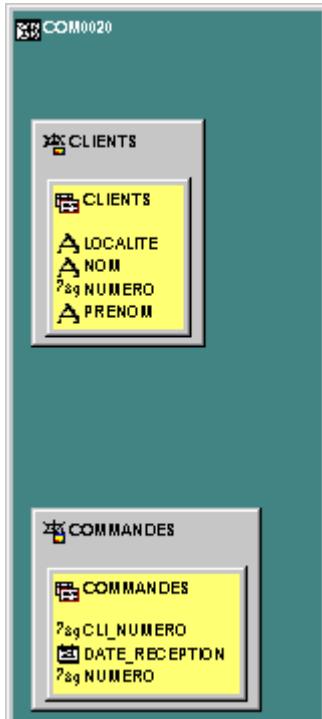


Figure 15 - 1ère partie du squelette de module COM0020

Après transformation des fonctions en modules candidats et acceptation de ceux qui feront partie de l'application, nous obtenons des squelettes de modules.

Chaque squelette de module est constitué d'autant de composants qu'il y a d'utilisation d'entités dans la fonction qu'il implémente. Chaque composant est doté d'une seule utilisation de table de base; la table de base est celle qui est implémentée à partir de la transformation de l'entité correspondante.

Les figures 15 & 16 représentent le squelette de module COM0020 obtenu par la transformation de la fonction **Enregistrer commande**. Nous pouvons observer les 4 composants et leur utilisation de table de base. Chacune de ces tables de base a été spécifiée par l'outil de transformation à partir des utilisations d'entités définies pour la fonction **Enregistrer commande** et illustré par la figure 9.

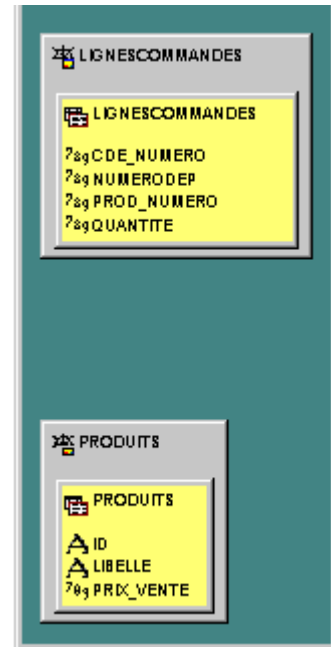


Figure 16 - 2ème partie du squelette de module COM0020

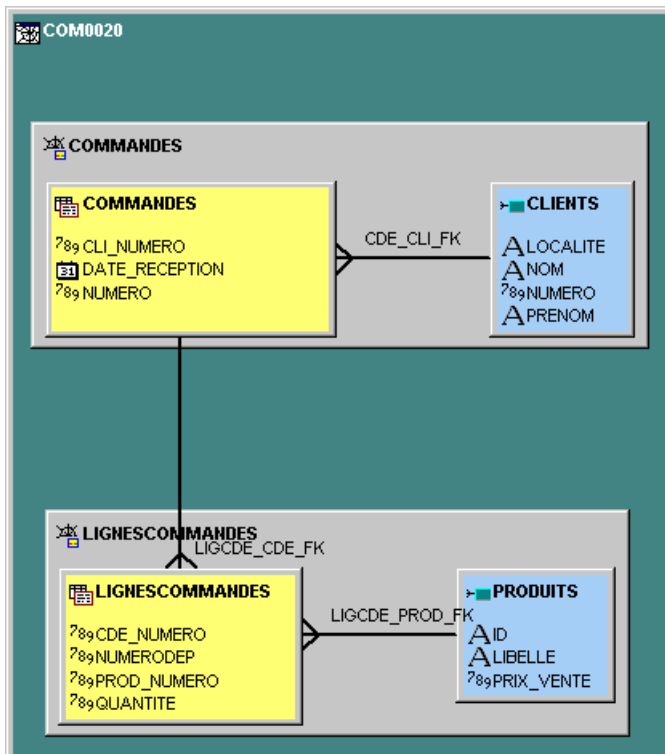


Figure 17 - Module COM0020 enrichi

Dataflow Diagrammer et le référentiel de *Designer* nous permettent de définir les entités utilisées par les fonctions mais pas les éventuelles associations existantes entre les entités. Dès lors, il est évident que les squelettes générés ne contiennent aucune relation entre tables et en cascade, il est impossible d'avoir des tables de référence ou des dépendances maître et détails entre composants.

Les squelettes doivent être enrichis en conception. Nous définissons les choix de présentation des formulaires et nous établissons les liens entre tables pour obtenir une structure d'utilisation de tables appropriée.

La figure 17 montre les liens que nous avons établis entre les tables ; il nous a suffi de déplacer les tables **CLIENTS** et **PRODUITS** dans les composants appropriés pour obtenir le modèle ci-contre.

7.3 Génération du code applicatif

Pour la génération du code de création de la structure de données et des modules de traitements, nous renvoyons le lecteur à nos premiers articles, essentiellement, les trois premiers.

8 Liens utiles

Portail de l'atelier de génie logiciel de la HE-Arc

<http://lgl.isnetne.ch>

- Site de formation à Designer
Rubrique: *Enseignement / Atelier de génie logiciel / Designer 6i/9i – Cours de formation de base*
- Site de référencement des anciens articles.
Rubrique: *Publication / Informatique / Designer et l'ingénierie du logiciel*