Niveau logique

Les outils sont évalués sur leur capacité à générer un schéma relationnel correct et sur la possibilité de définir d'éventuelles contraintes de répercussion sur les clés étrangères (CASCADE) pour la suite de processus de conception.

Le modèle relationnel attendu est le suivant.

Figure 4-6 *Modèle relationnel attendu*

Client[num, nom, adresse, numtel, numParrain#]

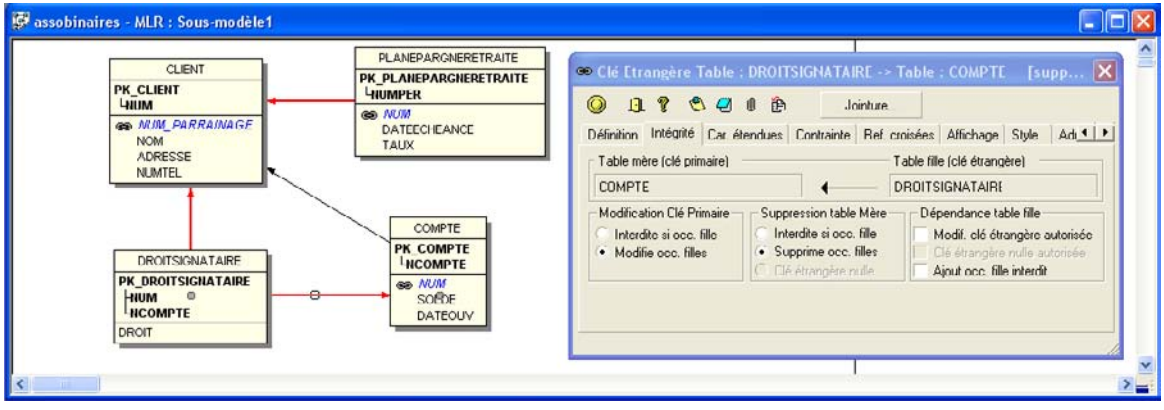
PlanEpargneRetraite[numPER, dateEcheance, taux, num#]

DroitSignataire[num#, nCompte#, droit]

Compte[nCompte, solde, dateOuv, num#]

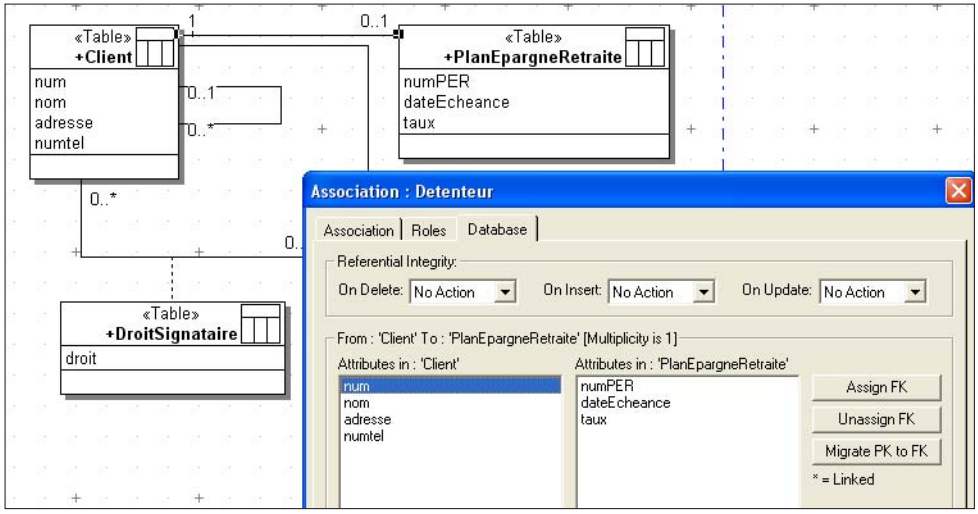
Bon nombre d'outils permettent d'ajouter des contraintes au niveau des clés étrangères. Win'Design note en rouge les liens qui symbolisent la répercussion d'une suppression (ici la suppresssion d'un compte devrait entraîner celle des lignes dans la table des signataires, de même pour la suppression d'un client).

Figure 4-7 *Modèle logique Win'Design*



Un mauvais point pour Visual UML qui nécessite (comme JDeveloper d'Oracle) de définir manuellement les clés étrangères pour chaque association ! Dans le cas de l'association présente, il faudrait ajouter un attribut dans la classe `PlanEpargneRetraite` puis le lier à l'identifiant de la classe `Client`.

Figure 4-8 *Définition d'une clé étrangère avec Visual UML*



Script SQL

Les scripts SQL générés sont évalués sur la dénomination des contraintes (utilisation des noms des associations ou des rôles) et la position de la clé étrangère de l'association *un-à-un* (qui devrait, en théorie, se situer dans la table `PlanEpargneRetraite`).

Bilan intermédiaire

La majorité des outils permettent une saisie correcte au niveau conceptuel. ModelSphere et MyEclipse pêchent par manque de fonctionnalités tandis que Visual Paradigm nécessite une manipulation complexe inter-modèles. Au niveau logique, des différences d'implémentation (ou des insuffisances) apparaissent. Seuls Win'Design, Rational Rose et Objecteering font un sans faute. Un bémol pour MagicDraw, Visual Paradigm, Together et Enterprise Architect qui ne disposent pas du mécanisme d'identifiant de classe. À noter que Together représente une classe-association avec le même symbole que celui de l'association *n*-aire.

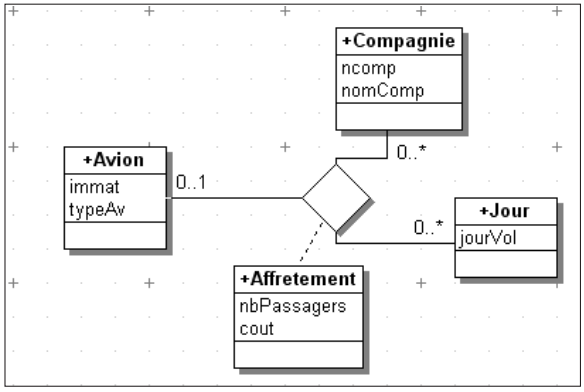
Tableau 4.1 Associations binaires

Associations binaires	Niveau conceptuel avec UML (identifiant, associations, multiplicités, rôles, etc.)	Modèle logique	Code SQL (contraintes de clés étrangères)
Enterprise Architect			
MagicDraw			
MEGA Designer			
ModelSphere			
MyEclipse			
Objecteering			
Poseidon			
PowerAMC			
Rational Rose Data Modeler			
Together			
Visio			
Visual Paradigm			
Visual UML			
Win'Design			

Associations *n*-aires

L'exemple 4-9 décrit une association *3*-aire (avec attributs) qui modélise les affrètements d'avions par différentes compagnies au cours du temps.

Figure 4-9 *Association* n*-aire (Visual UML)*



Niveau conceptuel

Les outils sont évalués sur la possibilité de représenter une association *n*-aire à l'aide du symbole losange (*diamond*), de nommer l'association et d'y apposer différentes multiplicités maximales (1 ou plus généralement *).

Figure 4-10 *Association* n*-aire (Together)*

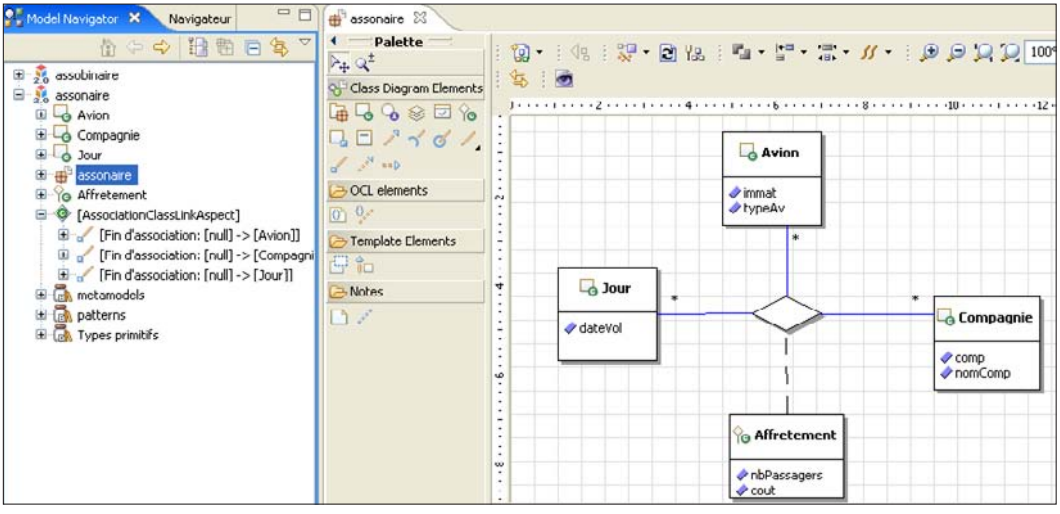
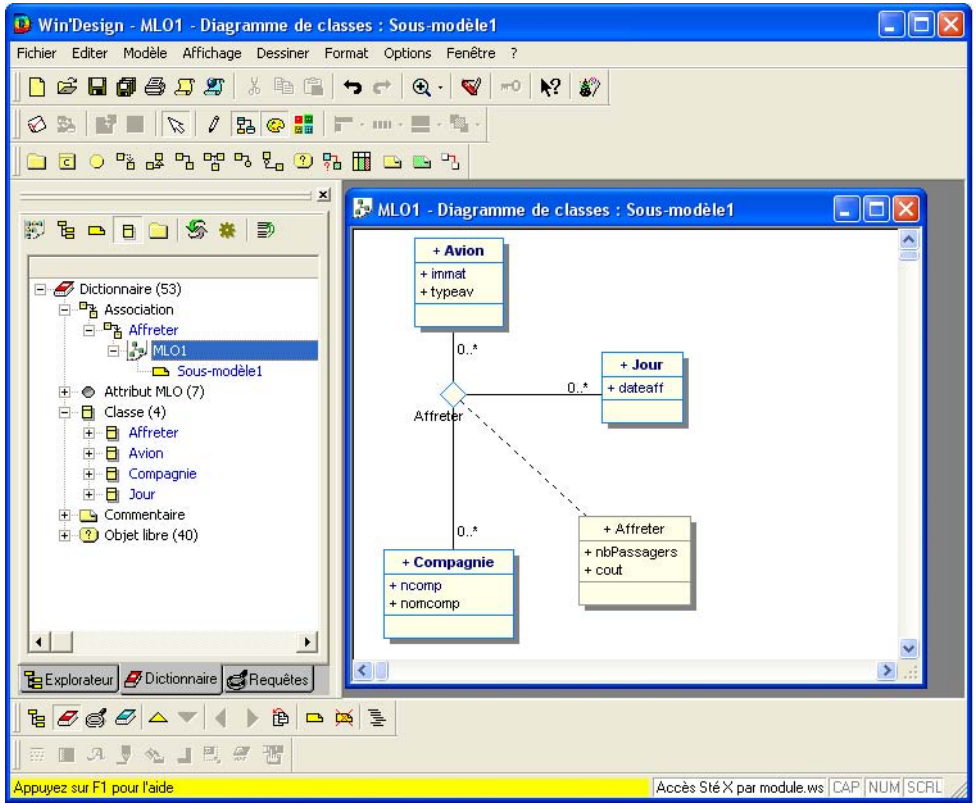


Figure 4-11 *Association* n-*aire (Win'Design)*



Niveau logique

Les outils sont évalués sur leur capacité à transformer une association *n*-aire selon la valeur des multiplicités (en particulier pour 0..1 et 1, le degré de la clé primaire de la relation dérivée doit être réduit).

Le modèle relationnel attendu est le suivant.

Figure 4-12 *Modèle relationnel attendu*

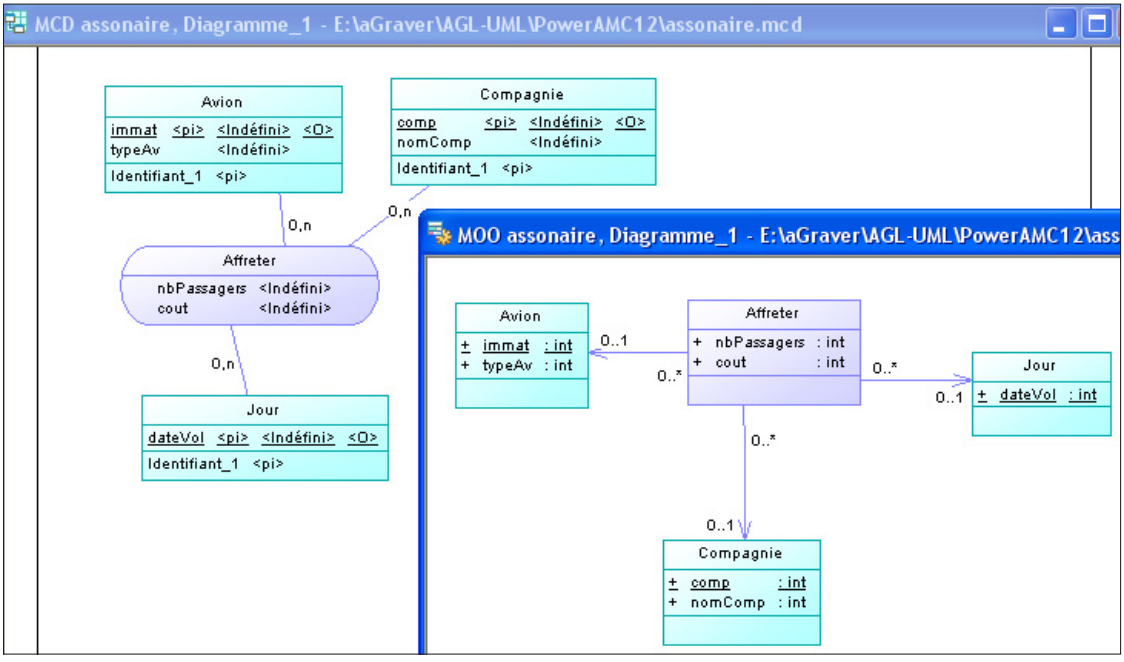
Avion[immat, typeAv]

Compagnie[comp, nomComp]

Affretement[immat#, comp#, dateAff#, nbPassagers, cout]

Jour[dateAff]

Figure 4-13 *Association* n*-aire (Power AMC)*



Script SQL

Les outils sont évalués sur la possibilité de ne pas traduire une relation en table tout en la conservant au niveau logique (dans notre exemple, il n'est pas souhaitable de transformer la relation `Jour` en une table) et la traduction d'éventuelles multiplicités 0..1 et 1 (clé primaire réduite ou contrainte `UNIQUE`).

Bilan intermédiaire

Seuls cinq outils sont vraiment à l'aise au niveau conceptuel avec les associations *n*-aire. Le diagramme test ajoutait à la difficulté le fait de connecter une classe-association. Le grand gagnant de cet exercice est Win'Design qui maîtrise le processus de bout en bout.

Tableau 4.2 Associations *n*-aires

Associations *n*-aires	Niveau conceptuel avec UML (symbole losange, multiplicités)	Modèle logique	Code SQL
Enterprise Architect			
MagicDraw			
MEGA Designer			
ModelSphere			
MyEclipse			
Objecteering			
Poseidon			
PowerAMC			
Rational Rose Data Modeler			
Together			
Visio			
Visual Paradigm			
Visual UML			
Win'Design			

Classes-associations

L'exemple 4-14 regroupe sur une modélisation plusieurs associations sur une classe-association (compilation des exemples 2-52, 2-56 et 2-59). Différentes missions peuvent être suivies par un employé (en tant que passager ou conducteur). De plus, un employé pourra choisir une mission particulière pour calculer un bonus. Il est à noter que cette même base de données nous servira de test pour le processus de rétroconception.

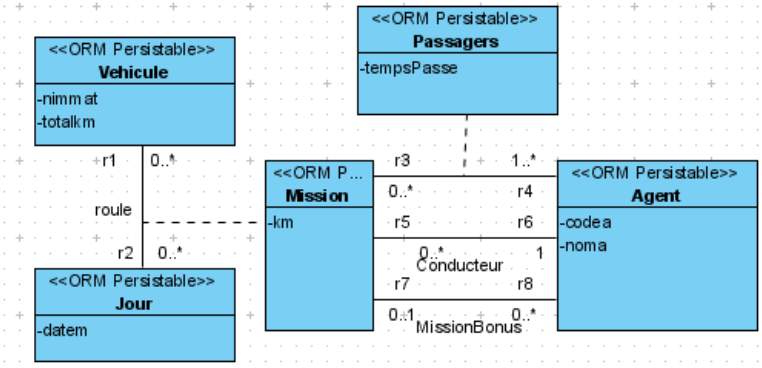


Figure 4-14 *Classes-associations (Visual Paradigm)*

Niveau conceptuel

Les outils sont évalués sur la capacité de représenter toutes les multiplicités et plusieurs liens sur la même classe-association.

Figure 4-15 *Classes-associations (Together)*

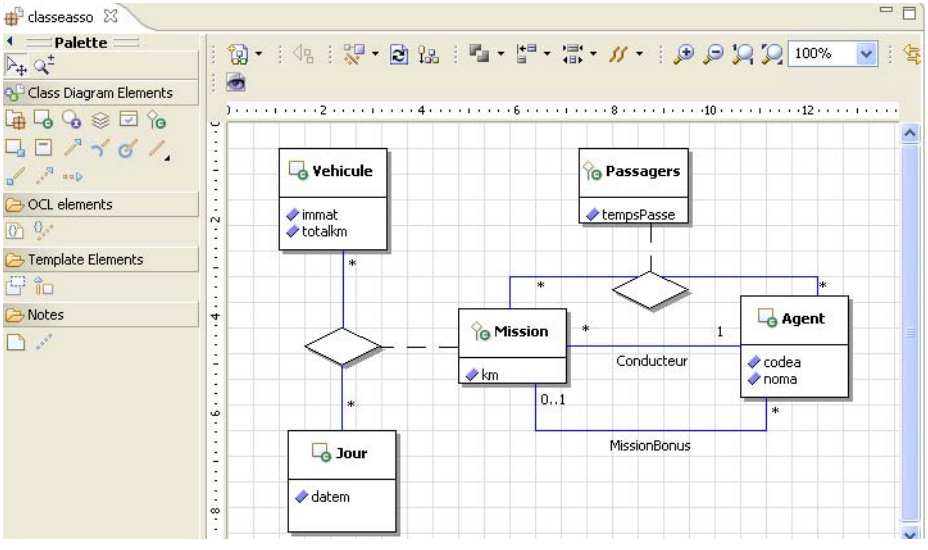
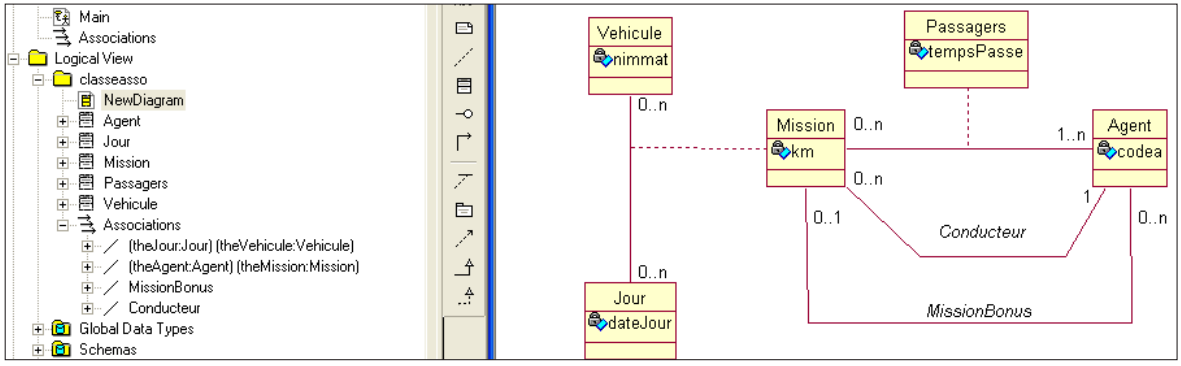


Figure 4-16 *Classes-associations (Rational Rose)*



Niveau logique

Les outils sont évalués sur la capacité de générer correctement un schéma relationnel en fonction des multiplicités des classes-associations.

Le modèle relationnel attendu est le suivant.

Figure 4-17 *Modèle relationnel attendu*

Vehicule[nimmat, totalkm]

Agent[codea, noma, (nimmat, dateJour)#]

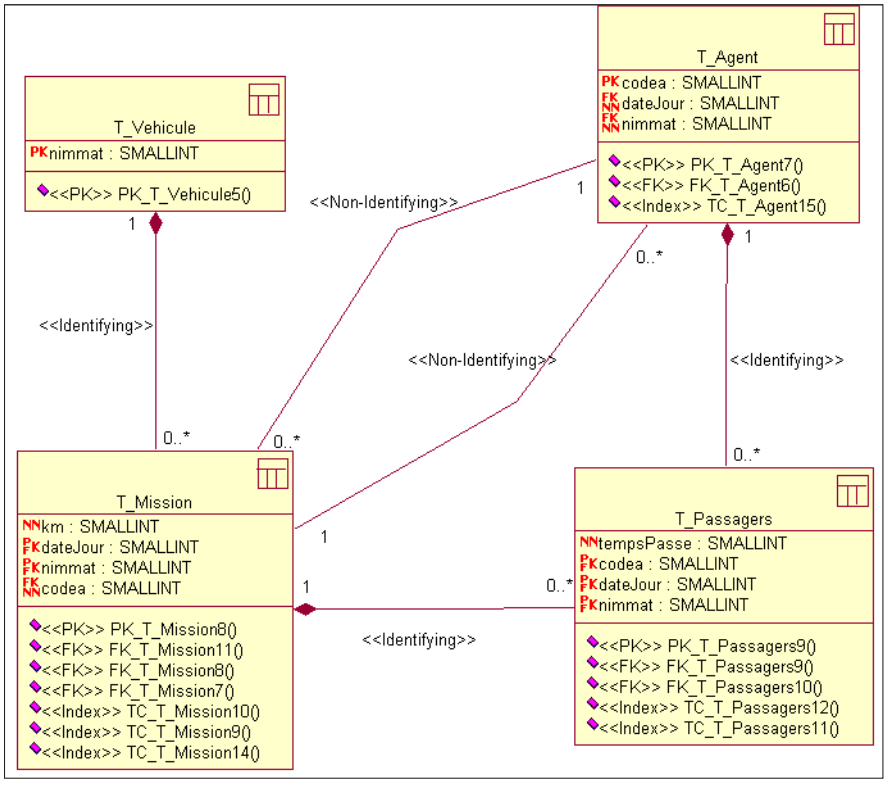
Mission[nimmat#, dateJour#, km, codea#]

Jour[dateJour]

Passagers[(nimmat, dateJour)#, codea#, tempsPasse]

Le modèle logique de Rational Rose est décrit à l'aide du profil UML.

Figure 4-18 *Classes-associations (Rational Rose)*



Script SQL

Les outils sont évalués sur la dénomination des clés étrangères (on devrait pouvoir retrouver le nom de certaines associations liées à la classe-association).

Bilan intermédiaire

C'est au niveau logique, durant la phase de migration des clés étrangères, que les disparités se produisent. Seuls trois outils passent ce test haut la main, il s'agit de Objecteering, Rational Rose et PowerAMC.

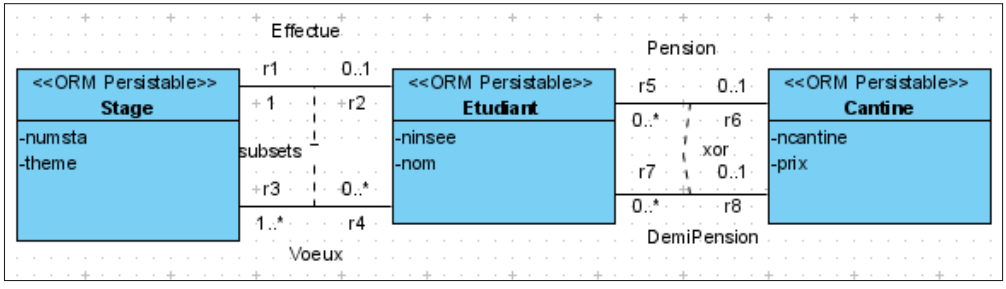
Tableau 4.3 Classes-associations

2	Niveau conceptuel avec UML (plusieurs liens sur la classe-association)	Modèle logique	Code SQL
Enterprise Architect			
MagicDraw			
MEGA Designer			
ModelSphere			
MyEclipse			
Objecteering			
Poseidon			
PowerAMC			
Rational Rose Data Modeler			
Together			
Visio			
Visual Paradigm			
Visual UML			
Win'Design			

Contraintes

L'exemple 4-19 illustre deux contraintes prédéfinies de UML 2 (partition : `xor` et inclusion : `subsets`). La partition exprime qu'un étudiant est soit pensionnaire soit demi-pensionnaire, l'inclusion signifie que le stage d'un étudiant doit être au préalablement souhaité.

Figure 4-19 *Contraintes (Visual Paradigm)*



Niveau conceptuel

Les outils sont évalués sur la capacité de représenter ces contraintes (stéréotypes prédéfinis, menu contextuel, etc.). Dans la majorité des cas, la contrainte doit être saisie manuellement ou le stéréotype doit être créé pour pouvoir être réutilisé par la suite. Peu d'outils proposent des stéréotypes prédéfinis comme le montrent les écrans suivants.

Figure 4-20 *Contraintes (Objecteering)*

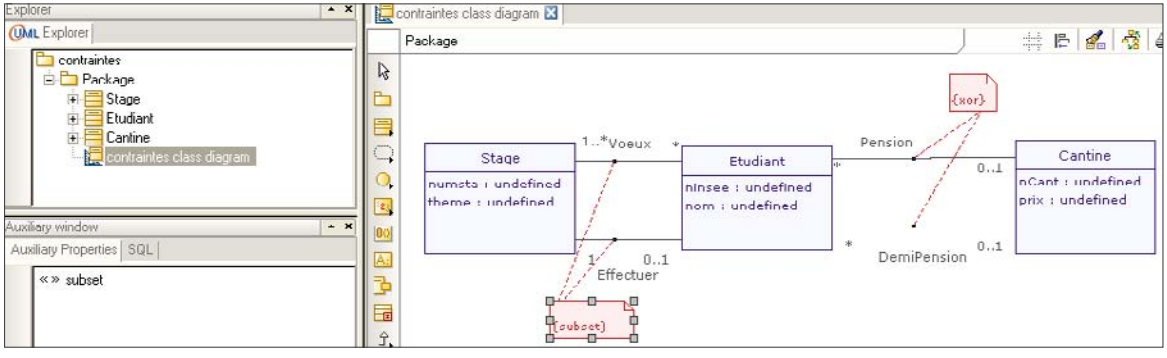
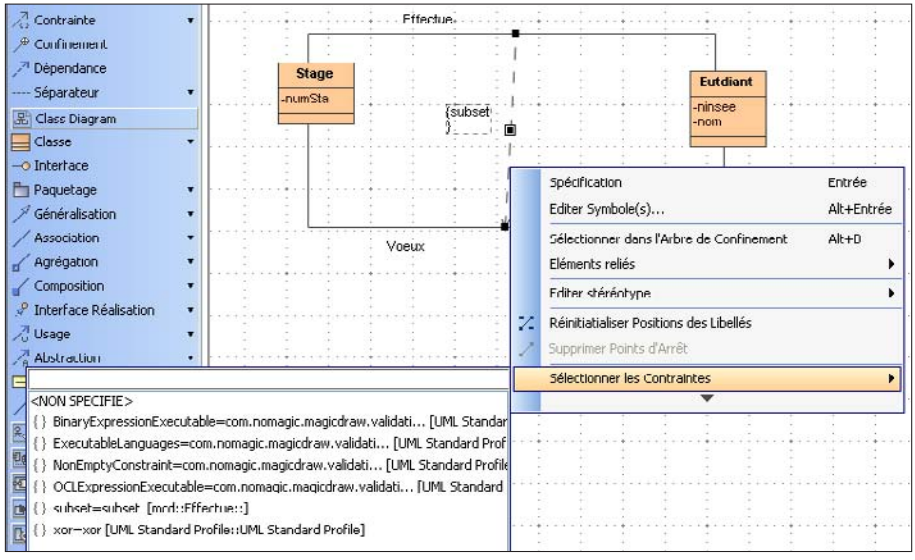


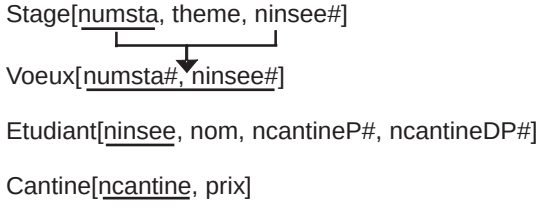
Figure 4-21 *Contraintes (MagicDraw)*



Niveau logique

Le modèle relationnel attendu est représenté à la figure 4-22. Ces deux contraintes n'ont pas d'influence sur la structure des relations générées (idéalement, une clé étrangère devrait être toutefois ajoutée entre `Stage` et `Voeux` pour assurer l'inclusion).

Figure 4-22 *Modèle relationnel attendu*



Script SQL

Les outils sont évalués sur la capacité à générer la clé étrangère induite de la contrainte d'inclusion et la directive SQL de vérification (`CHECK`) pour implémenter la contrainte de partition (non-nullité exlusive des clés étrangères liant `Etudiant` à `Cantine`). Cette dernière contrainte pourrait aussi être assurée par un déclencheur (qui pourrait être déclaré automatiquement mais qu'il faudrait par la suite programmer manuellement).

Bilan intermédiaire

Tableau 4.4 Contraintes

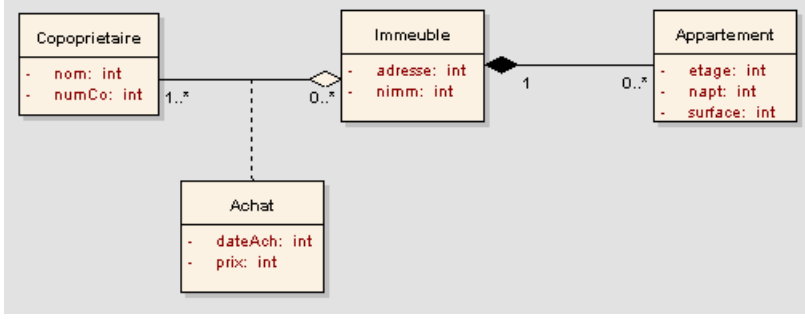
Contraintes	Niveau conceptuel avec UML (partition et inclusion)	Code SQL (clé étrangère, contrainte CHECK ou déclencheur
Enterprise Architect		
MagicDraw		
MEGA Designer		
ModelSphere		
MyEclipse		
Objecteering		
Poseidon		
PowerAMC		
Rational Rose Data Modeler		
Together		
Visio		
Visual Paradigm		
Visual UML		
Win'Design		

Bien que certaines contraintes prédéfinies de UML 2 soient déjà répertoriées par certains outils, la traduction du conceptuel au physique n'est pas encore automatisée. Il vous incombera donc de programmer explicitement au niveau du code SQL l'enrichissement sémantique de vos diagrammes de classes.

Agrégations

L'exemple 4-23 met en œuvre une composition (qui exprime qu'un appartement est un composant d'un immeuble) et une agrégation partagée (qui renforce l'association d'achat entre un copropriétaire et un immeuble).

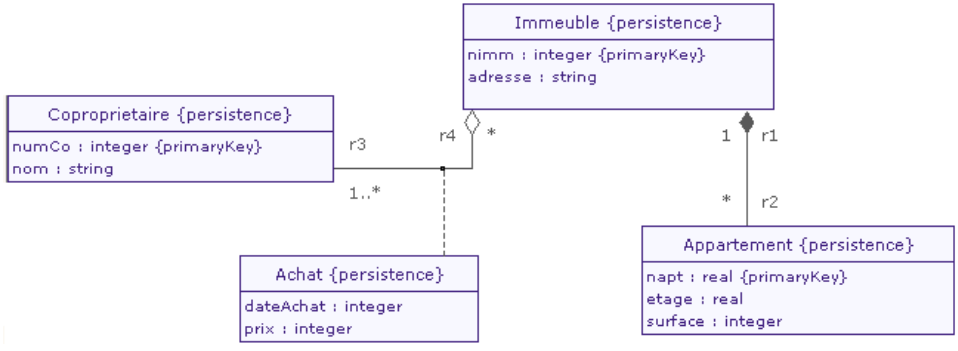
Figure 4-23 *Agrégations (Enterprise Architect)*



Niveau conceptuel

Les outils sont évalués sur la capacité de représenter ces deux formes d'agrégation.

Figure 4-24 *Agrégations (Objecteering)*



Niveau logique

Les outils sont évalués sur la capacité de générer un schéma relationnel prenant en compte la composition des tables (clé primaire de la table composant enrichie par la clé primaire de la table composite). Le modèle relationnel attendu est le suivant.

Figure 4-25 *Modèle relationnel attendu*

Coproprietaire[numco, nom]

Immeuble[nimm, adresse]

Achat[nimm#, numco#, prix, dateAchat]

Appartement[nimm#, napt, surface, etage]

Les captures d'écran suivantes illustrent un tel modèle logique.

Figure 4-26 *Agrégations (Win'Design)*

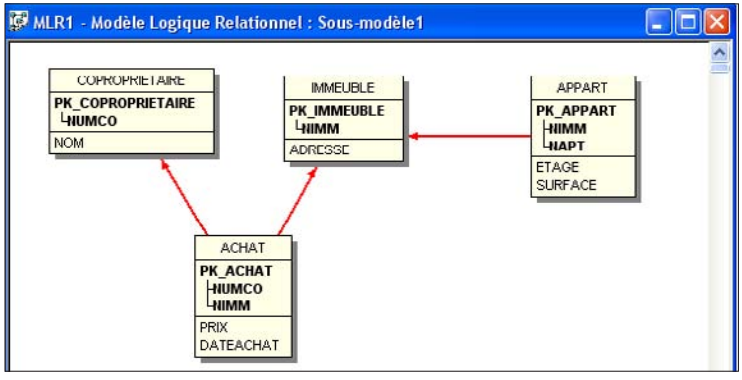
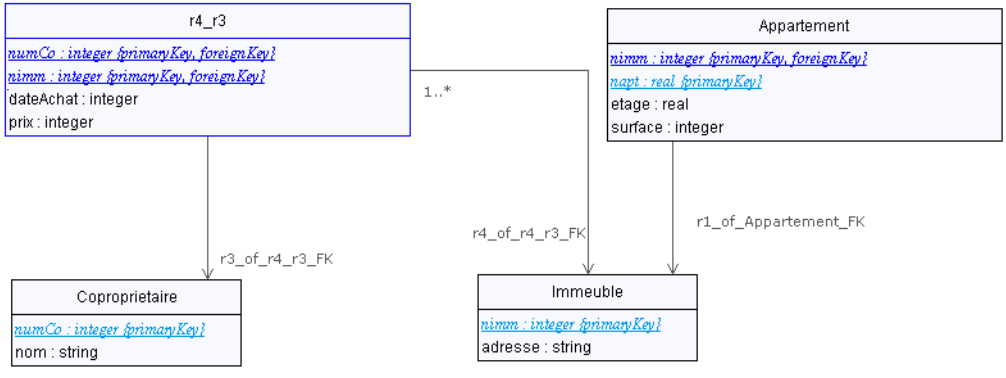


Figure 4-27 *Agrégations (Objecteering)*



Script SQL

Les scripts sont évalués sur la capacité d'inclure la directive CASCADE au niveau de la clé étrangère de la table composite et de celle liée à l'agrégation partagée (ici `Achat` et `Appartement`).

Bilan intermédiaire

Tableau 4.5 Agrégations

Agrégation	Niveau conceptuel avec UML (symboles de l'agrégation partagée et de la composition)	Modèle logique (clé composite)	Code SQL (contraintes CASCADE)
Enterprise Architect			
MagicDraw			
MEGA Designer			
ModelSphere			
MyEclipse			
Objecteering			
Poseidon			
PowerAMC			
Rational Rose Data Modeler			
Together			
Visio			
Visual Paradigm			

Tableau 4.5 Agrégations (suite)

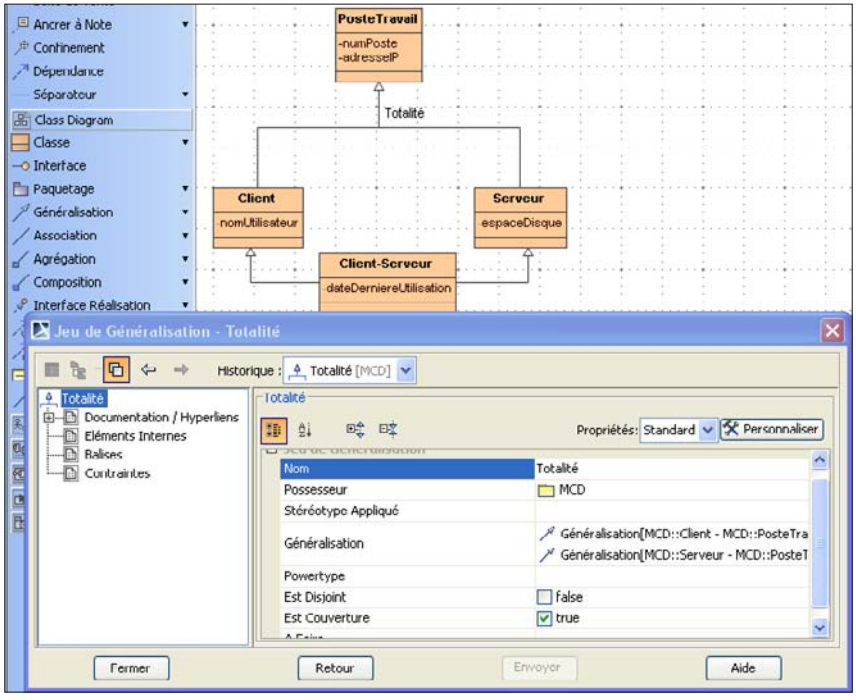
Agrégation	Niveau conceptuel avec UML (symboles de l'agrégation partagée et de la composition)	Modèle logique (clé composite)	Code SQL (contraintes CASCADE)
Visual UML			
Win'Design			

C'est encore une fois au niveau logique, durant la phase de transformation des agrégations (dont une couplée à une classe-association), que les disparités apparaissent. Seuls trois outils réussissent à construire un modèle logique correct : il s'agit de Objecteering, PowerAMC et Win'Design. Rational Rose et Visio ne proposent qu'une seule forme d'agrégation. En revanche, aucun outil ne génère de directive CASCADE au niveau du code SQL.

Héritage

L'exemple 4-28 décrit une hiérarchie à deux niveaux avec un héritage multiple. On devra pouvoir définir la contrainte de totalité au premier niveau.

Figure 4-28 *Héritage (Magic Draw)*



Niveau conceptuel

Les outils sont évalués sur la capacité de représenter d'une part la contrainte de totalité d'autre part l'héritage multiple.

Figure 4-29 *Héritage (MEGA)*

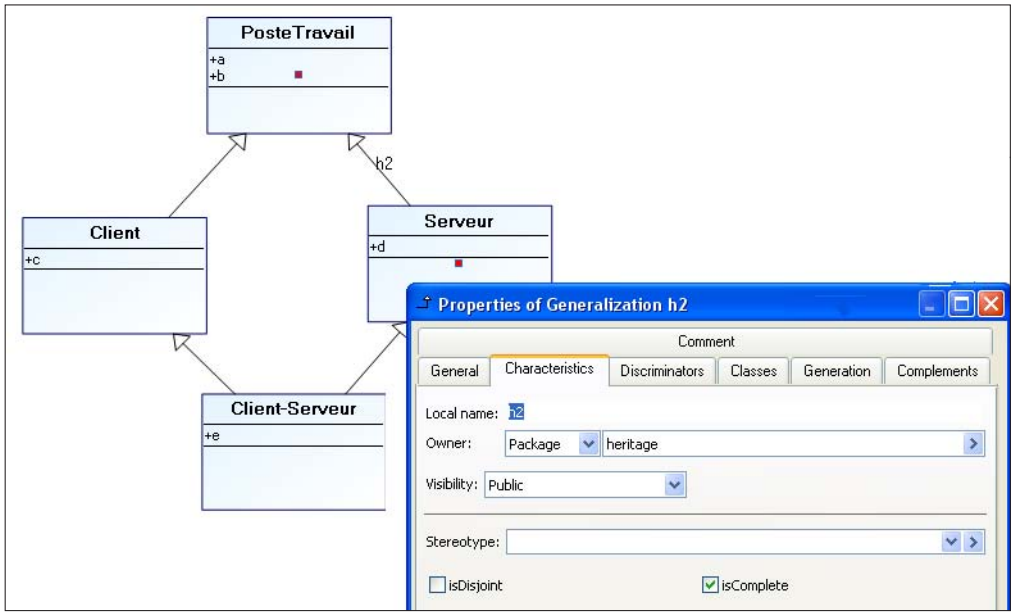
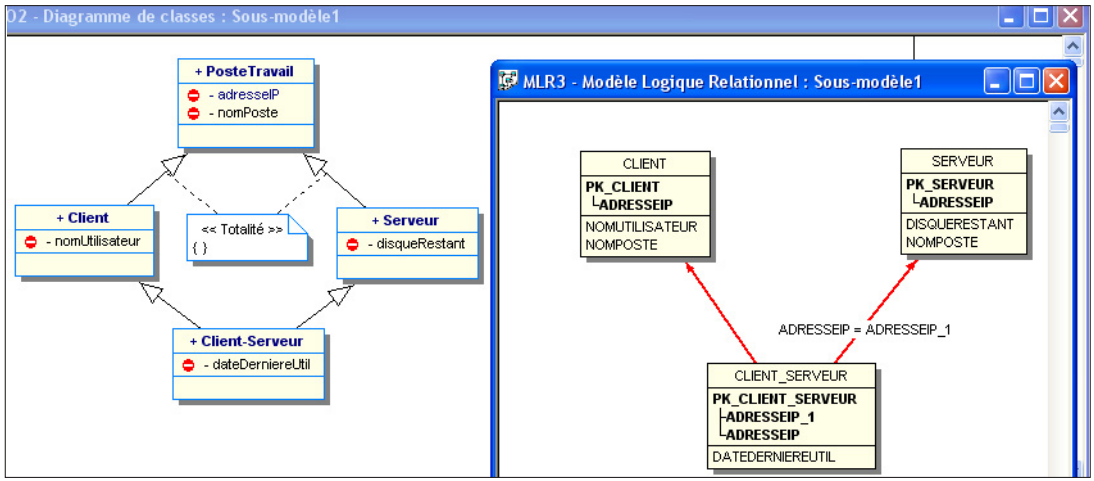


Figure 4-30 *Héritage (Win'Design)*



Niveau logique

Les outils sont évalués sur la capacité de proposer les trois cas de décompositions pour chaque niveau du graphe d'héritage. On devrait pouvoir choisir, par exemple, de définir une contrainte de totalité au premier niveau, de transformer par *push-down* le premier niveau et par distinction le second niveau.

Le modèle relationnel attendu est le suivant.

Figure 4-31 *Modèle relationnel attendu*

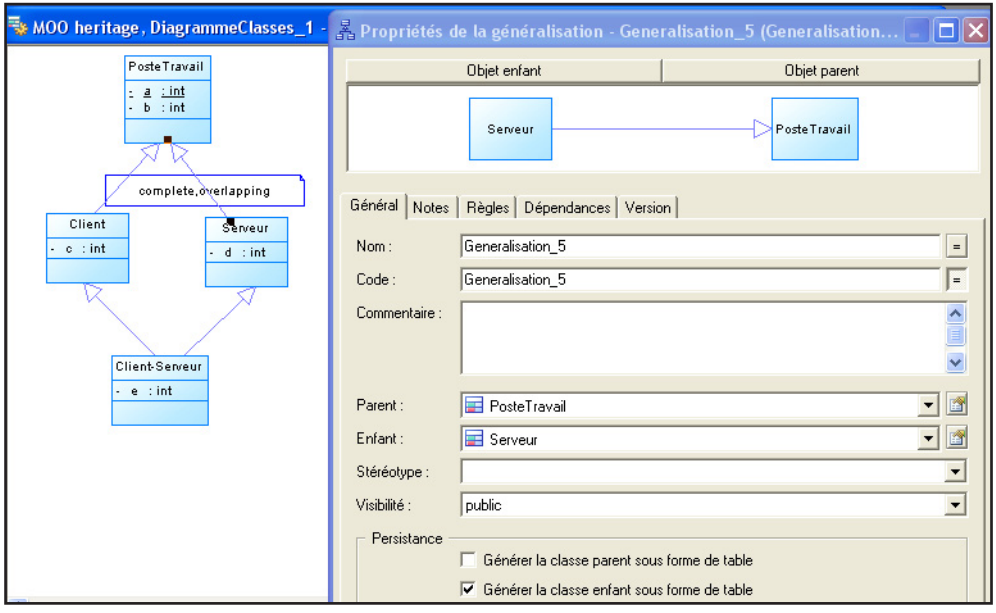
Client[adresseIP, nomPoste, nomUtilisateur]

Serveur[adresseIP, nomPoste, disqueRestant]

Client-Serveur[adresseIPC#, adresseIPS#, dateDerniereUtil]

Les captures d'écran suivantes illustrent les menus relatifs aux choix de décomposition du premier niveau du graphe d'héritage.

Figure 4-32 *Héritage (PowerAMC)*



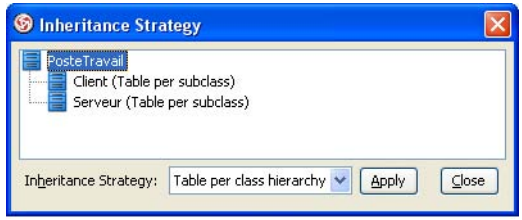


Figure 4-33 *Décomposition avec Visual Paradigm*

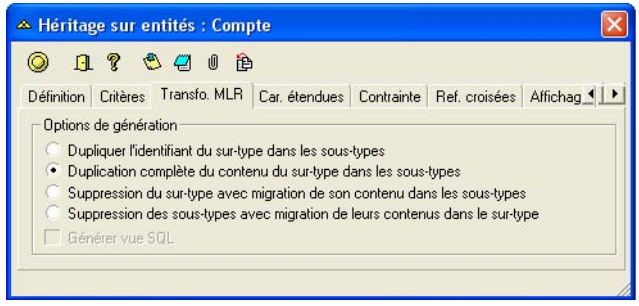
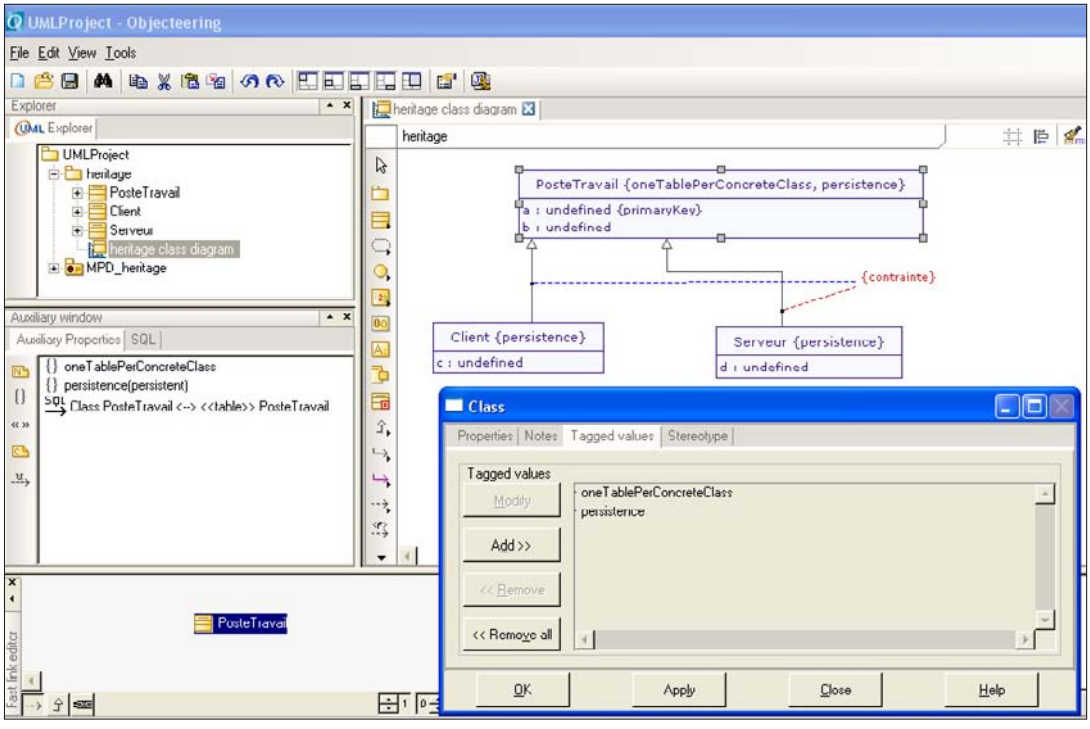


Figure 4-34 *Décomposition avec Win'Design*

Figure 4-35 *Décomposition avec Objecteering*



Script SQL

Aucun outil ne génère actuellement du code SQL implémentant les contraintes d'héritage (idéalement, il faudrait générer soit un déclencheur, soit des directives CHECK en fonction du type de la contrainte).

Bilan intermédiaire

Tableau 4.6 Héritage

Héritage	Niveau conceptuel avec UML (contrainte *complete-disjoint-incomplete-overlapping* et héritage multiple)	Modèle logique (3 cas de décomposition)
Enterprise Architect		
MagicDraw		
MEGA Designer		
ModelSphere		
MyEclipse		
Objecteering		
Poseidon		
PowerAMC		
Rational Rose Data Modeler		
Together		
Visio		

Tableau 4.6 Héritage (suite)

Héritage	Niveau conceptuel avec UML (contrainte *complete-disjoint-incomplete-overlapping* et héritage multiple)	Modèle logique (3 cas de décomposition)
Visual Paradigm		
Visual UML		
Win'Design		

Seuls Magic Draw et MEGA maîtrisent les contraintes prédéfinies de UML 2 relatives à l'héritage. Seuls PowerAMC et Win'Design (en passant par un MCD Merise) construisent un modèle logique correct en proposant les trois types de transformation. Visual Paradigm et Objecteering ne proposent que deux cas de décomposition.

La rétroconception

Les outils sont évalués sur la capacité à générer un diagramme UML à partir de la base de données MySQL suivante. Le processus devrait produire, en théorie, un diagramme contenant trois associations sur une classe-association voir figure 4-14).

```
CREATE TABLE vehicule (immat CHAR(10), genre CHAR(20),
       kilometrage INTEGER, CONSTRAINT pk_vehi PRIMARY KEY(immat));
CREATE TABLE agent (codea CHAR(10), noma CHAR(20), immat  CHAR(10),
       jour DATETIME, CONSTRAINT pk_agent PRIMARY KEY(codea));
CREATE TABLE mission (immat  CHAR(10), jour DATETIME,  km INTEGER,
       codea CHAR(10),CONSTRAINT pk_mision PRIMARY KEY(immat,jour),
        CONSTRAINT fk_mission_agent FOREIGN KEY(codea)
                   REFERENCES agent(codea),
         CONSTRAINT fk_mission_vehi FOREIGN KEY(immat)
                    REFERENCES vehicule(immat));
ALTER TABLE agent ADD CONSTRAINT fk_agent_mission
            FOREIGN KEY(immat,jour) REFERENCES mission(immat,jour);
CREATE TABLE passagers (immat  CHAR(10), jour DATETIME,
           codea CHAR(10), tempsPasse INTEGER,
           CONSTRAINT pk_passagers PRIMARY KEY(immat,jour,codea),
           CONSTRAINT fk_pax_mission FOREIGN KEY(immat,jour)
                       REFERENCES mission(immat,jour),
             CONSTRAINT fk_pax_agent FOREIGN KEY(codea)
                       REFERENCES agent(codea));
```

Les captures d'écran suivantes illustrent quelques résultats de rétroconception au niveau conceptuel. Bon nombre de solutions implémentent une composition. Ce faisant, une mission est identifiée par un numéro de véhicule et par un jour. Chaque mission pourra être reliée à un agent de trois manières distinctes.

Figure 4-36 *Rétroconception (Visual Paradigm)*

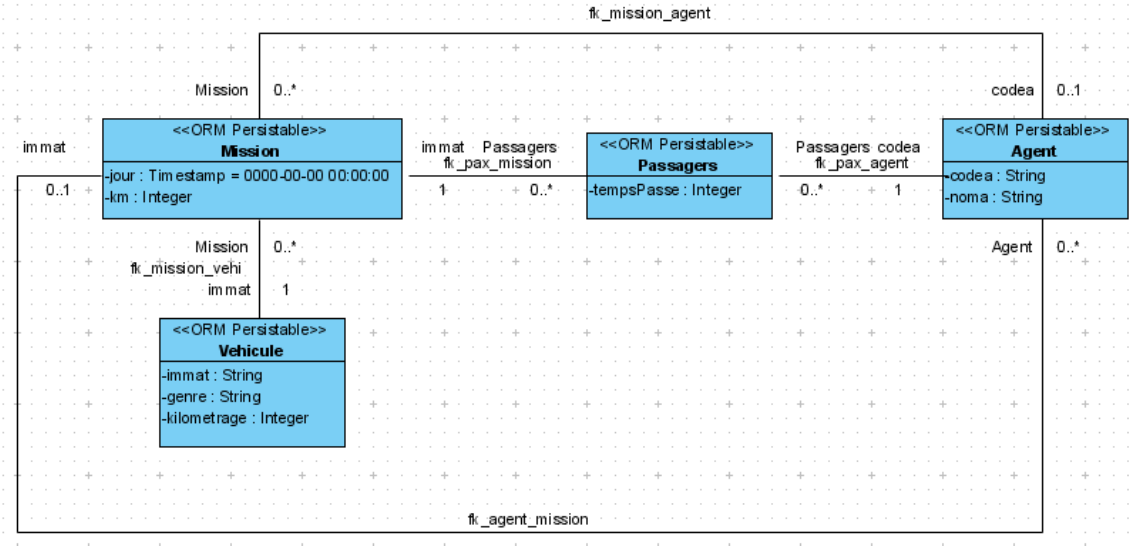


Figure 4-37 *Rétroconception (Rational Rose)*

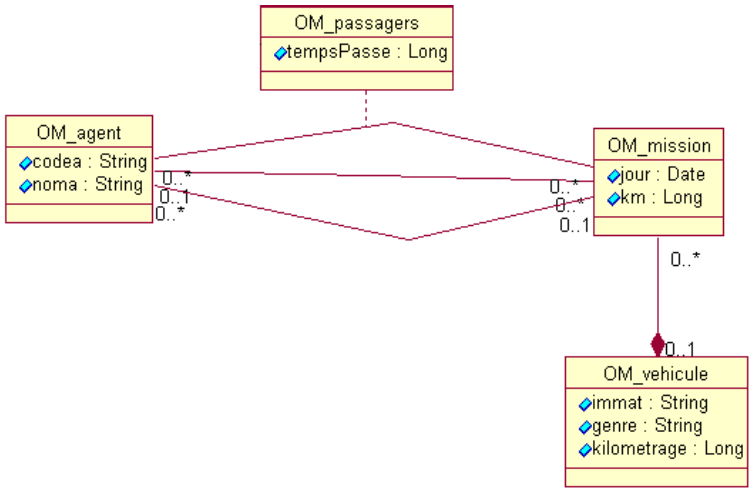


Figure 4-38 *Rétroconception (Win'Design)*

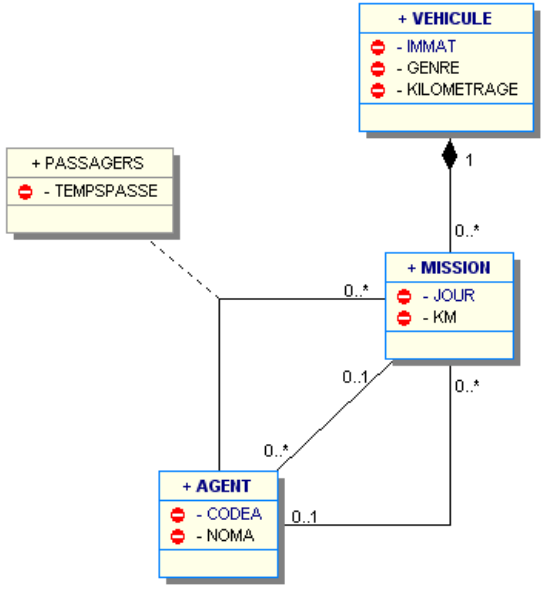


Figure 4-39 *Rétroconception (PowerAMC)*

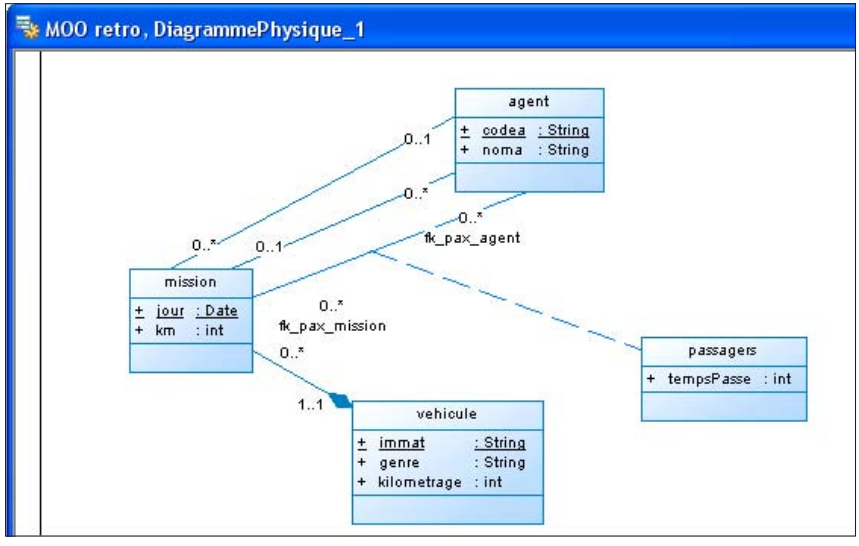
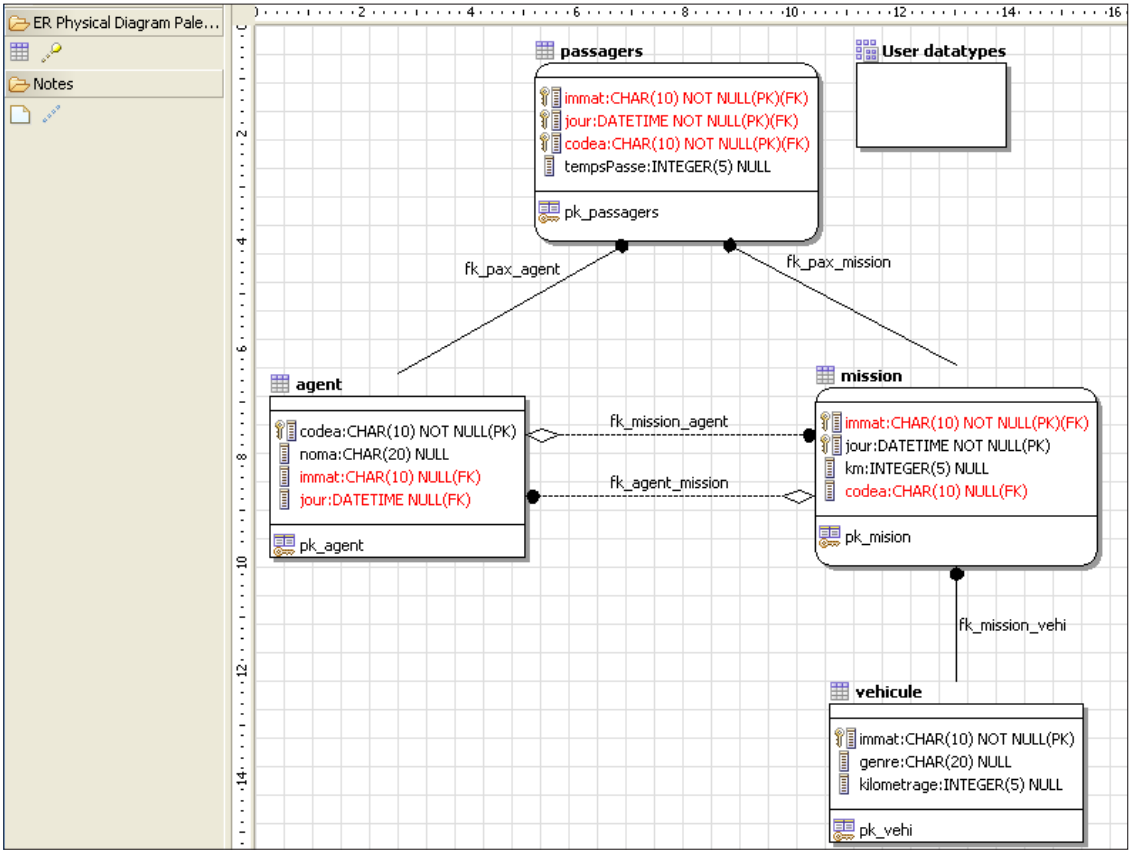


Figure 4-40 *Rétroconception (Together)*



Bilan intermédiaire

Seuls quatre outils sont capables de produire un diagramme UML correct suite à la rétro-conception de la base. La majorité des outils permettent une connexion via OBDC ou JDBC. Certains sont encore plus puissants car ils permettent d'analyser en plus d'une base, le script SQL de création des tables, index et contraintes (il s'agit de PowerAMC, Rational Rose, Together et de Win'Design).

Tableau 4.7 Rétroconception

Rétroconception	Sources de données (connexion base, script SQL)	Modèle logique (3 cas de décomposition)
Enterprise Architect		
MagicDraw		
MEGA Designer		
ModelSphere		
MyEclipse		
Objecteering		
Poseidon		
PowerAMC		
Rational Rose Data Modeler		
Together		
Visio		
Visual Paradigm		
Visual UML		
Win'Design		

Bilan général

La note finale est déterminée à partir des résultats des tests précédents, de la robustesse, de l'ergonomie, de la documentation et du support éventuel que j'ai dû solliciter.

Tableau 4.8 Produit

278	278	Prix indicatif pour 1 licence	Version évaluée
PowerAMC	★★★★☆	2 800 €	12
Win'Design	★★★★☆	1 800 €	7
Rational Rose Data Modeler	★★★½☆	3 500 €	7.0
Objecteering	★★★½☆	2 000 €	6
MagicDraw	★★½☆☆	4 250 €	12.0
Enterprise Architect	★★½☆☆	118 €	6.5
Visual Paradigm	★★½☆☆	540 €	3
MEGA Designer	★★½☆☆	NC	2005 SP3
Together	★★½☆☆	3 500 €	2006 R2
Visual UML	★★☆☆☆	230 €	5.0
MyEclipse	★½☆☆☆	45 € /an	5.1
Visio	★½☆☆☆	630 €	2007
Poseidon	★½☆☆☆	700 €	5.x
ModelSphere	★☆☆☆☆	1 160 €/an	2.5

Quelques mots sur les outils

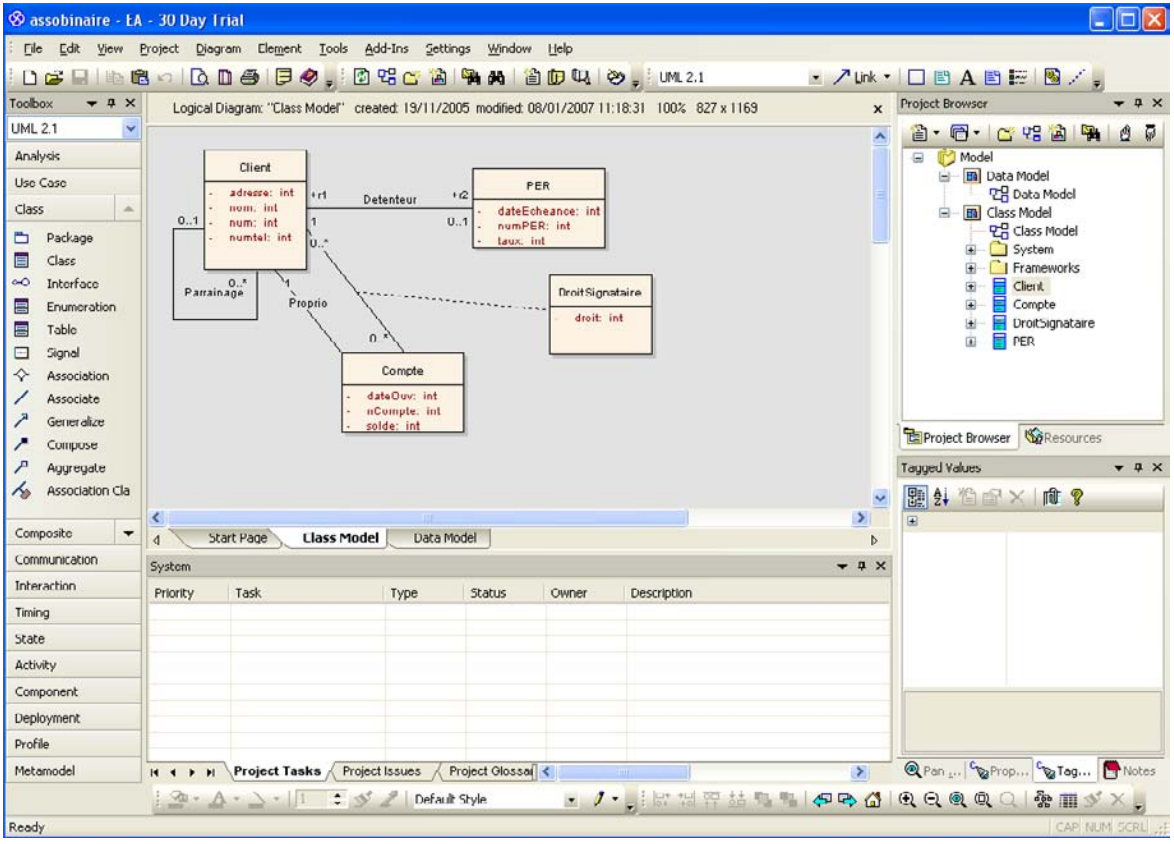
Si vous envisagez d'évaluer vous-même un des outils, ces quelques remarques vous feront sans doute gagner du temps lors de la mise en œuvre.

Enterprise Architect

Ergonomie soignée et robustesse pour cet outil Australien, qui pêche malheureusement dans les transformations de modèles comme nous le verrons par la suite. Créez un nouveau projet, puis choisissez les modules *Class* et *Database*. Sélectionnez `Data Architect` dans la fenêtre Toolbox pour disposer des icônes nécessaires à la construction de diagrammes de classes.

Ajoutez un nouveau diagramme dans la fenêtre de droite. Il n'est pas possible de définir un attribut identifiant par classe. Pour créer une classe-association (clic droit sur la classe `Advanced/Make Association Class`...). Si vous supprimez des éléments, pensez à vérifier dans la fenêtre du *browser* qu'ils sont bien effectivement retirés du modèle.

Figure 4-41 *Enterprise Architect*

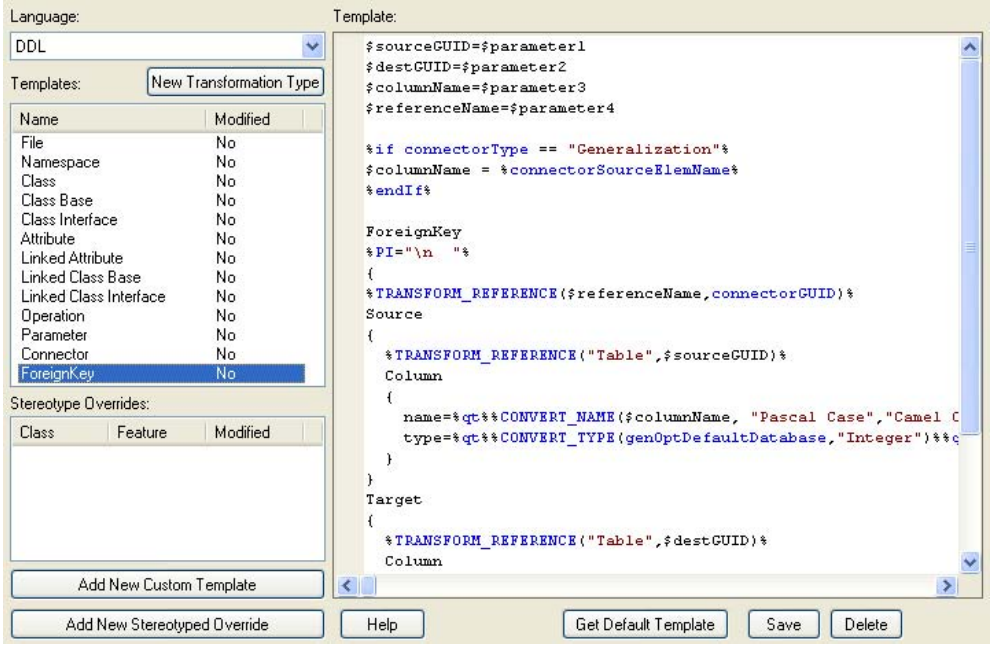


Pour générer un modèle logique exprimé dans le formalisme UML, utilisant le profil pour les bases de donnés (celui initié par Rational Rose), il faut d'abord le transformer en paquetage (`Project/Source Code Engineering/Generate Package Source Code`...) puis `Project/Model Transformation/Transform Current Package` (choisir DDL et le répertoire cible).

Pour générer un modèle physique suivez l'assistant qui apparaît après avoir sélectionné : `Project/Database Engineering/Generate Package DDL`...

En ce qui concerne les points forts de ce produit, citons sa documentation de bonne qualité avec un lien direct vers le forum Internet dédié, la possibilité de personnaliser un grand nombre de paramètres et processus et la facilité de travailler avec différents paquetages au sein d'un même diagramme de classes.

Figure 4-42 *Programmation du mapping*



Les points faibles principaux sont l'absence de transformation native entre modèles logiques et conceptuels et la faible qualité du mécanisme de transformation de modèles (génération très basique des clés étrangères). En effet, seules les associations binaires sont correctement traduites. Pour les associations *n*-aires, les classes-associations ou les agrégations, le concepteur doit enrichir le processus de transformation (reprogrammation du *mapping*). Chose compliquée et hasardeuse. L'héritage ne propose qu'un seul cas de décomposition. La rétroconception est correcte, seul un schéma logique au formalisme UML est produit (pas de remontée d'un modèle conceptuel).

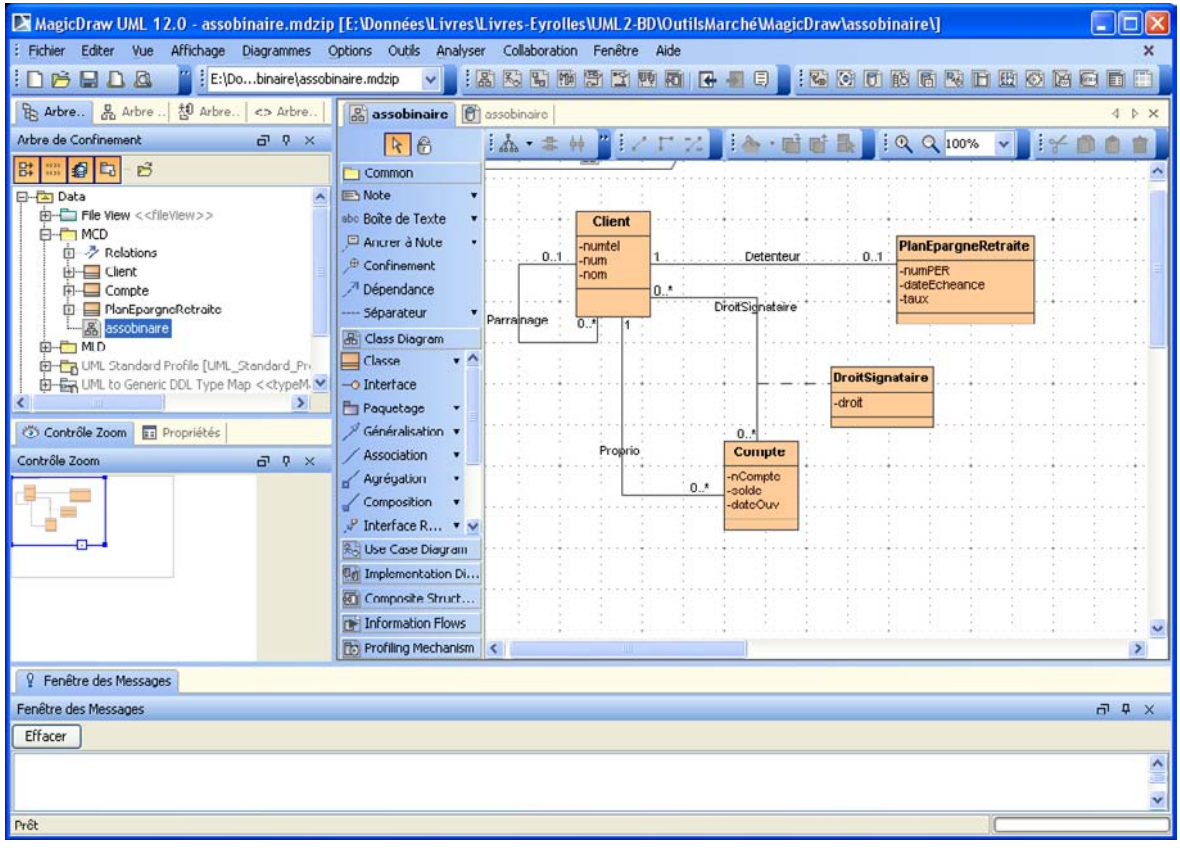
MagicDraw

Outil robuste à l'ergonomie semblable à celle de Visual Paradigm, MagicDraw de l'éditeur No Magic, Inc. se targue sur son site Web, par de nombreuses nominations, d'avoir été élu à plusieurs reprises meilleur outil de modélisation. Bien qu'il propose des assistants de

conversion puissants, son mécanisme de transformation de classes-associations n'est pas encore au point.

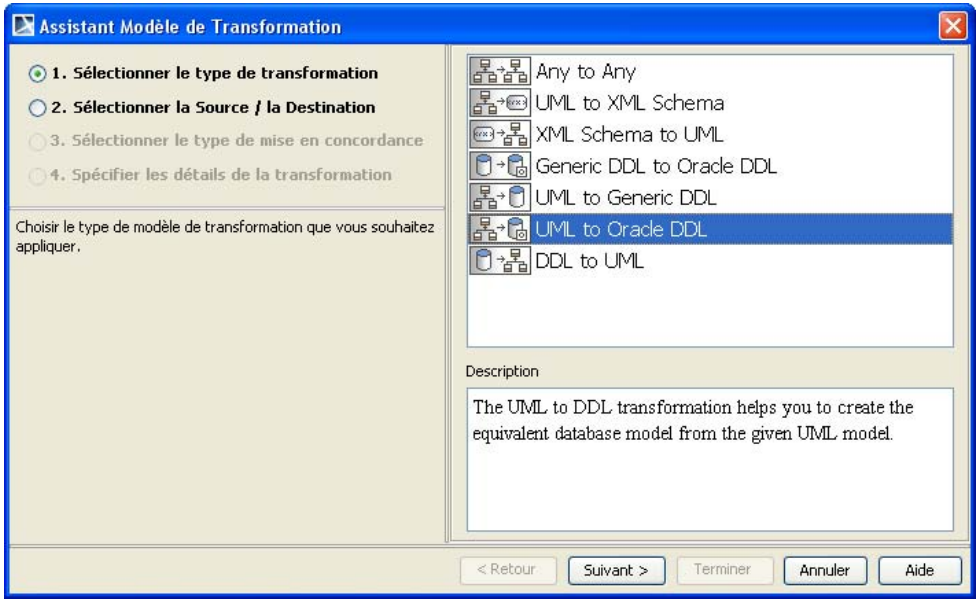
Créez un nouveau projet dans lequel vous installerez un nouveau paquetage contenant votre diagramme de classes. La saisie des éléments du diagramme de classes est intuitive mais il n'y a malheureusement pas de possibilité de définir d'identifant de classe. Pensez à créer un autre paquetage qui contiendra le modèle logique (modèle relationnel exprimé au profil UML).

Figure 4-43 *MagicDraw*



Pour tranformer les classes en relations, passer par `Outils/Transformer` qui lance un assistant. Pour définir une contrainte d'un graphe héritage, vous devez créer un ensemble de généralisations reliant les liens des sous-classes, donner un nom à votre contrainte et choisir parmi les options relatives à la disjonction et à la couverture.

Figure 4-44 *Assistant de transformation (Magic Draw)*



La génération du script SQL s'opère via le répertoire `Jeux de Code d'Ingéniérie` (`New`/`DDL`...) : créez une base puis faites glisser vos relations du paquetage du niveau logique. Ensuite, il suffit de séléctionner la base puis de lancer la généreration.

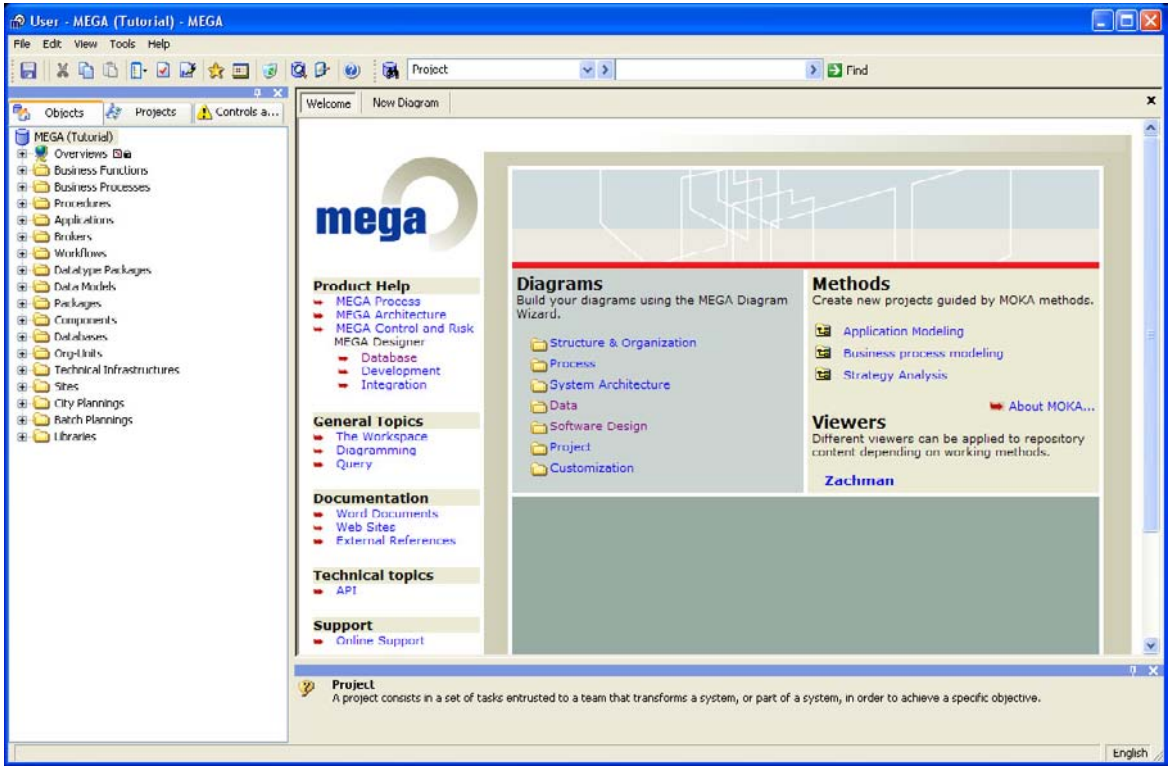
Pour la rétroconception, il faut créér une base dans ce même répertoire, puis `Editer`... lance un menu qui précise l'accès à la base (souvent une connexion JDBC). Ensuite, il suffit de lancer le processus par le menu `Inverser`. Le schéma relationnel obtenu par rétroconception est au profil UML. Un assistant vous permet de le convertir en diagramme (conceptuel) de classes UML (menu `Outils`/`Transformer`/`DDL to UML`). Pour visualiser les associations, positionnez-vous sur une des classes du schéma conceptuel (clic droit `Elements reliés`/`Afficher les éléments liés`). Le diagramme risque fort d'être de qualité moyenne si votre base de donnés implémente des associations plusieurs-à-plusieurs ou d'agrégation que vous vous attendiez légitimement à visuliser en tant que classes-associations.

Les points forts résident dans la possibilité de transformation d'une base à une autre, dans la prise en compte des contraintes d'un graphe d'héritage, dans le choix varié des pilotes de SGBD et des animations des fonctionnalités téléchargeables sur le site. Les limitations concernant UML sont l'absence d'identifiant de classe et la transformation des classes-associations (qui n'est pas du tout prise en compte). De plus, aucun cas de décomposition n'est prévu pour l'héritage, l'éditeur semble attendre la tendance du marché pour se décider à implémenter une solution.

MEGA Designer

Faisant partie des solutions relatives à l'architecture et à la gouvernance du système d'information de la société Française MEGA International, l'outil Designer se consacre aux différents modèles de données servant à la conception.

Figure 4-45 *MEGA Designer*



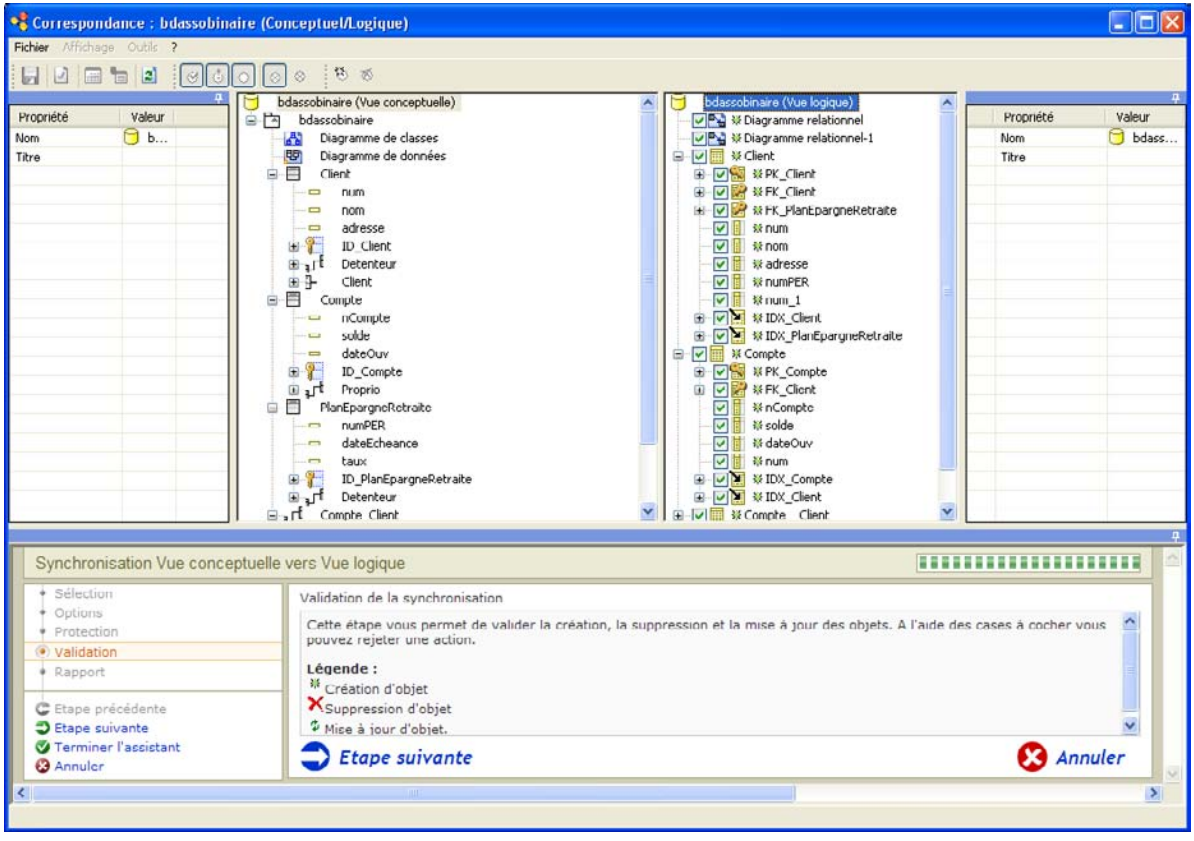
Pour débuter, créez une base de données (dans le répertoire `Databases`) dans laquelle vous définirez un nouveau modèle de données (clic droit `New`...). La création de ce modèle de données va entraîner automatiquement la création d'un paquetage (portant le nom de la base). Dans le répertoires `Paquetages`, sélectionnez ce paquetage et créez un diagramme de classes (`New/Generate a Class Diagram`).

Pour créér une classe-association, ne définissez pas de classe mais cliquez sur l'association puis `Association class`... Au même titre que PowerAMC et Win'Design, MEGA permet l'identification relative.

Afin de générer un modèle logique, positionnez-vous sur votre base dans le répertoire `Databases` et choisissez `Editer` puis `Outils` (synchroniser du conceptuel au logique). Un assistant en quatre étapes est alors lancé. Le script SQL est créé via l'option `Générer`.

Pour la rétroconception, créez une base puis `Reverse Database`, sélectionnez l'entrée ODBC. Il faudra définir un diagramme (`New/Diagram/Relational Diagram`) pour faire un glisser-déposer des tables obtenues. Pour obtenir l'équivalent UML du modèle, il faut synchroniser (clic droit sur la base puis `Edit/Tools/Synchronisation`...). Le schéma obtenu par rétroconception peut être de qualité moyenne. À noter enfin l'existence d'un modèle de données hybride entre UML et le niveau logique relationnel appelé *Data diagram*.

Figure 4-46 *Synchronisation (MEGA)*



La principale limitation concernant UML est l'absence d'association *n*-aire et la qualité de transformation des classes-associations et de l'héritage multiple. Par ailleurs, aucun pilote de SGBD *Open Source* n'est pris en compte dans le processus de conception.

Les points forts sont la robustesse de son ergonomie, son outil de recherche d'éléments inter-objets (processus, paquetages, bases de données, etc.), son outil de synchronisation par étapes guidées par un assistant et les passerelles possibles avec PowerAMC et Rational Rose.

ModelSphere

Outil de la société québecoise Grandite, ModelSphere permet de créer des modèles conceptuels de type entité-association, des modèles relationnels (associés ou non à une base) selon différents formalismes et notamment UML. La partie UML de l'outil permet de créér un diagramme de classes. Le processus de transformation vers le niveau logique n'est pas finalisé notamment au niveau des identifiants et des clés étrangères car des modèles de liens (pas encore opérationnels) doivent être mis en œuvre. Le niveau logique est donc composé de tables sans clés liées entre elles graphiquement.

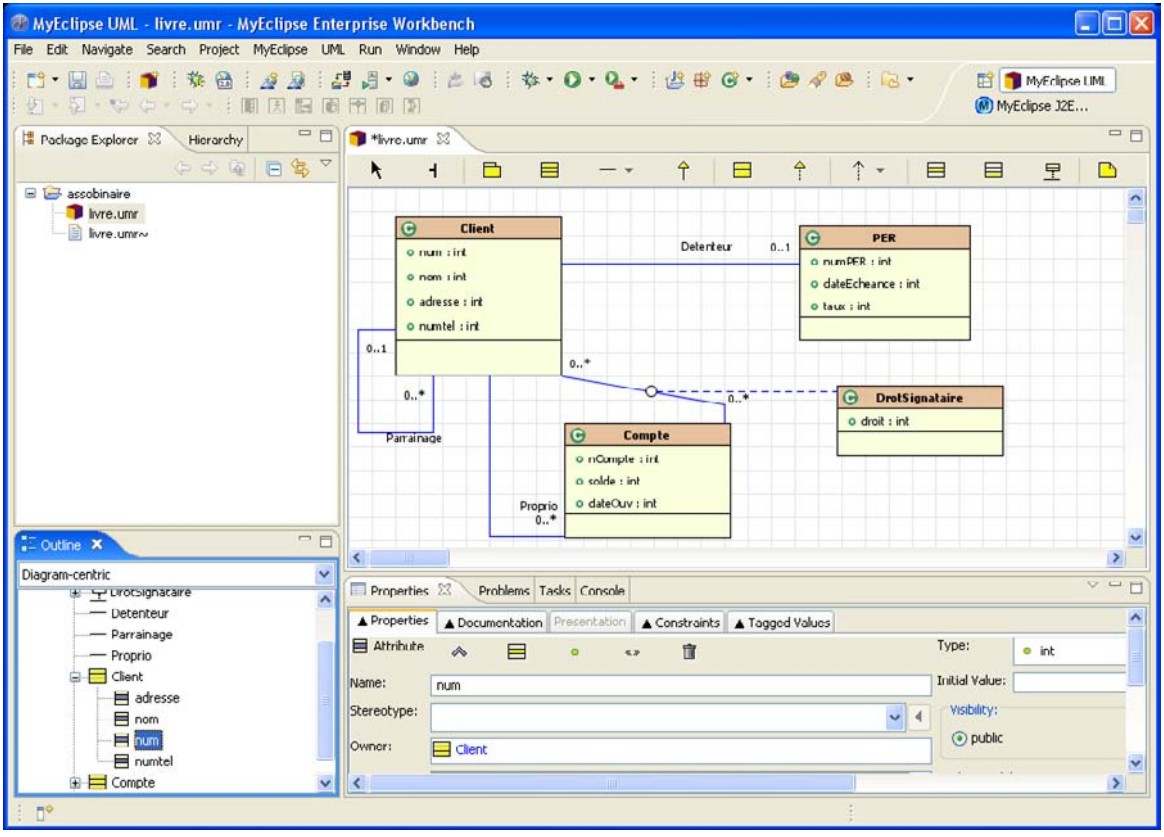
Pour générer un modèle logique, sélectionnez le modèle de classes (clic droit `Générer Modèle de données`...). Par la suite, positionnez-vous dans le répertoire contenant les tables générées et ajoutez un diagramme (clic droit `Ajouter`). Faites glisser les tables et les associations générées pour découvrir le schéma relationnel généré.

Les principales limitations concernant UML sont l'absence d'identifiant de classe, de notation graphique pour les classes-associations et les associations *n*-aires. Le processus de rétro-conception d'une base est correct (via ODBC ou JDBC) mais le schéma généré UML est de qualité moyenne puisque le processus traduit toute table en une classe. Pas de moyen pour l'heure de traduire un MCD en diagramme de classes et inversement.

MyEclipse

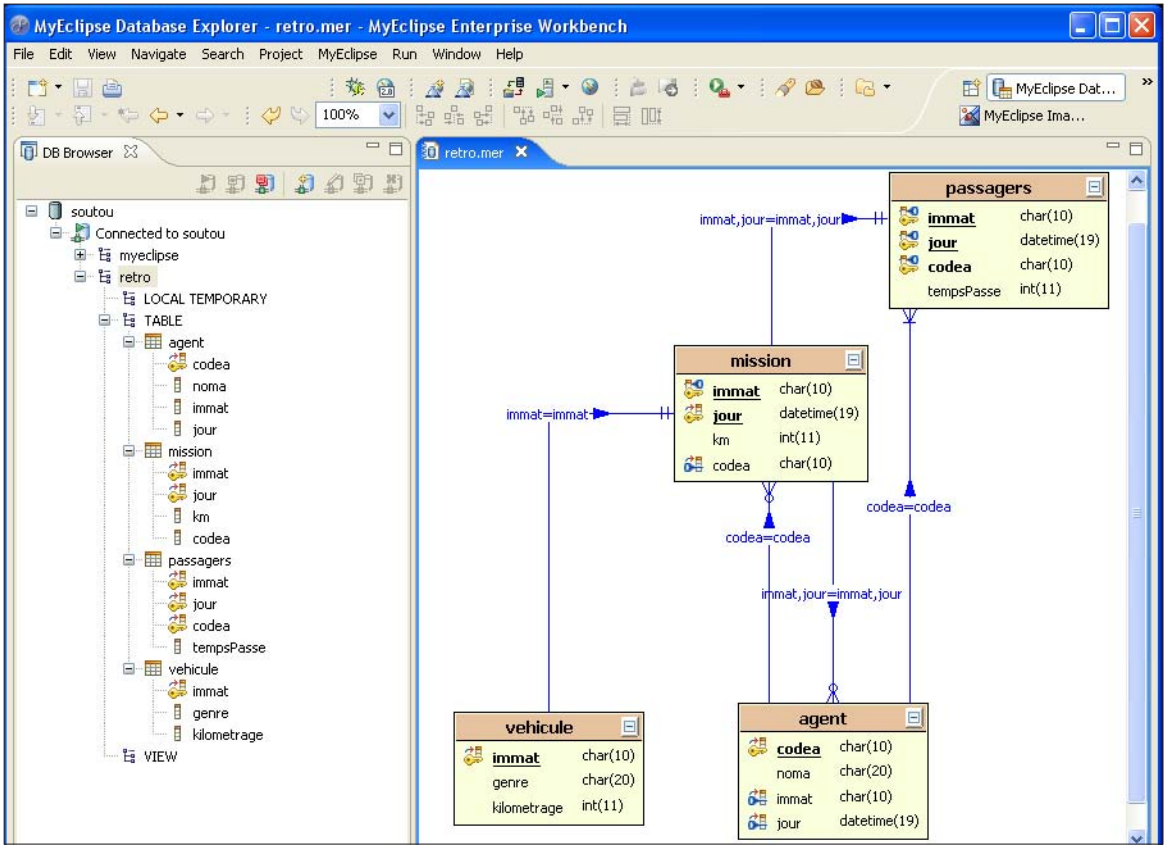
Comme Poseidon, cet outil n'est pas vraiment fait pour les bases de données (niveau logique inexistant de même que la génération de code SQL à partir d'un diagramme UML). L'ergonomie est toutefois mieux réussie que Poseidon. Cet outil oblige l'utilisateur à se servir du modèle *Entity-Relationship* pour concevoir ses schémas conceptuels.

Figure 4-47 *MyEclipse*



Les principales limitations côté UML se rapportent à l'absence d'identifiant de classe et de notation pour les associations *n*-aires. Le processus de rétroconception d'une base est correct (grand choix de SGBD) mais le schéma généré est de type *Entity-Relationship* (notation *crow's foot)*. Pour vous faire une idée de ces possibilités `Preferences/MyEclipse/Database Explorer`, configurez votre accès à la base puis créez une connexion `Window/Open Perspective/Other/MyEclipse/Database Explorer`. Créez ensuite un diagramme (clic droit `New ER Diagram`), enfin sélectionnez les tables à rétroconcevoir.

Figure 4-48 *Niveau logique (MyEclipse)*



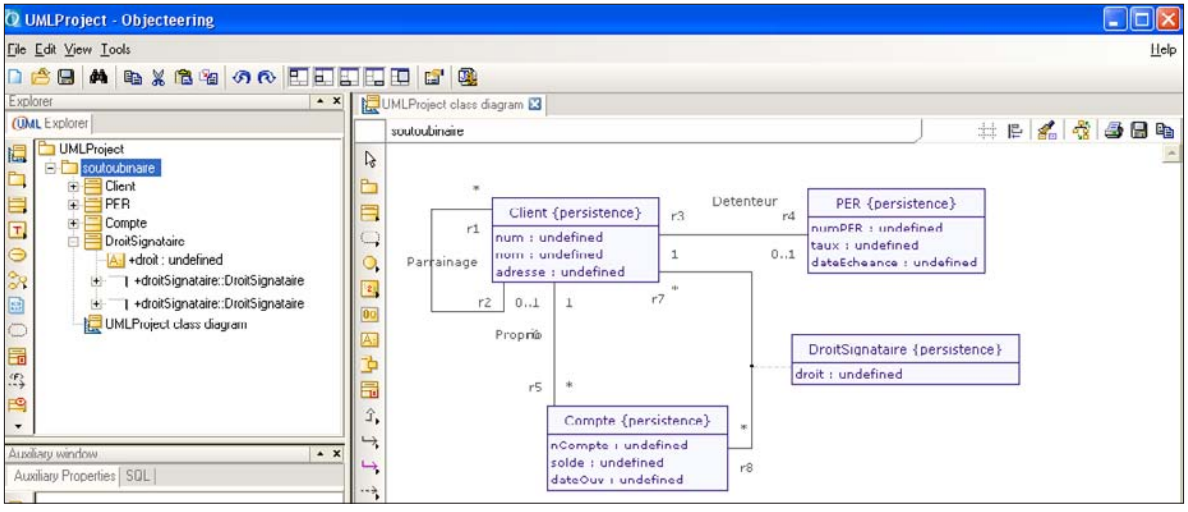
Objecteering

Objecteering est un outil français de la société éponyme, filiale de SOFTEAM, qui a été acteur majeur dans la comunauté des technologies objet et premier membre Européen de l'OMG. Dans ce logiciel, le terme *physical model* désigne un schéma relationnel (niveau logique) et *logical model* désigne un script SQL.

Avant tout, créez un projet puis configurez l'affichage des tags qui serviront à annoter un diagramme de classes pour sa transformation (`Tools/Diagram graphic options/Properties Class Diagram`, rendre visible les *tagged values*). Ensuite déployez le module SQL (`Tools/Deploy an MDAC`... choisir `SQLDesigner`). Concernant ce module (`Tools/MDAC options`...), configurez l'accès à votre base. Dans l'option `Diagram generation`... cochez la case `Generates diagram`...

Vous pouvez créer un paquetage (dans la racine) qui contiendra votre diagramme de classes. La transformation nécessite d'enrichir le diagramme UML de *tagged values*. Ainsi vous devrez annoter chaque classe du tag `persistence` (avec la propriété `persistent`). Les identifiants sont choisis via un clic droit sur la classe `MDA Components/SQL Designer/Primary key`... Pendant la saisie du diagramme de classes servez-vous plutôt de la fenêtre de gauche (*UML Explorer*) pour supprimer les éléments superflus.

Figure 4-49 *Objecteering*



La génération d'un schéma relationnel se fait au niveau du paquetage contenant le diagramme UML (clic droit `MDA Components/SQL Designer/Generate physical model`). La génération d'un script SQL se fait au niveau du paquetage contenant le modèle physique (clic droit `MDA Components/SQL Designer/Generate SQL files`).

Un grand nombre de points forts sont à noter : rigueur de la démarche et ergonomie à la fois sobre et puissante, documentation détaillée du module SQL Designer (principes de transformations), réactivité et efficacité du support ainsi que l'existence d'un forum dédié.

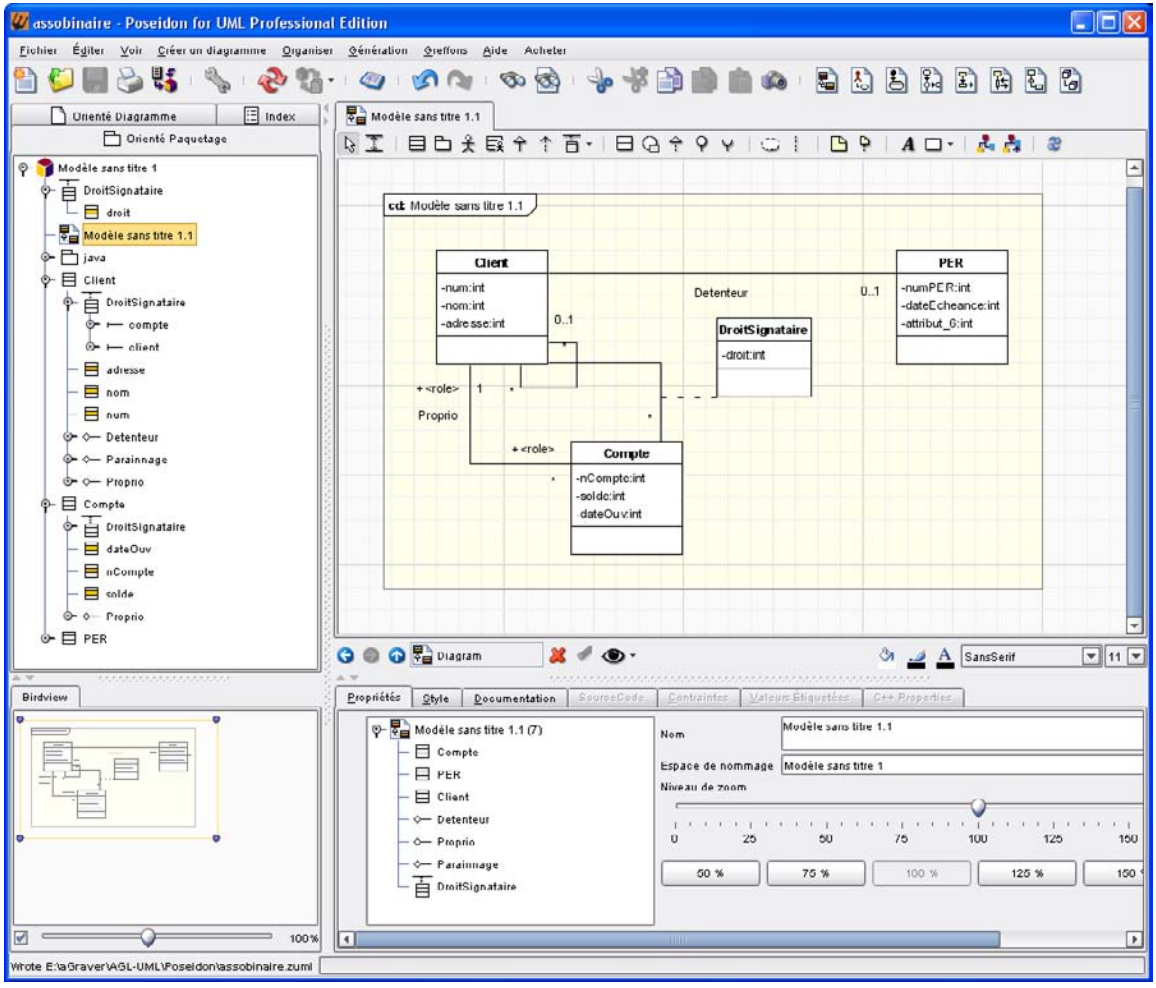
Les points faibles concernent le faible nombre de SGBD pris en compte (notamment SQL92 et tous les *Open Source*), l'absence de processus de rétroconception et la non-prise en compte de l'héritage multiple. Enfin, certaines transformations de diagrammes ont posé des problèmes qui sont désormais identifiés et en cours de correction.

Poseidon

Cet outil n'est pas vraiment fait pour les bases de données. Le niveau logique n'est pas supporté, la génération de code SQL est très contestable (les classes-associations ne sont pas traduites et les associations sont parfois inexistantes). L'ergonomie surprend au début mais on

s'y fait. La création de classes-associations et d'associations réflexives n'est pas aisée. Les options ne sont pas toutes traduites en français, les associations *n*-aires ne sont pas prises en compte, il n'existe pas de processus de rétroconception d'une base, n'en jetez plus !

Figure 4-50 *Poseidon*

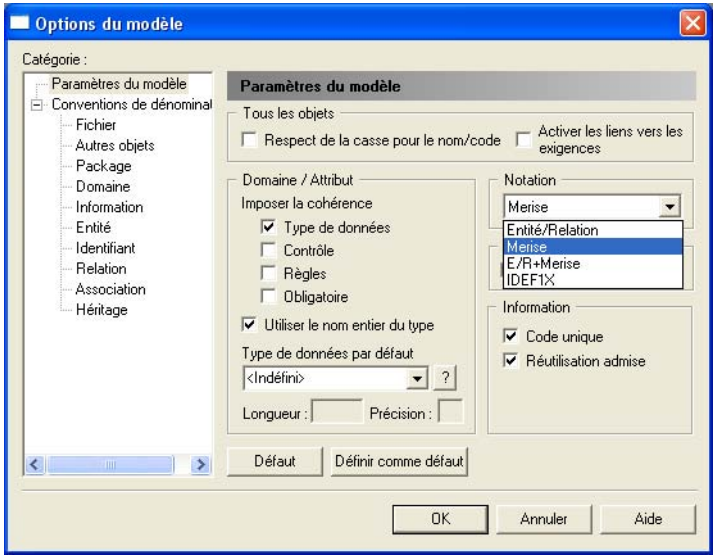


PowerAMC

PowerAMC (anciennement AMC*Designor) est la version française de l'outil de modélisation PowerDesigner de Sybase. Concernant les bases de données, l'outil prend en charge trois types de modèles qu'on peut transformer entre eux :

- le modèle conceptuel de données (MCD) qu'on pourra construire avec la notation Merise, entité-relation ou IDEF1X ;
- le modèle physique de données (MPD) qui correspond au niveau logique ;
- le modèle orienté objet (MOO) au formalisme UML.

Figure 4-51 *Formalismes du MCD (PowerAMC)*



L'ergonomie de l'outil est très réussie : il est très intuitif de créér différents diagrammes, de les transformer entre eux, de générer le script SQL ou de rétroconcevoir une base de données (à partir des tables ou d'un script).

Les transformations de modèles se font très facilement en ouvrant le diagramme puis `Outils/Générer un Modèle...`). Pour manipuler une base de données (génération ou rétro-conception), il faut travailler avec un modèle physique pour que l'option `SGBD` soit active.

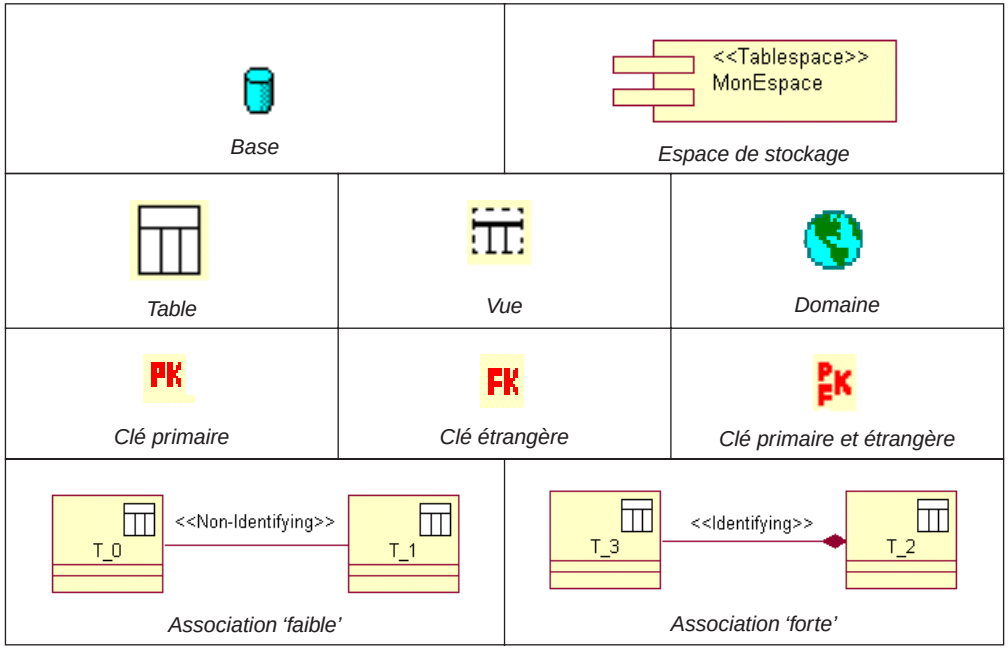
Quelques points forts sont à mettre en exergue : un grand choix de SGBD est pris en compte, la robustesse et la qualité de l'ergonomie qui permet de travailler facilement avec différents diagrammes dans le même environnement. Les seuls points faibles se réduisent à la non-prise en compte des contraintes et des associations *n*-aires avec la notation UML.

Rational Rose Data Modeler

À l'origine de la standardisation d'UML, Rational a proposé un profil convenant aux bases de données. Un profil, proposition d'une communauté, regroupe un ensemble d'éléments (composants, stéréotypes, icônes, propriétés…) qui s'appliquent à un contexte particulier tout

en conservant le métamodèle d'UML intact. Le profil UML, pour la modélisation de données *(UML profile for Data Modeling)*, permet de manipuler des bases de données à l'aide notamment de classes UML possédant des stéréotypes prédéfinis (`Table`, `RelationalTable`, `View`…). Les icônes de ce profil sont illustrées figure 4-52.

Figure 4-52 *Icônes du profil UML*



Il est nécessaire dans un premier temps de créer un composant `Database` dans le répertoire `Component View` (clic droit sur un répertoire, `Data Modeler/New/Database`). On donne ensuite un nom à la base et on choisit la nature du SGBD utilisé en cible (`Name` et `Target`). Il faut ensuite créer un schéma dans le répertoire `Logical View` (clic droit sur le répertoire, option `Data Modeler/New/Schema`). Associez ensuite ce schéma à la base de données cible précédemment créée.

Le diagramme de classes UML doit être situé dans un paquetage (clic droit sur le compartiment `Logical View`, `New/Package`). Les classes doivent être marquées `Persistent` (clic droit, `Open Specification`, onglet `Detail`). Tous les éléments du modèle objet ne se transforment pas forcément en classes de stéréotype `<<Table>>` au niveau logique. La documentation stipule qu'il est prudent d'examiner en détail le résultat de la transformation de manière à s'assurer de la bonne structure de la base. On retrouve ici l'idée de départ de l'ouvrage, à savoir la maîtrise des concepts pour une meilleure utilisation de l'outil.

La transformation du modèle objet se fait en sélectionnant le paquetage des classes UML à transformer (clic droit puis `Data Modeler/Transform to Data Model...`). Pour visualiser le modèle logique créez dans le schéma un diagramme (`Data Modeler/New/Data Model Diagram`) puis faites glisser chaque relation obtenue par la transformation.

Le processus de rétroconception démarre du schéma (compartiment `Logical View/Schemas`) qu'on sélectionne par un clic droit (`Data Modeler/Transform to Object Model`). Un schéma se transforme en un paquetage à créer initialement. Un assistant permet de sélectionner le nom du paquetage en sortie, et si oui ou non, les clés primaires doivent être transformées en identifiant de classe. Le fait de transformer un modèle de données dans un paquetage modifie les nouveaux éléments du paquetage mais ne détruit ni ne modifie les éléments mis à jour au niveau du modèle de données. Si cette option du logiciel fonctionne bien pour les associations binaires, classes-associations et l'héritage, elle n'est pas opérationnelle pour les associations *n*-aires.

Bien que peu de modifications ont été apportées à cet outil depuis la première version de cet ouvrage, le point fort du logiciel est sa très grande robustesse. On peut toutefois regretter qu'il n'inclue pas encore de pilotes pour les SGBD *Open Source* et qu'il ne propose pas l'agrégation simple et les associations *n*-aires.

Together

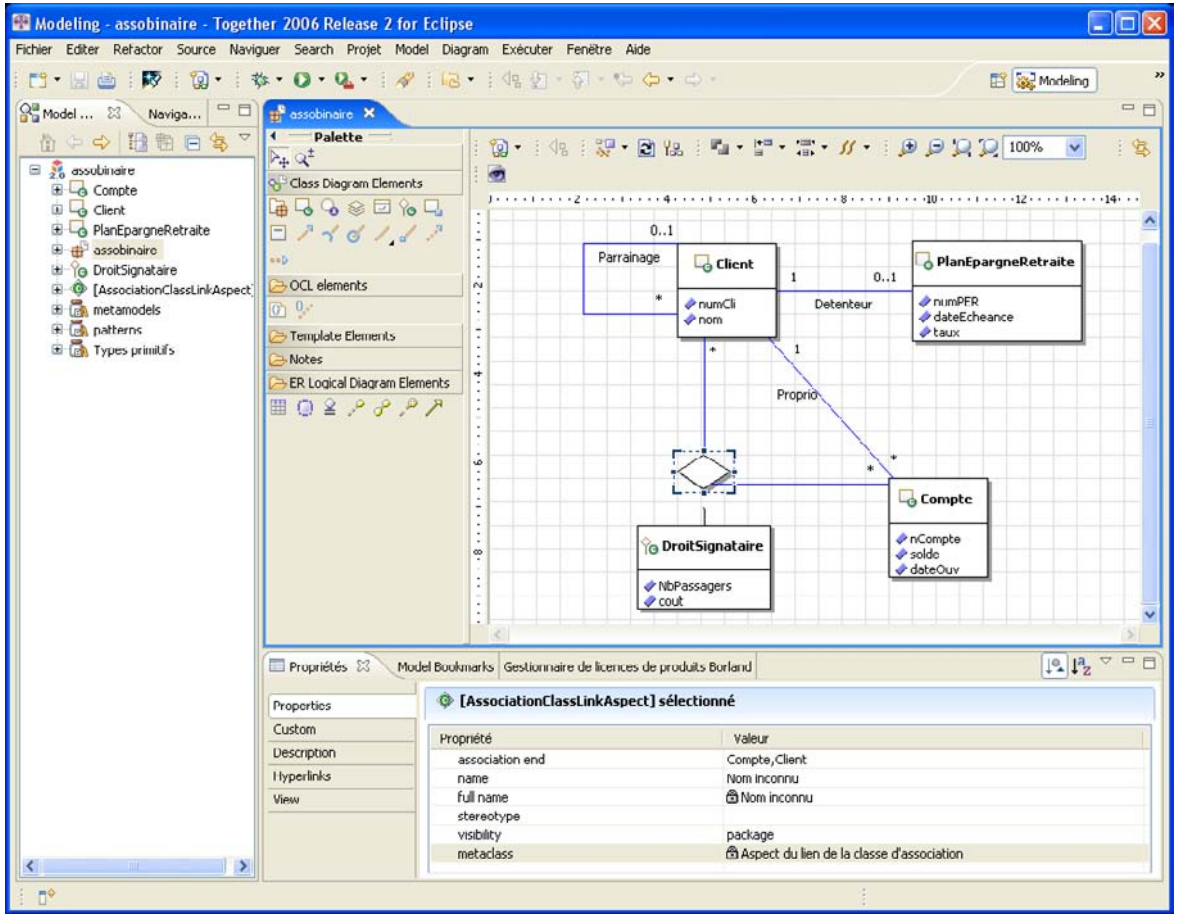
Together est l'outil de conception de Borland. La partie relative aux bases de données est nommée *Data Modeling* et est présente dans la version pour Eclipse. Une autre version existe pour VisualStudio mais elle ne prend pas en compte l'aspect modélisation de bases de données.

Dans la partie *Data Modeling*, seuls les niveaux logique et physique sont présents. Le formalisme des modèles logiques s'apparente au profil UML pour les bases de données. Dans ce modèle, deux types d'associations sont prévues : celles qui doivent donner lieu à la création d'une table (*many-to-many relationship*) et celles qui génèrent seulement une clé étrangère (*relationship*). Pour ces dernières, l'entité cible du lien créé graphiquement détermine l'entité parent. Les modèles physiques font apparaître les clés étrangères sous la forme graphique (formalisme IDEF1X). La transformation entre les deux modèles se fait par la fonction `Importer/DB schema from ER Logical Diagram`. La génération de script SQL s'opère par la fonction `Exporter/DDL SQL script`.

La création d'un diagramme de classes nécessite de créer un projet (`Nouveau/Autres/ULML 2.0 Project`). Une palette de symboles est alors mise à disposition. Pour créer une classe-association (ou une association *n*-aire), sélectionnez l'icône `Association Class`. Ensuite, reliez le symbole losange (*diamond*) aux différentes classes avec le lien `Association End`.

La transformation automatique du diagramme de classes en modèle logique n'est pas nativement proposée. On ne peut avoir accès à la base de données que par le modèle logique et UML n'est pas encore exploité au niveau conceptuel. Pour ce faire, il faudra programmer des

Figure 4-53 *Together*



transformateurs de modèles (similaires aux règles de mapping de Enterprise Architect). La documentation fait toutefois preuve d'optimisme "*The concept of entities and relationships in logical data modeling maps to the concept of classes and associations in the UML 2.0 class diagram*". Il n'empêche que la programmation des règles de transformation des classes-association, associations *n*-aires et l'héritage par les éléments du modèle logique profilé par Borland risque de ne pas être de tout repos.

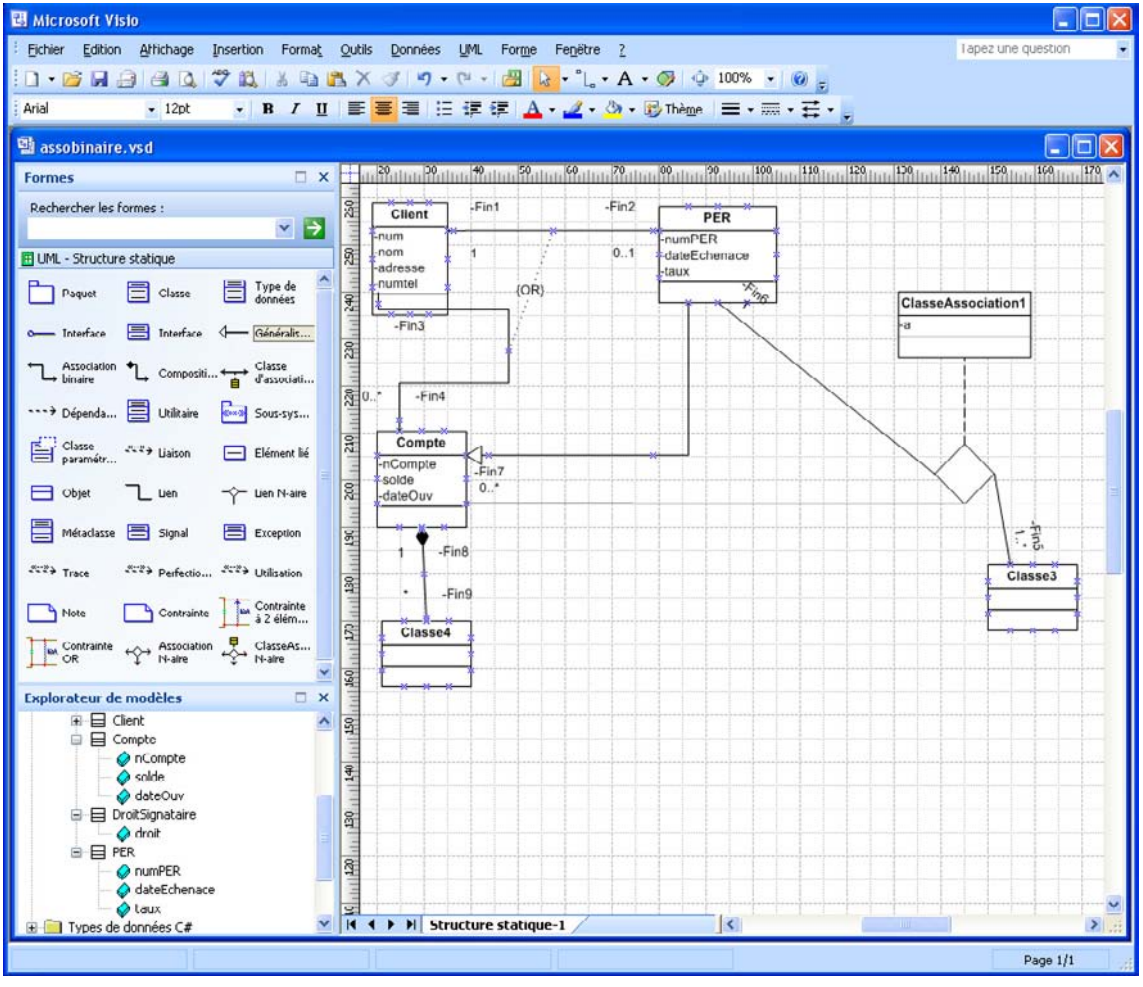
La rétroconception consiste à importer (`Fichier/Importer`, la source étant soit un script, soit une connexion JDBC à une base qu'il faudra paramétrer). Le schéma généré est un modèle physique dont le formalisme s'apparente au profil UML et qui fait apparaître explicitement les clés étrangères.

Les points forts de l'outil : sa qualité et sa robustesse, l'efficacité du support et l'expertise dans les fonctionnalités de transformation de modèles par l'utilisation de QVT et de EMF (Eclipse Metamodel Framework). Le point faible majeur concerne l'absence de processus natif de transformation entre schémas conceptuels et modèles logiques.

Visio

Visio fait partie de la suite Office de Microsoft. En utilisant la notation UML, il n'est pas possible de générer de modèle ni de code. Pour ce faire, vous devrez travailler soit avec le modèle ORM (modèle conceptuel binaire graphique), soit avec IDEF1X (modèle relationnel

Figure 4-54 *Visio*



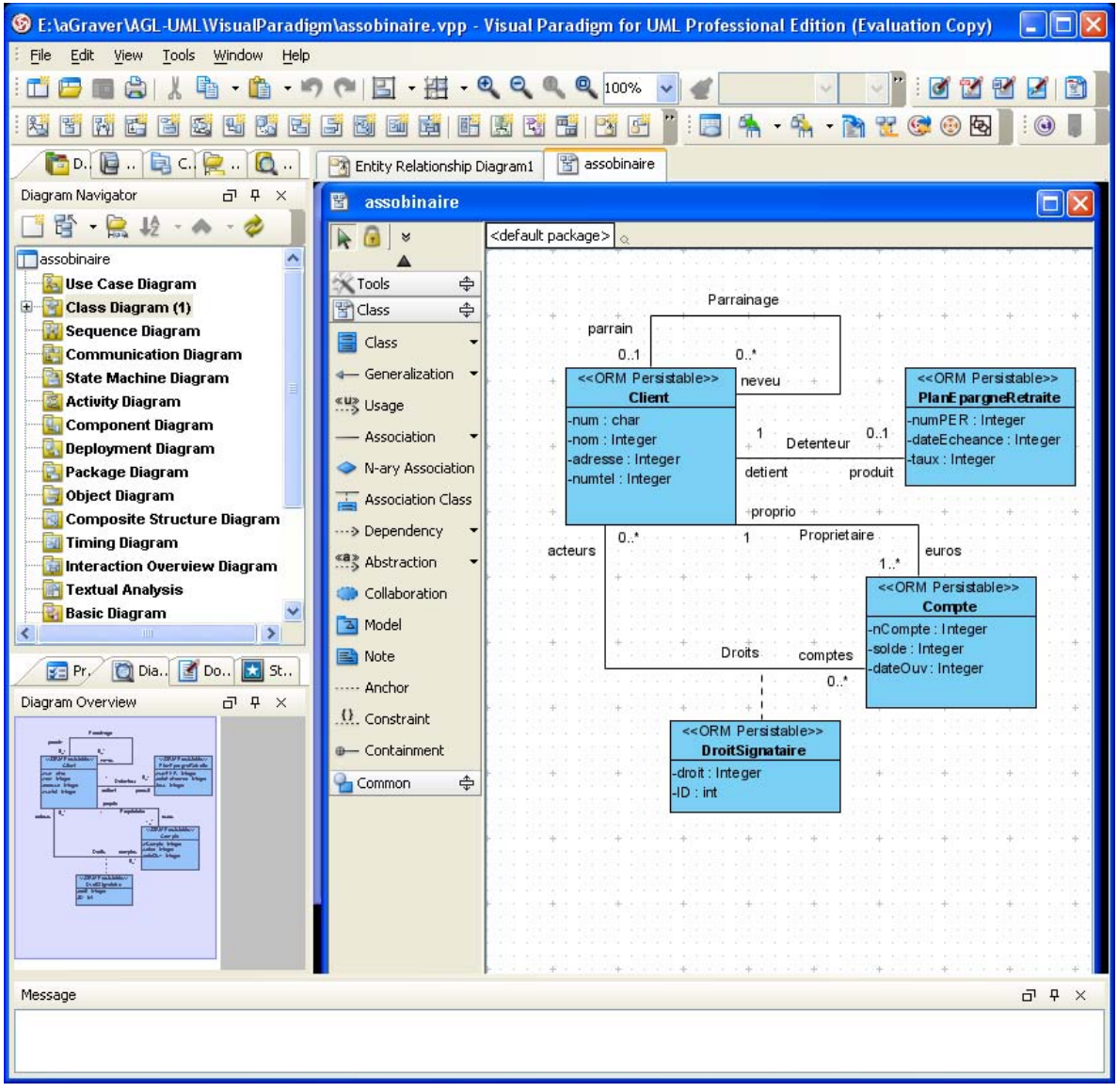
graphique). Ce sont les seuls formalismes qui permettent de se connecter à une base. Dans ces deux cas, vous devrez utiliser Visio au sein de l'environnement de Visual Studio. Si vous utilisez une base *Open Source*, il faudra trouver le bon pilote (en utilisant un pilote générique ODBC pour MySQL j'ai eu la surprise de constater que les colonnes de mes tables étaient inscrites en caractères japonais !).

Visual Paradigm

Une fois saisi le diagramme de classes UML, nommez impérativement tous les rôles des liens d'associations (sinon la génération au niveau logique ne sera pas possible). Ensuite configurez votre transformation (`Tools/Object... (ORM)/Wizards...`). Rendez persistantes vos classes, choisissez un identifant pour chacune et paramétrez votre connexion à la base de données cible. Une fois le modèle logique généré, il est nécessaire de le synchroniser (`Tools/Object... (ORM)/Synchronize...`) avec le diagramme de classes avant de pouvoir générer un script SQL. De toute façon, vous devrez souvent agir sur le modèle logique (surtout pour les identifiants des classes-associations) avant de générer une base de données.

Quelques points forts : un grand choix de SGBD est pris en compte ; la réactivité et l'efficacité du support. Les points faibles concernent le déplacement des objets et des liens sur un graphique du niveau logique (*Entity-Relationship*) qui n'est pas toujours réussi et la non-prise en compte des identifiants et des associations *n*-aires.

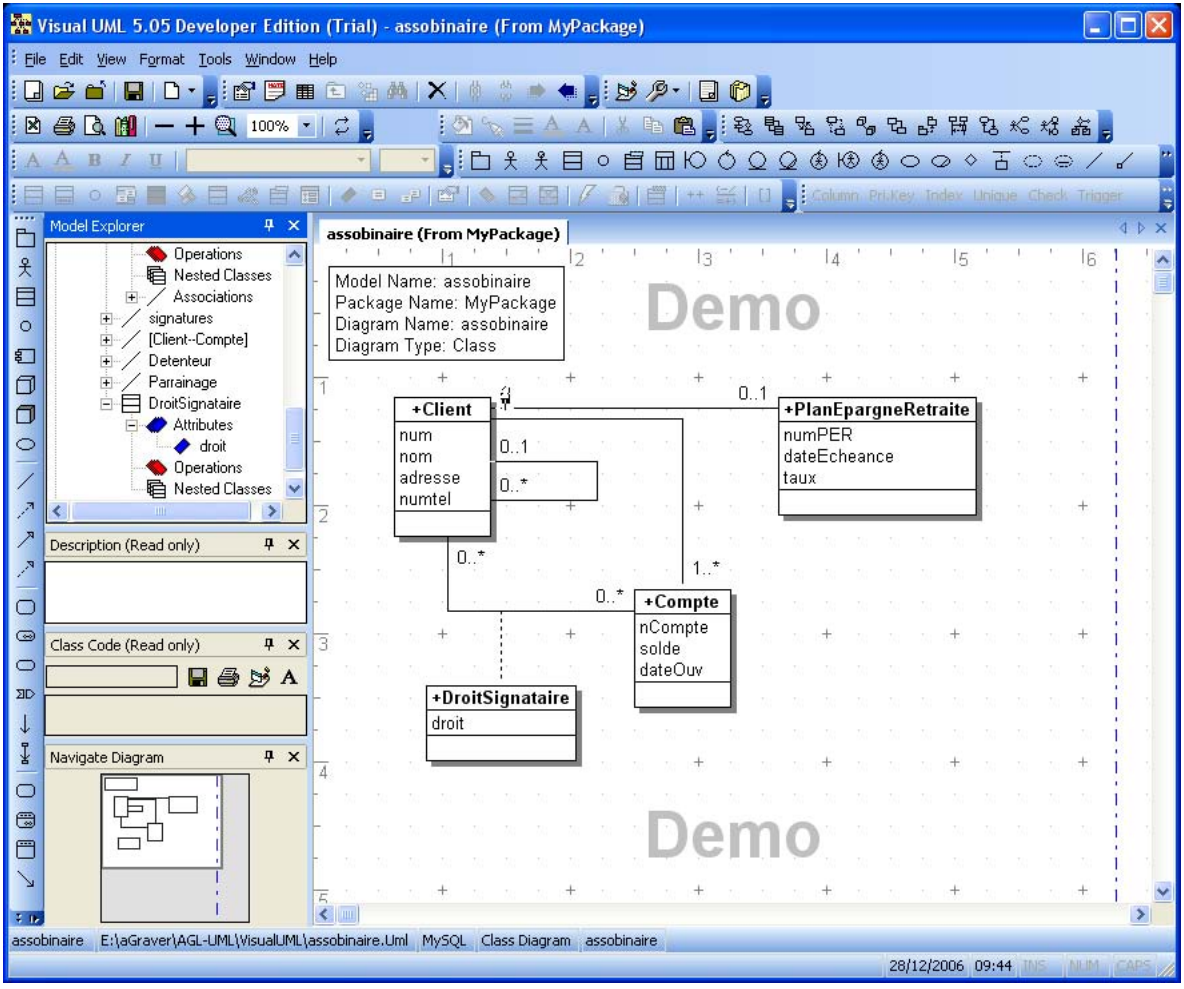
Figure 4-55 *Visual Paradigm*



Visual UML

Pas grand-chose à dire sur cet outil qui ne prend en compte que le niveau logique. Si vous êtes devenu adepte de la saisie manuelle des clés étrangères, rendez vos classes persistantes, allez chercher un pilote ODBC pour votre base de données et bon courage pour la suite.

Figure 4-56 *Visual UML*

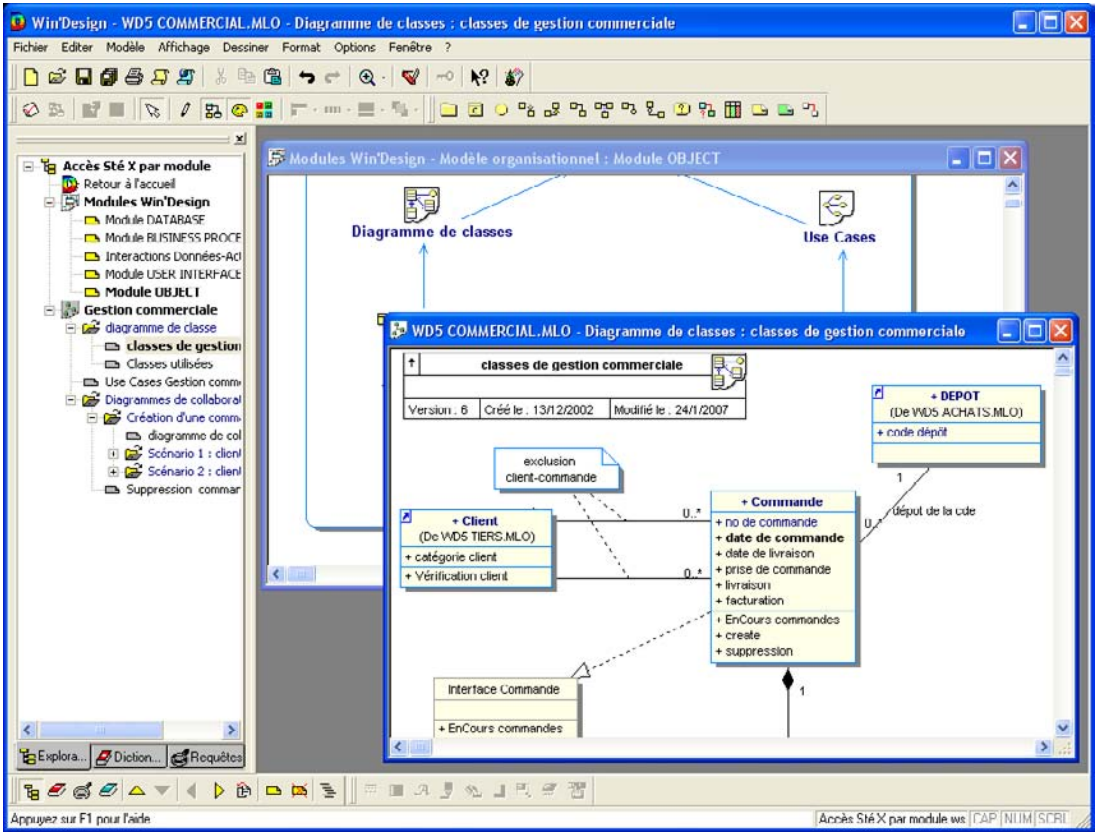


Win'Design

Win'Design est le produit de la société CECIMA basée à Aix en Provence. Il est présent sur le marché français depuis 1995. Développé initialement pour Merise/2, la notation UML arrive en 2002 avec la version 5. Depuis l'outil est en évolution constante.

La Gamme comprend quatre modules autonomes et complémentaires, qui s'articulent autour d'un référentiel (*Database* pour la conception et le reverse des bases de données, *Business Process* pour la modélisation des processus métier, *Object* pour la modélisation UML et *UserInterface* pour le maquettage des IHM). Vous devrez disposer du premier et du troisième module pour traduire des diagrammes de classes en script SQL. Comme PowerAMC, l'outil permet la double notation Merise/2 et UML 2 (sans le mode mixte de PowerAMC qui peut porter à confusion). Cet outil est le plus complet en ce qui concerne les contraintes Merise/2. Win'Design est probablement l'outil le plus facile à prendre en main et son interface est très intuitive.

Figure 4-57 *Win'Design*



Citons d'autres points forts : un très grand choix de SGBD est pris en compte, un grand nombre d'options de transformation de modèles (à tous les niveaux), la réactivité et l'efficacité du support. Notez que Win'Design est le seul à traiter correctement la transformation des associations *n*-aires et qu'avec PowerAMC et Rational, il est capable d'extraire un modèle conceptuel sans connexion à la base (à partir du seul script SQL de création des tables et contraintes). Le point faible de la version évaluée concerne la transformation de diagramme UML complexes (quelques cas spéciaux relevés lors de ces tests) en MLD et MCD. Les problèmes sont connus et en cours de correction.

Conclusion

Depuis 5 ans, le nombre d'outils a plus que doublé et les fonctionnalités deviennent de plus en plus puissantes. L'évolution d'UML semble se concrétiser car l'OMG a proposé en 2006 une RFP (*Request For Proposal*) nommée *Information Management Metamodel* dont le but est de définir un métamodèle incluant différentes sous-parties : *Common Warehouse Metamodel*, *UML2 Profile for Relational Data Modeling*, *UML2 Profile for Logical (Entity Relationship) Data Modeling*, *UML2 Profile for XML Data Modeling* et *UML2 Profile for Record Modeling* (*COBOL*). De nouvelles spécifications à propos de la modélisation sont sans doute à attendre prochainement comme par exemple la notion d'identifiant de classe. Les prochaines versions des outils devraient s'enrichir de ces nouvelles fonctionnalités. Enfin, sachez qu'il existe un forum francophone sur http://www.developpez.net relatifs aux outils Rational Rose, PowerAMC, Together et Win'Design.

Annexe 1

URL utiles

Outils

EnterPrise Architect	http://www.sparxsystems.com.au/products/ea.html
MagicDraw UML	http://www.magicdraw.com/
MEGA Designer	http://www.mega.com/en/product/mega_designer/
ModelSphere	http://www.silverrun.com/modelsphere.html
MyEclipse	http://myeclipseide.com
Objecteering	http://www.objecteering.com/
Poseidon	http://gentleware.com/index.php?id=30
PowerAMC	http://www.sybase.com/products/developmentintegration/poweramc
Rational Rose	http://www-306.ibm.com/software/awdtools/developer/datamodeler/
Together	http://www.borland.com
Visio	http://www.microsoft.com/france/office/visio
Visual Paradigm	http://www.visual-paradigm.com/product/vpuml/productinfovpumlse.jsp
Visual UML	http://www.visualuml.com/Products.htm
Win'Design	http://www.win-design.com/fr/

UML

http://www.omg.org/technology/documents/modeling_spec_catalog.htm

http://www.uml.org

http://uml.developpez.com/faq/

Annexe 2

Bibliographie

[ABR 74] J.R. ABRIAL, « Data Semantics », *Data Base Management*, North Holland, 1974.

[ACS 90] ACSIOME, *Modélisation dans la conception des systèmes d'information*, Masson, 1990.

[AST 76] M. ASTRAHAN *et al.*, « System R : Relational Approach to Database Management », *ACM Transactions on Database Systems*, Vol. 1, N° 2, 1976.

[ABI 87] S. ABITEBOUL, R. HULL, « IFO : A Formal Semantic Database Model System », *ACM Transactions on Database Systems*, Vol. 12, N° 4, 1987.

[ADI 93] M. ADIBA, C. COLLET, *Objets et Bases de données, le SGBD O2*, Hermès, 1993.

[AKO 01] J.AKOKA, I. COMYN-WATIAU, *Conception des bases de données relationnelles*, Vuibert, 2001.

[ANS 75] ANSI/X3/SPARC, « American National Standard Institute Study Group on DBMS : Interim report », *Bulletin of the ACM SIGMOD*, 1975.

[ANS 74] W. W. ARMSTRONG, « Dependencies Structures of Data Base Relationships », *Proceedings IFIP*, 1974.

[ARM 93] D. BEECH, « Collections of objects in SQL3 », *Proceedings of the Conference Very Large DataBases*, 1993.

[AST 76] M. ASTRAHAN *et al.*, « System R : Relational Approach to Database Management », *ACM Transactions on Database Systems*, Vol. 1, N° 2, 1976.

[ATK 89] M. ATKINSON, F. BANCILHON, D. DE WITT, K. DITTRICH, D. MAIER, S. ZDONIK, « The Object-Oriented database manifesto », *Proceedings of the Conference Deductive and Object-Oriented DataBases*, 1989.

[BAC 69] C.W. BACHMAN, « Data structure diagrams », *Data Base Journal of the ACM SIGBDP*, Vol. 1, N° 2, 1969.

[BEE 93] D. BEECH, « Collections of objects in SQL3 », *Proceedings of the Conference Very Large DataBases*, Dublin, 1993.

[BEE 77] C. BEERI, R. FAGIN, J.H. HOWARD, « A Complete Axiomatization for Functional and Multivalued Dependencies », *Proceedings of the Conference ACM SIGMOD*, Toronto, 1977.

[BER 76] P.A. BERNSTEIN, « Synthesizing Third Normal Form from Functional Dependencies », *ACM Transactions on Database Systems*, Vol. 1, N° 4, 1976.

[BOO 91] G. BOOCH, *Object Oriented Design with Application*, Benjamin Cummings, 1991.

[BOO 00] G. BOOCH, I JACOBSON, J. RUMBAUGH, *Le guide de l'utilisateur UML*, Eyrolles, 2000.

[BOR 97] S. BOBROWSKI, *Oracle8 Architecture*, Oracle Press, 1997.

[BOU 99] N. BOUDJILDA, *Bases de données et systèmes d'informations, le modèle relationnel : langages, systèmes et méthodes*, Dunod, 1999.

[BRO 05] F. BROUARD, C. SOUTOU, *SQL*, Pearson Education, 2005.

[CAT 91] R. G. G. CATTELL, *Object Data Management*, Addison-Wesley, 1991.

[CAT 93] R. G. G. CATTELL, *Object Databases : The ODMG-93 Standard*, Morgan-Kaufman, 1993.

[CAV 00] J.L. CAVARERO, R. LECAT, *La conception orientée objet, évidence ou fatalité*, Ellipses, 2000.

[CHA 96] D. CHAMBERLIN, Using the new DB2 : IBM's object-relational database system, Morgan & Kaufman, 1996.

[CHE 76] P.P. CHEN, « The Entity-Relationship Model : Towards a Unified View of Data », *ACM Transactions on Database Systems*, Vol. 1, N° 1, 1976.

[CHR 87] C. CHRISMENT, *Mise en œuvre des bases de données*, Eyrolles, 1987.

[COA 91] P. COAD, E. YOURDON, *Object Oriented Analysis*, Prentice Hall, 1991.

[COD 69] E.F. CODD, « Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks », *IBM research report RJ599*, 19 août 1969.

[COD 70] E.F. CODD, « A Relational Model for Large Shared Data Banks », *Communications of the ACM*, Vol 13, N°6, 1970.

[COD 72] E.F. CODD, « Further Normalization of the Data Base Relational Model », *Data Base Systems, Courant Computer Science Symposia Series 6*, Prentice Hall, 1972.

[COD 74] E.F. CODD, « Recent Investigations into Relational Database Systems », *Proceedings of the IFIP*, Stockholm, 1974.

[CON 05] T. CONNOLY, C. BEGG, *Systèmes de bases de données*, Reynald Goulet-Eyrolles, 2005.

[COP 84] G. COPELAND, D. MAIER, « Making Smalltalk a database system », *Proceedings of the Conference ACM SIGMOD*, Boston, 1984.

[DAH 66] O.DAHL, K. NYGAARD, « An algol-based simulation langage », *Communications of the ACM*, Vol. 9, N° 9, 1966.

[DAT 96] C.J. DATE, *An Introduction to Database Systems*, Addison-Wesley, 6th edition, 1996.

[DAT 98] C.J. DATE, H. DARWEN, *Foundation for Object/Relational Databases, The Third Manifesto*, Addison-Wesley, 1998.

[DAT 00] C.J. DATE, *Introduction aux bases de données*, International Thomson Publishing, traduction de la septième édition de l'ouvrage d'Addison-Wesley, 2000.

[DEL 82] C. DELOBEL, M. ADIBA, *Bases de données et systèmes relationnels*, Dunod, 1982.

[DEL 91] C. DELOBEL, C. LÉCLUSE, P. RICHARD, *Bases de données : des systèmes relationnels aux systèmes à objets*, InterEditions, 1991.

[DEL 94] C. DELOBEL, « Convergence and/or divergence between SQL3 standard evolution and ODMG-93 standard for object-oriented database systems », *Proceedings of the Basque International Workshop on Information Technology (BIWIT'94)*, 1994.

[DEL 00] P. DELMAL, *SQL2-SQL3, Applications à Oracle*, De Boeck Université, 2000.

[DIO 94] D. DIONISI, *L'essentiel sur Merise*, Eyrolles, 2e édition, 1998.

[DOR 99] P. DORSEY, J. R. HUDICKA, *Oracle8, Design Using UML Object Modeling*, Oracle Press, 1999.

[ELM 04] R. ELMASRI, S. NAVATHE, *Conception et architecture des bases de données*, Pearson Education, 2004.

[FRE 93] P. FRESNAIS, « Comparaison des SGBD relationnels et orientés objets à travers une étude de cas », *Actes du congrès biennal de l'AFCET*, Versailles, 1993.

[GAR 94] G. GARDARIN, « Translating relational to object databases », *Ingénierie des systèmes d'information*, Vol. 2, N° 3, *Hermès*, 1994.

[GRI 98a] S. GRIMES, « Modeling Object Relational Databases », *Object-Relational Database Summit, http://www.dbsummit/or/grimes.html*, septembre 1998.

[GRI 98b] S. GRIMES, « Object Relational Reality Check », *Database Programming and Design on line*, *http://www.dbpd.com/9807feat.htm*, 1998.

[HAI 05] J.L. HAINAULT, *Bases de données et modèles de calculs*, Dunod, 2005.

[HAL 01] T. HALPIN, *Information Modeling and Relational Databases*, Mogan Kaufmann, 2001.

[ISO 92] ISO/IEC 9075:1992, «Database Langage SQL ».

[JAC 92] I. JACOBSON, M. CHRISTERSON, P. JONSSON, G. OVERGAARD, *Object-Oriented Software Engineering : a Use Case Driven Approach*, Addison-Wesley, 1992.

