

Cours d'Ateliers de Génie Logiciel

Table des matières

Chapitre I : Les Langages Informatiques

- I. 1. Introduction
- I. 2. Segmentation Informatique
- I. 3. Notion d’Informatique
- I. 4. Typologie des langages
- I. 5. Diversité des langages
- I. 6. Conception d’un langage
- I. 7. Développement d’un programme
- I. 8. Historique
- I. 9. Syntaxe et Sémantique
- I. 10. Structure des programmes
- I. 11. Langages de 4ème et 5ème Génération
- I. 12. Performance et popularité
- I. 13. Conclusion

Chapitre II : Les Ateliers de Génie Logiciel

- II.1. Introduction
- II.2. Le processus logiciel, ou cycle de vie du logiciel
- II.3. Ateliers de Génie Logiciel
- II.4. Les SGBD avancés

Chapitre III : Travaux Pratiques avec les AGL

- III.1. Travaux Pratiques avec Oracle
- III.2. Travaux Pratiques avec WinDev
- III.3. Travaux Pratiques avec Visual Studio

Cours d'Ateliers de Génie Logiciel

Objectifs du cours

Il existe un très grand nombre de langages de programmation (>2000) et un grand nombre d'Ateliers de Génie Logiciel (AGL).

Le but du cours est de vous permettre de :

- ✓ utiliser la majorité de ces langages et AGL efficacement en peu de temps.
- ✓ savoir quel langage ou AGL utiliser en fonction de la finalité du logiciel à développer.
- ✓ connaître et comprendre les standards et les outils du marché pour le développement d'applications.
- ✓ réaliser une application de gestion simple à l'aide :
 - ✓ d'un environnement de développement () .
 - ✓ de composants graphiques standards.
 - ✓ de la programmation événementielle (procédurale ou orientée objet).
 - ✓ appliquer les principes des standards de développement.
 - ✓ maintenir et faire évoluer une application existante.
 - ✓ mettre en œuvre les principes d'accès aux données.
 - ✓ implémenter une base de données efficiente.

Méthodes pédagogiques

Le contenu du cours est offert pendant 60 heures. Il varie entre apports théoriques et démonstrations pratiques.

Ce cours est fortement orienté sur l'acquisition de compétences opérationnelles pratiques, il demande donc une forte implication quant à la réalisation des exercices de la part des étudiants.

Les travaux pratiques sont distribués pendant le cours.

A la fin du cours, les étudiants réaliseront un projet de conception et de développement.

Mode d'évaluation (sous réserve de modification)

Présence et ponctualité en classe : 10% de la note finale du cours.

Projet de Conception/développement et soutenance/démonstration : 50% de la note finale du cours.

Examens : 40% de la note finale du cours.

Chapitre I - Les langages informatiques

I. 1 - Introduction

Un langage de programmation informatique est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un alphabet, d'un vocabulaire, de règles de grammaire, de significations, mais aussi d'un environnement de traduction censé rendre sa syntaxe compréhensible par la machine.

Les langages de programmation permettent de décrire d'une part les structures des données qui seront manipulées par l'appareil informatique, et d'autre part d'indiquer comment sont effectuées les manipulations, selon quels algorithmes. Ils servent de moyens de communication par lesquels le programmeur communique avec l'ordinateur, mais aussi avec d'autres programmeurs; les programmes étant d'ordinaire écrits, lus, compris et modifiés par une équipe de programmeurs.

Un langage de programmation est mis en œuvre par un traducteur automatique : compilateur ou interprète. Un compilateur est un programme informatique qui transforme dans un premier temps un code source écrit dans un langage de programmation donné en un code cible qui pourra être directement exécuté par un ordinateur, à savoir un programme en langage machine ou en code intermédiaire, tandis que l'interprète réalise cette traduction « à la volée ».

Les langages de programmation offrent différentes possibilités d'abstraction et une notation proche de l'algèbre, permettant de décrire de manière concise et facile à saisir les opérations de manipulation de données et l'évolution du déroulement du programme en fonction des situations. La possibilité d'écriture abstraite libère l'esprit du programmeur d'un travail superflu, notamment de prise en compte des spécificités du matériel informatique, et lui permet ainsi de se concentrer sur des problèmes plus avancés.

La facilité d'utilisation, la portabilité et la clarté sont des qualités appréciées des langages de programmation. La facilité d'utilisation, qui dépend de la syntaxe, du vocabulaire et des symboles, influence la lisibilité des programmes écrits dans ce langage et la durée d'apprentissage. La portabilité permet à un programme écrit pour être exécuté par une plateforme informatique donnée (un système d'exploitation) d'être transféré en vue d'être exécuté sur une autre plateforme.

Les programmeurs apprécient qu'un langage de programmation soit en ligne avec les bonnes pratiques de programmation et d'ingénierie, qu'il encourage la structuration du programme, facilite la maintenance des programmes et qu'il dissuade, voire interdise les mauvaises pratiques. L'utilisation de l'instruction « goto », par exemple, qui existe depuis les premiers

Cours d'Ateliers de Génie Logiciel

langages de programmation, est considérée comme une mauvaise pratique. Son utilisation est déconseillée, voire impossible dans les langages de programmation récents.

L'alignement sur les standards industriels, la possibilité d'utiliser des fonctionnalités écrites dans un autre langage de programmation et l'exécution simultanée de plusieurs threads sont des possibilités appréciées des langages de programmation.

Chaque langage de programmation supporte une ou plusieurs approches de la programmation – paradigmes. Les notions induisant le paradigme font partie du langage de programmation et permettent au programmeur d'exprimer dans le langage une solution qui a été imaginée selon ce paradigme.

Les premiers langages de programmation ont été créés dans les années 1950 en même temps que l'avènement des ordinateurs. Cependant, de nombreux concepts de programmation ont été initiés par un langage ou parfois plusieurs langages, avant d'être améliorés puis étendus dans les langages suivants. La plupart du temps la conception d'un langage de programmation a été fortement influencée par l'expérience acquise avec les langages précédents.

I. 2 - Segmentation Informatique

a) Autrefois

Découpage

Matériel et Logiciel : clivage traditionnel hardware-software.

Logiciels

Système d'exploitation avec des applications.

Développement d'applications à partir d'un langage de programmation : compilateur (pas d'éditeur de texte ni d'environnement de développement, au début).

b) Maintenant

Découpage

Matériel, Logiciel, Contenu (signal, donnée, information, document, connaissance)

Terminal (fixe, mobile) & Réseaux.

L'informatique se conçoit en réseau.

Logiciels

Système d'exploitation, Middleware (Intergiciels) et protocoles.

Cours d'Ateliers de Génie Logiciel

Environnement de développement (éditeur, compilateur, bibliothèque de composants, debugger symbolique, framework, ...).

I. 3 – Notion d’Informatique

a) Informatique, Ordinateur, Logiciel, Langage

Un ordinateur est un outil qui résout des problèmes au moyen de programmes ou logiciels développés dans un (ou plusieurs) langages.

La taille et la complexité des logiciels ont augmenté dans le temps.

De manière indicative, un logiciel important possède, en moyenne, la taille suivante :

- Année 70 : 10 000 lignes de code
- Année 80 : 50 000 lignes de code : début des interfaces graphiques (clavier, souris, écran)
- Année 90 : 100 000 lignes de code
- Année 00 : 1 000 000 lignes de code : Word, Excel, ... occupent 5 Mo.

b) Langage

Passage de la langue naturelle via le langage de haut niveau au langage machine.

Langage : syntaxe et sémantique => Compilateur, Interpréteur, Analyseur sémantique.

Un système d’exploitation gère le temps, l’espace (mémoire, disque) et les entrées-sorties.

Une application va donner du sens, en transformant un contenu (des entrées) en un autre contenu (des sorties), par un calcul informatique en prenant de la place et en prenant du temps. La valeur ajoutée dépend de la force du langage et de l’intelligence de l’application.

Au fil du temps se sont développées les applications interactives qui agissent en temps réel sur l’environnement ou interagissent avec l’utilisateur.

Un langage informatique est un langage destiné à décrire l’ensemble des actions consécutives qu’un ordinateur doit exécuter. Un langage informatique est une manière pragmatique de donner des instructions à un ordinateur.

Un langage informatique est rigoureux : à une instruction correspond une action du processeur. Le langage utilisé par le processeur est le langage machine. Il s’agit d’une suite de 0 et de 1 (en hexadécimal), pas compréhensible facilement par le commun des mortels.

Il est donc plus pratique de trouver un langage intermédiaire, compréhensible par l’homme, qui sera ensuite transformé en langage machine pour être exploitable par le processeur.

Cours d'Ateliers de Génie Logiciel

L'assembleur est un langage proche du langage machine mais il permet déjà d'être plus compréhensible.

Ainsi un programme développé pour une machine ne pourra pas être « porté » sur un autre type de machine. Pour pouvoir l'utiliser sur une autre machine il faudra alors parfois réécrire entièrement le programme.

Un langage informatique a donc plusieurs avantages:

- ✓ il est plus facilement compréhensible que le langage machine
- ✓ il permet une plus grande portabilité, c'est-à-dire une plus grande facilité d'adaptation sur des machines de types différents.

I. 4 – Typologie des langages

On va définir des structures de données qui ont des natures complexes, des types hétérogènes, etc.

On veut obtenir un résultat : par exemple, on ne veut pas prouver qu'il existe des nombres premiers, on veut les calculer.

Les langages informatiques vont être conçus pour écrire des programmes qui permettent de définir des structures de données complexes et de mettre en œuvre des calculs et algorithmes efficaces.

Il existe plusieurs écoles (paradigmes) de programmations. Citons quelques-uns.

Langage Concurrent

En paradigme concurrent un programme peut effectuer plusieurs tâches en même temps. Ce paradigme introduit les notions de thread, d'attente active et d'appel de fonction à distance. Ces notions ont été introduites dans les années 1980 lorsque, à la suite de l'évolution technologique, un ordinateur est devenu une machine comportant plusieurs processeurs et capable d'effectuer plusieurs tâches simultanément. Les langages de programmation contemporains de 2013 tels que C++ et Java sont adaptés aux microprocesseurs multi-cœur et permettent de créer et manipuler des threads. Plus récemment, on a vu apparaître des langages intégralement orientés vers la gestion de la concurrence, comme le langage Go.

Langage Visuel

Dans la grande majorité des langages de programmation, le code source est un texte, ce qui rend difficile l'expression des objets bidimensionnels. Un langage de programmation tel que Delphi ou C# permet de manipuler des objets par glisser-déposer et le dessin ainsi obtenu est ensuite traduit en une représentation textuelle orientée objet et événementielle. Le

Cours d'Ateliers de Génie Logiciel

paradigme visuel a été introduit à la fin des années 1980 par Alan Kay dans le langage Smalltalk, dans le but de faciliter la programmation des interfaces graphiques.

Langage Événementiel

Alors qu'un programme interactif pose une question et effectue des actions en fonction de la réponse, en style événementiel le programme n'attend rien et est exécuté lorsque quelque chose s'est passé. Par exemple, l'utilisateur déplace la souris ou presse sur un bouton. Dans ce paradigme, la programmation consiste à décrire les actions à prendre en réponse aux événements. Et une action peut en cascade déclencher une autre action correspondant à un autre événement. Le paradigme événementiel a été introduit par le langage Simula dans les années 1970. Il est devenu populaire à la suite de l'avènement des interfaces graphiques et des applications web.

I. 5 – Diversité des langages

Langages anciens encore utilisés (Fortran, Cobol)

D'autres émergent (Java en 95, XML en 98)

700 langages en 1969 à la NASA pour la mission Apollo.

2000 langages en l'an 2000.

Langage orienté objet : Java, Smalltalk, Eiffel, Delphi

Langage procédural : Pascal, C, Fortran

Hybride orienté objet : C++, ADA 95

Langage fonctionnel : FP, ML, LISP

Langage logique : Prolog

Langage à balise : SGML, HTML, XML

Langage de Script : Perl, CGI

Résumé de la liste des principaux langages.

Langage	Applications classiques	Compilé/interprété
ADA	Embarqué	compilé
BASIC	Macro de traitement bureautique	interprété
C	Programmation système	compilé
C++	Programmation système objet	compilé
Cobol	Gestion	compilé

Fortran	Calcul	compilé
Java	Programmation orientée Internet	intermédiaire
LISP	Intelligence artificielle	intermédiaire
Pascal	Enseignement	compilé
Prolog	Intelligence artificielle	interprété
Perl	Traitement de chaînes de caractères	interprété

I. 6 – Conception d'un langage

Il faut choisir des langages conviviaux pour pouvoir réaliser des logiciels faciles à concevoir, écrire, lire, tester, exécuter, documenter et modifier.

Propriétés des langages

Puissance d'expression

Simplicité

Implémentation

Détection d'erreur et correction

Correction et standards

a) Puissance d'expression

Le programmeur doit réfléchir à des solutions de problèmes, exprimées dans les termes du problème plutôt qu'en terme d'informatique sur laquelle la solution est implémentée en langage machine.

Le programmeur doit se concentrer sur le problème à résoudre.

La puissance d'expression se situe dans la notation pour décrire des algorithmes et des structures de données.

Le langage doit supporter des idées de programmation structurée et de modularité.

Un langage informatique est un outil pour des types d'applications donnés. Ce ne peut être un outil universel et intemporel. Il dépend de la technologie informatique tout entière : Ada possède une puissance d'expression forte en calcul numérique, moindre en traitement de données.

Les types

Les langages modernes utilisent des types de plus en plus puissants.

Cours d'Ateliers de Génie Logiciel

Certains types définis une fois pour toute.

Dans les langages modernes, il existe possibilité de construire des nouveaux types : package Ada et classes C++ ou Java.

On peut alors définir a priori des types prédéfinis par l'utilisation de bibliothèques standards que le programmeur peut utiliser. Mais le programmeur peut en définir lui-même.

Traitement des exceptions

Événements exceptionnels : division par zéro, erreur d'un indice de tableau qui va au-delà des limites, rencontre inopinée de la fin de fichier, ...

Le programme doit anticiper les fautes.

Ada a été vraiment le premier à lever les erreurs.

Celles impossibles à détecter à la compilation le seront à l'exécution : Raised (Ada, Delphi) ou thrown (C++ et Java).

L'exécution normale est interrompue et le contrôle est transféré à une partie spéciale appelée traitement des exceptions.

Le traitement des exceptions est crucial pour les programmes tolérant aux fautes et la sûreté de fonctionnement.

b) Simplicité

Facilité à lire, écrire et comprendre (succès de Visual Basic)

Problème de lisibilité pour comprendre facilement ce qui est écrit (Commentaires, convention de noms).

Modification des programmes avec leur documentation (difficile à mettre en œuvre).

c) Implémentation

Wirth, concepteur de Pascal et de Modula-2, a voulu concevoir des langages pour des programmes efficaces.

Les caractéristiques qui sont difficiles à compiler sont difficiles à comprendre.

Compilateur

Un programme écrit dans un langage dit "compilé" va être traduit une fois pour toutes par un programme annexe (le compilateur) afin de générer un nouveau fichier qui sera autonome, c'est-à-dire qui n'aura plus besoin d'un programme autre que lui pour s'exécuter. De plus, la traduction étant faite une fois pour toute, il est plus rapide à l'exécution. Toutefois il est moins souple que le programme écrit avec un langage interprété car à chaque modification du fichier source (fichier intelligible par l'homme: celui qui va être compilé) il faudra recompiler le programme pour que les modifications prennent effet.

Certaines applications sécurisées nécessitent la confidentialité du code pour éviter le piratage.

Un programme compilé a pour avantage de garantir la sécurité du code source.

Un langage interprété, étant directement intelligible (lisible), permet à n'importe qui de connaître les secrets de fabrication d'un programme et donc de copier le code voire de le modifier. Il y a donc risque de non-respect des droits d'auteur.

L'exécution d'un programme écrit dans un langage impératif se fait par traduction (compilation) du programme source dans un programme en code machine qui est « équivalent ».

Le code en langage machine est exécuté.

La facilité de la traduction du langage et l'efficacité du code résultant est un facteur majeur pour le succès de ce langage.

Les « gros » langages auront des compilateurs qui seront importants, lents et coûteux.

La vérification des types a lieu à la compilation, pour détecter des erreurs logiques.

Interpréteur

Un langage informatique est différent du langage machine. Il faut donc le traduire pour le rendre intelligible du point de vue du processeur. Un programme écrit dans un langage interprété a besoin d'un programme auxiliaire (l'interpréteur) pour traduire au fur et à mesure les instructions du programme.

Alternative au compilateur : interpréteur.

Un interpréteur peut directement exécuter le programme source. Mais plus communément, le programme source est plutôt traduit dans une forme intermédiaire qui peut être exécutée par l'interpréteur. Dans ce cas, l'interpréteur implémente une machine virtuelle.

Exécuter un programme sous le contrôle d'un interpréteur est beaucoup plus lent qu'exécuter le code équivalent en langage machine mais cela donne plus de flexibilité à l'exécution.

Cours d'Ateliers de Génie Logiciel

Exemple : langage de traitement de chaîne SNOBOL4, le langage orienté objet Smalltalk, le langage fonctionnel Lisp, le langage logique Prolog, le langage de script Perl.

Notion de machine virtuelle

Java est impératif et devrait donc être compilé mais les programmes sont interprétés. Le programme source en Java est traduit en code pour une machine virtuelle pour se faire exécuter sur une machine différente.

Certains langages appartiennent donc aux deux catégories compilables et interprétables (LISP, Java, ...) car le programme écrit avec ces langages peut dans certaines conditions subir une phase de compilation intermédiaire vers un fichier écrit dans un langage qui n'est pas intelligible (donc différent du fichier source) et non exécutable (nécessité d'un interpréteur). Les applets Java, petits programmes insérés dans les pages Web, sont des fichiers qui sont compilés que l'on exécute à partir d'un navigateur internet.

d) Détection d'erreurs et correction d'erreurs

Ce serait idéal si toutes les erreurs apparaissaient à la compilation. En fait, c'est impossible. Quand on met au point un programme, on le modifie au fur et à mesure ; différentes erreurs apparaissent à la compilation et à l'exécution. Il faut veiller à ne pas détruire la structure du programme quand on corrige.

e) Correction et standards

Pour prouver qu'un programme est correct, il est nécessaire d'avoir une définition rigoureuse de chaque construction du langage (méthode pour définir la syntaxe et la sémantique du langage).

I. 7 – Développement d'un programme

a) Maîtrise du langage par le développeur

Il est difficile de bien maîtriser un langage.

Certains parlent de la maîtrise d'un langage et d'un OS par génération : la génération Pascal, la génération C et Unix, la génération Java et Internet

Il existe une attitude irrationnelle vis-à-vis de « son » langage.

Il faut s'approprier (au moins) un langage et ne pas tomber dans le piège de devenir un fanatique de son langage ou de son environnement de développement.

Cours d'Ateliers de Génie Logiciel

Développer un programme est un travail difficile qui demande de la concentration.

Développer un programme ce n'est pas un travail d'artisan ou d'artiste. Il faut le considérer comme un travail industriel intégré dans une équipe.

Un développeur peut maîtriser parfaitement 10 000 lignes de code.

Pour développer un programme de 50 000 à 100 000 lignes de code, il faut 5 à 10 personnes en parallèle avec une bonne méthodologie.

Pour réaliser un programme de 1 million de lignes de code (Netscape, Excel, Word, etc.), il faut toute une équipe car une seule personne ne peut maîtriser ces programmes.

b) Productivité et Génie logiciel

Le problème de l'informatique aujourd'hui, c'est la faible productivité.

Le génie logiciel a émergé en 1968 : la productivité en logiciel depuis a été multipliée par 4 seulement.

Entre temps, les logiciels se sont complexifiés (d'un facteur beaucoup plus grand que le facteur de productivité), d'où la volonté de réutilisation, de structuration par modules.

Aujourd'hui, on ne crée plus ex nihilo (à partir de rien) un programme, on « reprend » un vieux programme (avec avantages et inconvénients).

Un développeur a une productivité de 3 à 5 lignes par heure (tout compris) pour un programme difficile et de 2000 lignes par mois (quand on sait parfaitement ce qu'il faut faire) pour un prototype facile (à titre indicatif).

Pour améliorer la productivité :

- ✓ il faut travailler en parallèle : méthodologie pour travailler en équipe ;
- ✓ il faut réutiliser le travail des autres : notion de composants logiciels et méthodologie logiciel (modularité).

En général, on n'écrit pas un programme seul, on modifie ou adapte le programme d'un autre.

c) Modèle de développement

Les différentes étapes sont les suivantes :

Analyse du besoin utilisateur : parfois vague, incohérent, ambigu, incomplet.

Analyse des exigences pour régler les conflits des différents points de vue.

Cours d'Ateliers de Génie Logiciel

Spécification pour définir ce qu'on fait : document pour dire ce que le logiciel fera (et ne fera pas).

Définitions, en parallèle de la spécification, des plans de test et de validation.

Conception d'une solution et implémentation en utilisant un ou plusieurs langages de programmation.

Validation et vérification (ou preuve de programme) : ce que fait le programme c'est ce qui est spécifié.

Test avec des données (prouve la présence d'erreurs, ne prouve pas l'absence d'erreurs).

Maintenance curative (correction d'erreurs)

Modification du programme pour des évolutions ou des additions ou des changements de spécifications

Certification de logiciels en sécurité et en sûreté de fonctionnement pour les télécoms, l'avionique, ...

Méthodologie & cycle de vie d'un logiciel

La conception d'un logiciel est souvent influencée par le langage qu'on va utiliser à l'implémentation.

Le modèle en cascade (waterfall model) : les étapes sont réalisées en séquence.

Le modèle en spirale de Boehm (1988) : le modèle séquentiel waterfall est remplacé par un mode itératif et incrémental pour la gestion du risque.

Avec la méthode orientée objet on utilise un « framework » d'objets communicants pour spécifier, concevoir et implémenter. L'orienté objet est assez bien adapté pour une approche incrémentale. Les objets sont définis à différents degrés d'abstraction : à mesure que le développement se déroule, on ajoute des caractéristiques à la description des objets.

Développement orienté objet avec les notations UML (Booch, 1998) : diagramme de classe.

d) Langages ou systèmes

Les langages modernes supportent la programmation en réseau et la création d'interfaces graphiques à travers des bibliothèques. On peut se demander si ces bibliothèques font partie du langage ou de l'environnement de développement.

Souvent ces librairies sont dépendantes du système d'exploitation.

Cours d'Ateliers de Génie Logiciel

Delphi et Visual Basic dépendent de Microsoft Windows, langages liés à l'OS et à l'implémentation donc à l'environnement de développement système.

Java librairies indépendantes de l'OS.

Programmes avec des interfaces graphiques : ils sont « event driven », on attend un clic de l'utilisateur pour interagir.

I. 8 – Historique

Important de comprendre l'évolution des langages, pour comprendre les erreurs du passé et éviter de les répéter.

Fortran et Cobol sont encore utilisés 40 ans après leur apparition.

Difficile de faire apparaître un nouveau langage.

Énorme investissement dans les systèmes stables (voir le « bug de l'an 2000 »).

a) Survol historique

Années 50

Se débarrasser de l'inflexibilité du matériel (Von Neumann).

Années 60

Mission Apollo : 1 million de lignes de code, ordinateurs de 0,5 mips avec mémoire de quelques kilo-octets.

Course à la performance (pas assez de mips)

Tous les ordinateurs sont différents (structure du matériel, des systèmes d'exploitation, ...)

Fortran, Algol, Cobol, PL/I, Basic.

Fortran, Cobol et Basic sont restés (ils ont beaucoup évolué depuis).

En 70

Matériels très différents (IBM, HP, DEC, Control Data, Univac, ...).

Problèmes de portabilité, nécessité d'avoir un standard et un compilateur (motivation pour ADA du DoD).

Logiciels assez petits (pas d'interface graphique, peu de données, ...).

Cours d'Ateliers de Génie Logiciel

Problème du temps réel (Ada).

Année 70 - 80

3 langages sont restés importants dans les années 90.

1) Pascal était le langage enseigné dans les universités. Il n'est plus enseigné dans les universités comme le langage de base bien qu'il soit pédagogique.

2) Le langage C est le langage des programmeurs systèmes et a été, avec Unix, le langage des stations de travail. Il est devenu C++ qui a servi ensuite à créer Java.

3) Ada a démarré fin 70, Ada 83 et Ada 95 (le langage et ses évolutions sont venus toujours trop tard).

Tous ces langages ont eu une extension orientée objet.

Années 75 - 85

Mini-ordinateurs (PDP, VAX, SUN, HP, etc.) et Unix avec la station de travail et l'interface graphique.

Années 80

Années de l'Intelligence Artificielle.

Programmation fonctionnelle et logique.

Niveau d'abstraction plus grand, donc plus facile de raisonner.

Langages fonctionnels : moderne Scheme (Lisp), ML, StandardML, OCAML et Haskell.

En 85

Bouleversement avec l'arrivée des Personal Computers (Apple, Microsoft).

Années 85 - 95

Les langages orientés objet.

Smalltalk, Eiffel, C++, Delphi, Ada 95 et Java.

Années 90

L'utilisation des ordinateurs a changé radicalement.

Apparition du PC à la maison.

Les interfaces graphiques sont développées en 70 (Xerox), sur Apple en 80 mais c'est en 90 que les IHMs sont devenues la norme pour dialoguer avec des ordinateurs (grâce aussi à la programmation objet très adaptée aux IHMs).

En 90 est aussi apparu le séisme du Web avec Internet.

On s'écarte des langages de programmation, on parle de systèmes de programmation.

Les librairies standard de Java deviennent la partie centrale du langage.

Delphi est un ensemble de bibliothèques de classes et un environnement de développement au-dessus de Object Pascal.

Même chose pour Visual C++ ou Visual Basic.

En 95

Uniformisation : Intel + Microsoft + Cisco

Réaction Unix => Linux, logiciel libre, Java

Sur Internet, idée de Machine Virtuelle avec du bytecode pour la portabilité (à cause de la puissance des machines).

Années 00

Intel + Microsoft + Cisco dominent les stations de travail et l'Internet.

L'approche objet joue un rôle central, marquée par :

- ✓ Les architectures réparties de l'Internet (Web, XML et Java)
- ✓ Les architectures embarquées des Télécoms mobiles (terminaux à faible consommation)

Mouvement vers l'interopérabilité des applications réparties (P2P, etc.).

Nécessité de faire intervenir des langages à balises pour gérer l'hétérogénéité.

b) Machines du début

Spéculation du 19^{ème} siècle

Ada, comtesse de Lovelace, est considérée la première développeuse car elle a travaillé avec Charles Babbage sur la Machine Analytique (spéculation académique) au 19^{ème} siècle. Son prénom a été utilisé pour le langage Ada.

1er Travaux

Fin des années 1940 travaux sur machine et langage (Turing, von Neumann).

Il n'y a pas de développement de langage de programmation de haut niveau.

Ce sont des « codes instruction » très primitifs par rapport à l'assembleur.

Les données sont dans le code, il n'y a pas de boucles.

Les programmes de l'époque ont des erreurs avec la nécessité de créer des patchs pour les corriger.

Au milieu des années 50, émerge une nécessité d'améliorer la programmation des 2 côtés de l'Atlantique (États-Unis, Grande Bretagne, France).

c) Premiers langages scientifiques

Fortran

IBM : Mathematical FORmula TRANslating system.

Fortran 1 en 1956, compilateur en 58

Concepteur : Backus. Il émet quelques principes nouveaux à l'époque :

Ce qui compte ce n'est pas la beauté de programmer mais son économie.

La version codée à la main doit aller moins vite que le programme compilé.

Le coût de développement et le coût de debugging est plus élevé que le coût de l'exécution.

D'où l'idée d'utiliser des notations mathématiques.

Fortran conçu pour les calculs scientifiques sur IBM 704 et ses cartes perforées.

Pas de facilités de traitement de chaînes de caractères

Structure de données : uniquement le tableau

Apparition des commentaires dans le programme.

Cours d'Ateliers de Génie Logiciel

Premier compilateur pas efficace mais succès quand même.

Évolution : Fortran en 66, en 77 (développé en 78) et Fortran 90 (développé en 91).

Algol

Profonde influence sur la définition et la conception de langages.

Défini par un comité européen et américain (International Algebraic Language IAL ALGOrithmic Language).

Proche du langage mathématique et lisible.

Algol 58 au début mais jamais implémenté.

Algol 60 est structuré en blocs, les variables ne sont pas visibles en dehors des blocs.

Algol 60 n'a pas dépassé Fortran (compilateur apparu 3 ans après).

Algol 68 se veut universel (traitement de données).

IBM ne supporta pas Algol. Les compilateurs Fortran étaient plus efficaces.

Pascal

Niklaus Wirth fin 60, début 70 grâce à son travail sur Algol W (Wirth).

Accent sur l'implémentation du programme bien structuré et bien organisé.

Pour défier le Fortran utilisé aux USA.

On introduit les types de données qui peuvent être construits à partir des types non structurés integer, real, boolean, char et enumeration type.

Les chaînes peuvent être manipulées mais c'est difficile !

Pascal a été adopté par les Universités dans les années 70 pour enseigner l'informatique.

Et mi 80 c'est le langage enseigné pour les étudiants. C'est le langage qui introduit un ensemble complet de structures de données et qui encourage un bon style de programmation.

Pascal a influencé les autres langages.

Wirth conçut ensuite Modula puis Modula-2, c'était l'introduction du concept des modules, une extension de Pascal.

Pascal a profondément influencé Ada.

d) Langages de traitement de données de gestion

A l'origine, on a des « computers » pour le calcul mais très vite on a vu qu'on pouvait manipuler des symboles.

Le problème, c'est qu'à l'époque, il n'y avait pas de notation de gestion et de manipulation (comme en mathématique). On a pensé à la langue anglaise pour les notations mais une langue naturelle est trop ambiguë.

1955 : Flow-Matic a conduit au COBOL implémenté sur Univac 1.

COBOL

(Common Business Oriented Language) fin des années 50 conçu par des représentants américains de constructeurs.

Le DoD a une grande influence sur son succès. Les contrats devaient utiliser les programmes en COBOL. DoD en a fait un de facto standard bien avant sa définition standard en 1968.

Importance du traitement de fichier et de données.

Les calculs sont très simples.

COBOL est verbeux si bien que les programmeurs et les managers peuvent lire les programmes écrits en Cobol.

Cobol n'était pas innovant mais a introduit l'idée de base de données : séparation entre la description de données, l'environnement physique dans lequel le calcul est réalisé et le calcul réellement effectué sur les données.

La description logique des données du problème peut être décrit indépendamment des caractéristiques physiques du support physique sur lesquels elles sont stockées et manipulées.

L'informatique est divisée en 2 mondes : à Cobol la gestion, à Fortran le calcul scientifique.

Langages de 4ème génération

1ère génération : Code machine

2ème génération : Assembleur

3ème génération : Langages de haut niveau Fortran, Cobol, Pascal, assez indépendants du hardware.

4ème génération : Le but des programmes est d'être orienté vers le problème à résoudre.

Les Langages de 4ème génération ont été développés à cause du mécontentement devant

Cours d'Ateliers de Génie Logiciel

Cobol (informatique de gestion) : Report Program Generator (RPG) dans les années 60.

D'autres problèmes exigent l'utilisation des bases de données.

Les systèmes de base de données utilisent SQL (Structured Query Language) : le langage est caché par une interface.

Les langages comme Java et Delphi incluent des bibliothèques avec des classes qui supportent des appels à des commandes SQL, si bien que ces langages peuvent être utilisés avec des bases de données et avec un éventuel accès distant sur Internet.

e) Langages généraux

PL/I

Dans les années 60

IBM était en avance et inventa l'IBM 360 avec le PL/I.

Les principes sont :

- ✓ Économiser le temps du programmeur
- ✓ Unité pour réconcilier la gestion et le scientifique
- ✓ Un langage très complet

Derrière PL/I, il y a cette idée de « default ». Chaque attribut, option d'une variable a une valeur par défaut, chaque spécification a une interprétation par défaut.

Venaient de Fortran et de Cobol mais aussi d'autres langages, le traitement des listes, les structures de contrôle et les méthodes de management de stockage.

Il y avait aussi du nouveau : le traitement des exceptions.

PL/I essayait d'être efficace et flexible : la pénalité était la complexité.

f) Développement interactif de programmes

Autrefois les ordinateurs étaient mono-tâches.

Les vitesses relatives différentes des entrées-sorties, du processeur central et des mémoires permettent le temps partagé pour utiliser au mieux les éléments de l'ordinateur.

Hypertrophie du processeur central avec terminaux distants reliés à la machine centrale.

Commença en 60s jusqu'à mi 70s. Les langages ont évolué et les compilateurs étaient aussi bien dans les stations de travail que dans l'ordinateur central. Le programme devient un fichier : gestion interactive, compilation, gestion des erreurs.

Cours d'Ateliers de Génie Logiciel

Certains langages permettent un développement interactif (langages fonctionnel et logique).

Le langage BASIC

Beginner's All Purpose Symbolic Instruction Code

Développé à Dartmouth College par Kemeny et Kurtz mi 60s pour les débutants, facile à apprendre et à se souvenir et à traduire, on n'oublie jamais.

Succès qui a surpris : langage simple pour les micro-ordinateurs des années mi 70.

Visual Basic : évolution vers l'orienté objet pour les applications avec une interface graphique sous Windows.

g) Langages spécifiques

Langage de manipulation de chaînes

COMIT (MIT 57-61, puis SNOBOL (Bell Labs) et SNOBOL4 (67).

Influence sur les éditeurs de texte : allocation dynamique de mémoire pour changer une chaîne de caractères par une chaîne plus grande.

Langages de traitement de Listes

En Fortran et Algol la seule structure de données est l'Array du même type (et de taille fixe en Fortran).

Un élément est divisé en une partie information et une partie pointeur.

Lisp

John McCarthy (fin des années 50)

Langage utilisable dans les débuts des années 60.

Plutôt pour les applications d'Intelligence Artificielle

Lisp est non typé.

Gestion de la mémoire.

C'est un langage fonctionnel.

Lisp a survécu et est toujours utilisé dans les systèmes experts.

Emacs (l'éditeur de texte) est écrit en Lisp.

Langages de simulation

La simulation des systèmes a été une des premières applications de l'ordinateur.

Ces systèmes sont modélisés par une série de changements d'états qui arrivent souvent en parallèle. Il peut y avoir des interactions entre les divers éléments.

Ces simulations servent à prédire le comportement, par exemple, simulation de trafic pour faire apparaître des goulets d'étranglements dans un réseau.

GPSS (General Purpose Simulation System) en 61.

Simula 67

Début des années 60 de Ole-Johan Dahl & Kristan Nygaard (Norvège).

Simula basé sur Algol 60 avec le concept de classe.

Il est possible de déclarer une classe et de générer des objets à partir de cette classe.

Ceci est la base de la programmation objet.

Simula est le père des langages orientés objet.

Le concept de classe a été repris dans C++, Ada, Smalltalk, Eiffel et Java.

Langages de Script

Langage awk

Langage Perl

Autres langages de script : Python, Tcl, JavaScript.

h) Langages de programmation systèmes

BCPL fin des années 60.

Maintenant c'est le C.

Le Langage C

Le paradigme d'Unix est : « tout est fichier ». Unix banalise le matériel (une imprimante est un fichier /dev/lpr) et le logiciel par la notion de fichiers. Un fichier est un ensemble de caractères. Langage C : évolution de BCPL en B (69-73) utilisé pour implémenter l'OS UNIX.

Cours d'Ateliers de Génie Logiciel

Unix et C développé aux Bell Labs par Ritchie et Thompson.

Sa structure est un mélange d'Algol et de Fortran.

C++ version orientée objet par Stroustrup au début des années 80.

i) Modules, classes, types de données abstraites, objets

Module et types de données abstraites

Année 70 : idée de module pour maintenir les programmes.

Besoin de réutiliser les logiciels et de construire des logiciels à partir de composants.

La modularité : décomposition physique en fichiers, décomposition logique pour éléver le niveau d'abstraction avec laquelle les programmeurs conçoivent leurs programmes.

Un problème peut être spécifié comme un ensemble de types de données abstraits.

Modula 2 et Ada : les modules structurent le programme (le package en Ada sert à définir des types abstraits de données).

La programmation objet a décollé dans les années 80 et est à la fin des années 90 le paradigme dominant.

Small talk pour le prototypage mais marche industriellement pour l'implémentation d'interface graphique.

C++ a ajouté des caractéristiques orientées objet à C.

Eiffel a eu une grande influence. C'est un langage purement objet qui a pour but les grands systèmes et la syntaxe est basée sur Pascal.

La position des langages orientés objet s'est renforcée avec Java.

En fait Eiffel, Delphi, C++ et Java sont responsables de la conversion des programmeurs en procédural à une approche orientée objet.

Objet

Simula a été le premier langage de programmation à implémenter le concept de classes en 67.

En 76, Smalltalk implémente les concepts d'encapsulation, d'agrégation, et d'héritage (les principaux concepts de l'approche objet).

Un objet est caractérisé par plusieurs notions:

Cours d'Ateliers de Génie Logiciel

- ✓ Les attributs
- ✓ Les méthodes
- ✓ L'identité

Classe

On appelle *classe* la structure d'un objet, c'est-à-dire la déclaration de l'ensemble des entités qui composeront un objet. Un objet est issu d'une classe. En réalité on dit qu'un objet est une instantiation d'une classe, c'est la raison pour laquelle on pourra parler indifféremment d'*objet* ou d'*instance* (éventuellement d'*occurrence*).

ADA

DoD le plus grand utilisateur des ordinateurs de l'époque.

Dans les années mi 70, le DoD a voulu favoriser et financer le développement d'un nouveau langage.

700 langages pour la mission Apollo.

Volonté d'avoir un seul langage de programmation standard pour les différents systèmes embarqués.

Le langage Ada a été publié en 83.

Ada est basé sur Pascal mais est beaucoup plus gros.

Ada contient la notion de bibliothèques et packages pour assembler des composants sans connaître l'intérieur.

Ada est un langage fortement typé.

Traitement des exceptions.

Le compilateur Ada est gros.

Le langage a eu du mal à être adopté par les programmeurs : difficile face à la compétition de C et ensuite C++.

Ada a été obligé de subir une révision en 95 pour des extensions objets (passage de Ada 83 à Ada 95).

Modula 2

Modula 2 conçu par Wirth fin des années 70.

Cours d'Ateliers de Génie Logiciel

Pour étendre Pascal avec le concept de module.

Moins compliqué que Ada, il était utilisé pour l'enseignement.

En compétition avec C++ et Ada puis récemment Java.

Smalltalk

Alan Kay : idée de fournir un PC à des non experts pour communiquer à travers une interface graphique conviviale.

Développé à Palo Alto par Xerox début 70 sur des stations puissantes en graphique.

Plusieurs versions dont Smalltalk80.

Dispose de menus, de fenêtre d'icône d'entrée par la souris.

Dans les années 80, Smalltalk était « le » langage orienté objet.

Tout dans Smalltalk est objet.

La philosophie de Smalltalk est que les ressources informatiques ne sont pas chères et que le temps humain est en revanche précieux. Valable pour l'informatique aujourd'hui.

Eiffel

Bertrand Meyer dans les années 80s.

Fournir un langage qui supporte la création de grands systèmes robustes à partir de composants testés et réutilisables.

Eiffel est un pur langage orienté objet.

Le type des objets est défini par des classes.

C++

Langage efficace pour ajouter à C le concept de classe de Simula.

Développé à Bell Labs par Stroustrup, la patrie de C.

A commencé en 79 la 1^{ère} version C avec des Classes.

C++ a été commercialisé en 85.

Ce n'est pas un langage orienté objet pur mais il supporte les caractéristiques de l'orienté objet.

Cours d'Ateliers de Génie Logiciel

C++ est un gros langage (il garde les capacités de C pour le bas niveau de programmation).

C++ a été à la base de Java.

Delphi

Extension de Pascal orienté objet Object Pascal.

Delphi est le développement de Object Pascal avec une extension d'interface graphique, de base de données et de programmation Internet.

Delphi est un RAD (Rapid Application Development) avec un environnement de développement graphique.

Java

La diffusion de Java a été impressionnante.

A débuté à Sun Microsystems en 90-91 (Oak) pour concevoir un langage petit pour les appareils de la maison.

Avec le développement du Web, le langage a été reciblé pour le langage de l'Internet et a changé de nom (Java) en 95.

Idée d'applet (petite application) pour animer des pages web : a été un succès immédiatement.

Est devenu le langage pour l'enseignement dans toutes les universités.

Java joue un rôle fondamental en faisant de l'orienté objet le paradigme dominant pour les langages.

C'est un petit langage et qui a un très grand nombre de bibliothèques.

Dans les autres langages, les bibliothèques sont un extra. Ici pour Java, c'est au centre du langage.

Java est conçu comme une simplification de C++ : les programmeurs de C++ l'ont adopté très facilement.

Dans Java, il y a un ensemble de Classes de bibliothèque qui permettent de faire des interfaces graphiques.

Java est portable (important pour Internet).

Java est transformé en un langage intermédiaire le JavaBytecode de telle manière qu'il puisse être interprété par l'Interpréteur.

Cours d'Ateliers de Génie Logiciel

JavaBytecode est un langage machine pour une Java Virtual Machine.

L'environnement de développement, le Java Development Kit peut être téléchargé gratuitement à partir d'Internet.

Java diffère de C++

C'est un langage purement orienté objet

Il est moins performant en terme de rapidité

Toutes les opérations sont parties d'une classe

j) Langage Fonctionnel et Logique

ML

Fins des années 70 et début 80 ML et Hope ont vu le jour.

Ils ont fusionné et donné Standard ML.

ML est fortement typé.

Prolog

PROgramming in LOGic.

A été créé à l'université de Marseille en 73.

Les développements ont été faits à Marseille et Edimbourg fin 70.

L'approche a été de programmer des problèmes d'Intelligence Artificielle.

L'intérêt de Prolog a été fort dans les années 80 en particulier quand les Japonais ont lancé leur initiative sur la 5^{ème} génération des ordinateurs.

Bataille entre Prolog (logique) et Lisp (fonctionnel) pour résoudre le même genre de problèmes.

Lisp est utile quand on traite des listes, Prolog est bien pour des applications de recherche avec des bases de données.

Prolog est adapté aux stratégies d'essais et erreurs.

I. 9 – Syntaxe et Sémantique

a) Syntaxe BNF

Ce qui est légal et ce qui ne l'est pas.

Syntaxe (grammaire) : décrit la forme correcte dans laquelle les programmes doivent être écrits.

BNF (Backus-Naur Form (Backus-Normal Form)) métalangage : langage pour définir un autre langage (utilisé pour la 1^{ère} fois avec Algol 60)

b) Sémantique

Sémantique est la signification qui est attachée aux différentes constructions syntaxiques.

Peu de progrès.

Sémantique ISO et ANSI encore en anglais.

La sémantique d'un programme est définie par les règles qui dictent comment un programme est écrit dans ce langage et comment il peut être exécuté par cette machine abstraite.

Fait pour les concepteurs de langage, pas pour les implémentateurs.

I. 10 – Structure des programmes

Dans un langage avec des blocs structurés comme Pascal, les procédures sont déclarées dans un environnement créé par des blocs fermés. Un appel de procédure est une instruction de plein droit tandis qu'un appel de fonction fait partie d'une expression et redonne une valeur.

Dans un langage procédural, une procédure peut exister comme une entité séparée, tandis que dans un langage purement orienté objet, une procédure est toujours un composant d'une classe.

Pour résoudre un grand problème, il faut le découper en parties distinctes qui interagissent entre elles uniquement à travers des interfaces strictement définies. Pour s'assurer qu'un programme sera modifiable facilement, il doit être conçu de telle manière que les changements soient localisés. Ceci est réalisé par l'utilisation de modules.

Un module est une collection de déclarations de variables, type et procédure en relation.

Un module en Ada ou Fortran 90 peut contenir la définition d'un type.

Un module (une classe) en C++ et Java est la définition d'un type.

Cours d'Ateliers de Génie Logiciel

Un module est généralement découpé en 2 parties : la partie publique décrit ce que le module fait alors que l'implémentation privée donne les détails de comment les opérations sont faites.

En Ada, ces deux parties sont la spécification et le corps (body) du package.

Les modules supportent une information cachée. Les identifiants déclarés dans l'interface peuvent être utilisés par d'autres modules alors que les identifiants déclarés dans l'implémentation sont cachés. La possibilité de cacher des détails est un outil majeur pour contrôler la complexité.

Un type est caractérisé par sa portée de valeurs et les opérations qui peuvent être exécutées sur les objets de ce type.

Les packages en Ada et les classes en C++ et Java peuvent être utilisées pour définir de nouveaux types.

Les opérations permises sont données dans la partie de l'interface alors que l'implémentation des opérations et la représentation du type sont cachées. Les types définis ainsi sont des types de données abstraits.

a) Procédures, fonctions, méthodes

En Pascal, C, Java, le mécanisme de passage de paramètres d'un appel par valeur est utilisé pour transmettre l'information à un sous-programme.

Dans un appel par valeur, le paramètre formel agit comme une variable locale.

Les appels par valeur sont coûteux en terme d'espace et de temps quand le paramètre est un objet structuré.

Le seul effet d'un appel de fonction sur le reste du programme se fait à travers le résultat de la fonction.

b) Structures de données

Un choix judicieux de structures de données appropriées est crucial dans la conception et la production de programmes bien structurés.

Tous les composants d'un array ont le même type et sont accédés par un indice calculable.

En C et C++ les pointeurs peuvent être utilisés pour accéder à des éléments de tableaux.

En Pascal (et ses successeurs), un tableau à 2 dimensions est un tableau de tableaux.

Pascal est fortement typé.

Les classes en C++ ont des parties fonctions et des parties données. Les classes permettent que les opérations sur les structures de données soient incorporées avec la définition des structures de données.

La représentation ou la manipulation de chaîne de caractères est possible si la conception du langage permet des opérations qui potentiellement augmentent la longueur de la chaîne.

I. 11 – Langages de 4^{ème} et 5^{ème} Génération

Un langage de programmation de cinquième génération, abrégé L5G, est un langage de programmation basé sur le concept de résolution de problèmes en utilisant des contraintes données au programme, plutôt que d'utiliser un algorithme écrit par un programmeur (3^e génération) ou une spécification formelle de représentation en tables et interrogations SQL (4^e génération). La plupart des langages fonctionnant par contraintes ou par programmation logique ainsi que quelques langages déclaratifs sont des langages de cinquième génération.

Les langages de programmation de quatrième génération (L4G) (4GL en anglais) sont un type de langage de programmation apparu en 1980, proche des langues naturelles, qui permet d'écrire plus de choses avec moins de lignes de programmes et moins d'erreurs.

Il n'existe pas de distinction formelle entre les 3^e et 4^e générations de langages de programmation. Selon Holger Herbst en 1997, la plupart des L4G sont en fait des langages de 3^e génération auxquels a été ajoutée une syntaxe déclarative basée sur SQL. De nombreux logiciels de base de données tels que dBase, Oracle ou Informix incluent un langage de programmation de 4^e génération. Celui-ci est similaire à un langage généraliste tel que Pascal et comporte des constructions additionnelles qui permettent un lien étroit avec le moteur de base de données telles que des commandes permettant de manipuler les bases de données.

Les langages de programmation de quatrième génération (L4G) transforment, avec plus ou moins d'efficacité selon les fournisseurs, des requêtes SQL normalisées en algorithmes, contrairement à ceux de troisième génération où le programmeur codait dans le détail ses propres algorithmes. Ces L4G mettent l'accent sur l'usage d'une (ou plus) base de données relationnelle, souvent avec interface graphique....

Les langages de cinquième génération prennent le problème encore en amont : idéalement, que l'ordinateur puisse résoudre n'importe quel problème juste formulé par ses contraintes et son objectif. L'utilisateur - il ne s'agit déjà plus d'un programmeur - a juste besoin (idéalement toujours) de recenser ce qui doit être fait, et sous quelles contraintes, sans rentrer dans le détail, et doit obtenir une ou des solutions correspondantes.

Les systèmes de cinquième génération utilisent une approche métier, et leurs internes sont programmés dans des langages comme Prolog, OPS5, Mercury, Haskell, etc.

I. 12 – Performance et popularité

a) Performance

Six chercheurs de trois universités portugaises ont mené une étude comparative de 27 langages de programmation, intitulée « Energy Efficiency Across Programming Languages ». Ils ont étudié la consommation d'énergie, le temps d'exécution et l'utilisation de la mémoire. Pour obtenir un ensemble de programmes comparables, les chercheurs ont exploré le Computer Language Benchmarks Game (CLBG).

Le tableau obtenu présente les résultats globaux (en moyenne) pour la consommation d'énergie (Energy), le temps d'exécution (Time) et la consommation maximale de la mémoire (Mb) normalisés par rapport au langage le plus efficace pour le critère mesuré.

Les cinq meilleurs langages sont :

Pour la consommation d'énergie

C : 1,00

Rust : 1,03

C++ : 1,34

Ada : 1,70

Java : 1,98

Pour le temps d'exécution

C : 1,00

Rust : 1,04

C++ : 1,56

Ada : 1,85

Java : 1,89

Pour la consommation maximale de mémoire

Pascal : 1,00

Go : 1,05

C : 1,17

Fortran : 1,24

C++ : 1,34

b) Popularité

La popularité de chaque langage est difficilement quantifiable ; néanmoins, il existe l'index TIOBE, calculé mensuellement, qui se base sur le nombre de formations/cours destinée aux ingénieurs et le nombre de revendeurs/free-lance spécialisés dans un langage de programmation. C'est une information parcellaire mais qui peut donner un ordre d'idée sur les tendances en matière de préférence des programmeurs.

I. 13 – Conclusion

Bien apprendre et maîtriser 2 langages : par exemple C et Java (au minimum).

Utiliser et maîtriser si possible une grande palette des possibilités du langage (puissance d'expression).

Écrire des programmes simples, clairs, lisibles et capables d'être repris et retravaillé par quelqu'un d'autre.

Ne pas se lancer dans l'optimisation à tout crin, car les compilateurs font un travail excellent en matière d'optimisation.

Ce qui se conçoit bien se programme clairement et les instructions pour sa réalisation et son exécution s'écrivent aisément ...

Chapitre II - Les Ateliers de Génie Logiciel

II. 1. Introduction

Qu'est-ce qu'un logiciel ?

“Le logiciel est l’ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d’un ensemble de traitement de l’information” (arrêté du 22 déc. 1981)

Autrement dit, et de façon plus générale, un logiciel est un ensemble de programmes informatiques (du code) mais également un certain nombre de documents se rapportant à ces programmes et nécessaires à leur installation, utilisation, développement et maintenance : spécifications, schémas conceptuels, jeux de tests, mode d'emploi, ...

La « crise du logiciel »

La “crise du logiciel” est apparue à la fin des années 60 et provient d'un décalage entre les progrès matériels d'une part et logiciels d'autre part : alors qu'apparaissaient les ordinateurs de la troisième génération, de plus en plus puissants et de moins en moins coûteux, la construction de logiciels restait dans le domaine de l'artisanat et du folklore, où chacun y allait de sa petite recette. De fait, alors que les nouvelles machines rendaient possibles des applications jusqu'alors irréalisables, les méthodes de développement logiciel ne s'appliquaient pas à de grands systèmes :

- ❖ la construction de logiciels coutait très cher (200 millions de dollars pour fabriquer OS-360),
- ❖ les délais n'étaient pas respectés (2 ans de retard pour les premiers compilateurs PL/1, Algol 68, ADA),
- ❖ les logiciels n'étaient pas évolutifs (parfois écrits en assembleur pour un type de machine) ce qui les rendait très rapidement obsolètes,
- ❖ avec des performances poussives (Univac, le système de réservation pour United Air Lines au début des années 75 n'a jamais servi car les temps de réponse étaient trop longs !),
- ❖ une fiabilité aléatoire (la sonde américaine qui devait aller sur Vénus s'est perdue, à cause d'une mauvaise instruction... plus récemment, la trajectoire de Ariane 5 a été modifiée à cause d'un débordement de capacité),
- ❖ et une convivialité discutable (des interfaces homme/machine inexistantes).

Le génie logiciel

“Le génie logiciel est l’ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi” (arrêté du 30 déc. 83)

Cours d'Ateliers de Génie Logiciel

Autrement dit, le génie logiciel est ``l'art'' de produire de bons logiciels, au meilleur rapport qualité/prix. Il utilise pour cela des principes d'ingénierie et comprend des aspects à la fois techniques et non techniques : le génie logiciel est basé sur des méthodologies et des outils qui permettent de formaliser et même d'automatiser partiellement la production de logiciels, mais il est également basé sur des concepts plus informels, et demande des capacités de communication, d'interprétation et d'anticipation. De fait, la ``crise du logiciel'' n'est toujours pas résolue. Le génie logiciel reste un ``art'' qui demande de la part de l'informaticien une bonne formation aux différentes techniques (le ``savoir''), mais également un certain entraînement et de l'expérience (le ``savoir-faire'').

Les qualités d'un logiciel

Si le génie logiciel est l'art de produire de bons logiciels, il est nécessaire de fixer les critères de qualité d'un logiciel. On peut séparer ces qualités en deux catégories, suivant que l'on regarde le logiciel de l'extérieur ou de l'intérieur:

- ❖ les qualités du logiciel lors de l'utilisation : fiabilité (correction et robustesse), adéquation aux besoins (y compris aux besoins implicites !), ergonomie (simplicité et rapidité d'emploi, personnalisation), efficacité, convivialité, ..., faible coût et respect des délais bien entendu.
- ❖ les qualités du logiciel lors de la maintenance : un logiciel doit pouvoir être maintenu (pour le corriger, l'améliorer, l'adapter aux changements de son environnement, ...). Pour cela, il doit être flexible (utilisation du paramétrage, de la générativité, de l'héritage), portable (éviter l'assembleur et les langages trop confidentiels), structuré (utilisation de modules ou de classes, de procédures ou de fonctions) avec une indépendance maximum entre les structures (utilisation de l'abstraction), ... et bien sûr, documenté.

Ces différentes qualités ne sont pas toujours compatibles ni même réalisables, et il est nécessaire de trouver des compromis. Dans tous les cas, les objectifs de qualité doivent être définis pour chaque logiciel, et la qualité du logiciel doit être contrôlée par rapport à ces objectifs.

II. 2. Le processus logiciel, ou cycle de vie du logiciel

Le processus logiciel désigne l'ensemble des activités nécessaires au développement et à la maintenance d'un logiciel. Il s'agit d'un processus variable (selon le type d'application) et complexe, composé de différentes phases interdépendantes.

Afin de tenter de résoudre la crise du logiciel, ce processus a fait l'objet de différentes modélisations. Historiquement, le premier modèle de développement proposé est celui dit ``de la cascade'', au début des années 70. Ce modèle a été assez largement mis en œuvre,

mais on s'est rapidement aperçu qu'il n'est pas toujours approprié. Sa vision simpliste du processus sous-estime le coût des retours en arrière dans le cycle de vie. Ainsi, plusieurs alternatives au modèle de la cascade ont été proposées, basées notamment sur le prototypage et l'assemblage de composants réutilisables.

Le modèle de la cascade est bien adapté au processus logiciel s'il y a peu de retours en arrière. Dans ce modèle, l'élaboration des spécifications est une phase particulièrement critique : les erreurs de spécifications sont généralement détectées au moment des tests, voire au moment de la livraison du logiciel à l'utilisateur. Leur correction nécessite alors de reprendre toutes les phases du processus.

Une difficulté majeure de la phase de spécification provient du fait que les différents partenaires ne parlent généralement pas le même langage :

- ❖ le client s'exprime dans le langage du domaine de l'application, un jargon qui fait souvent appel à des termes techniques très spécialisés, mais qui utilise pour support le langage naturel (le français, l'anglais). De plus, le client ne sait pas toujours précisément ce qu'il veut : sa demande peut varier en fonction de l'offre. De fait, les désirs du client sont généralement ambigus et incomplets.
- ❖ l'informaticien fait des petits schémas, utilise des langages formels pour représenter sa perception du problème. Le client non informaticien n'est généralement pas capable de comprendre la spécification résultante, et ne peut donc la valider en connaissance de cause.

Une solution, pour vérifier la conformité de la spécification avec les besoins du client, est de construire rapidement un prototype de l'application.

L'intégration d'une phase de prototypage dans le processus logiciel peut s'effectuer de deux façons différentes :

- ❖ l'approche "prototypage jetable" consiste à réaliser rapidement et dès le début du cycle de vie un prototype de l'application qui va permettre de valider les spécifications. Ce prototype sert alors de référence pour la définition des spécifications, puis il est "jeté" et le programme définitif est conçu.
- ❖ l'approche "développement incrémental" consiste à réaliser dès le début du cycle de vie un sous-ensemble du produit logiciel final. Ce sous-ensemble est alors raffiné incrémentalement jusqu'à obtenir le produit final.

II. 3. Ateliers de Génie Logiciel

Qu'est-ce qu'un atelier de génie logiciel ?

Un AGL (Atelier de Génie Logiciel) ou atelier CASE (Computer Aided Software Engineering) est un logiciel aidant à la réalisation de logiciels. Autrement dit, il s'agit d'un système pour le développement logiciel assisté par ordinateur. Un AGL intègre des outils adaptés aux différentes phases de la production d'un logiciel et facilite la communication et la coordination entre ces différentes phases. Un AGL est basé sur des méthodologies qui formalisent le processus logiciel, et à l'intérieur de ce processus, chacune des phases qui le composent.

Les AGL apportent une réelle solution à certains problèmes du génie logiciel et contribuent nettement à l'amélioration de la productivité et de la qualité du logiciel, notamment en faisant le suivi des différentes phases du processus logiciel et en offrant un cadre cohérent et uniforme de production. Néanmoins, cet enthousiasme doit être modéré : le processus logiciel est encore loin d'être maîtrisé et les différentes formalisations qui en sont proposées font encore l'objet de controverses, et dans tous les cas, sont bien loin d'être totalement automatisables. L'informaticien a encore de belles années de travail devant lui avant d'être supplanté par des AGL...

Les outils "CASE"

Les AGL intègrent différents outils d'aide au développement de logiciels, appelés outils CASE : éditeurs de texte (vi, emacs, ...), de diagrammes (TRAMIS VIEW, X-fig, ...), outils de gestion de configuration (make), SGBD, compilateurs, debuggers, outils pour la mise en forme (pretty-printers), la génération de tests, la génération d'interfaces homme-machine, ...

Ces différents outils interviennent lors d'une ou plusieurs phases du cycle de vie du logiciel : conception (éditeurs de texte, de diagrammes, ...), programmation (éditeurs de texte, compilateurs, pretty printers, générateurs d'interfaces homme/machine...), mise au point (debuggers, outils de génération de tests, ...), etc., ... Certains outils, concernant notamment la gestion de configurations, la gestion de projet, interviennent durant la totalité du processus logiciel.

L'intégration d'outils CASE

Un AGL intègre différents outils CASE, de manière à les faire coopérer de façon uniforme. Cette intégration peut (devrait) s'effectuer à trois niveaux :

Intégration des données : Les outils CASE manipulent (génèrent, utilisent, transforment, ...) des données : spécification, modèle conceptuel des données, jeux de test, code, manuel utilisateur, Différents outils sont amenés à partager une même donnée : les tables générées par un éditeur de diagrammes sont utilisées par un SGBD, le code généré par un éditeur de

texte est compilé par un compilateur, à partir d'une spécification algébrique on peut générer des jeux de test, ...

Un AGL doit prendre en charge la communication de ces données entre les différents outils. Cette intégration peut être simplement physique : tous les outils de l'AGL utilisent un seul format de représentation des données, par exemple des fichiers UNIX, sur une même machine. Cette approche implique que tous les outils de l'AGL connaissent la structure logique (l'organisation) des fichiers qu'ils sont amenés à utiliser : il est nécessaire de normaliser la structure logique des fichiers. L'intégration des données peut se faire également au niveau logique en utilisant un système de gestion des objets qui gère automatiquement les différentes entités et leurs interrelations (cette approche nécessite la définition des différents types de données manipulées).

Un AGL devrait également gérer la cohérence entre les différentes versions de ces données (gestion de configuration).

Intégration de l'interface utilisateur : tous les outils intégrés dans l'AGL communiquent avec l'utilisateur selon un schéma uniforme, ce qui facilite leur utilisation (voir par exemple l'interface du Macintosh, X11 sous Unix ou Windows95 sous DOS).

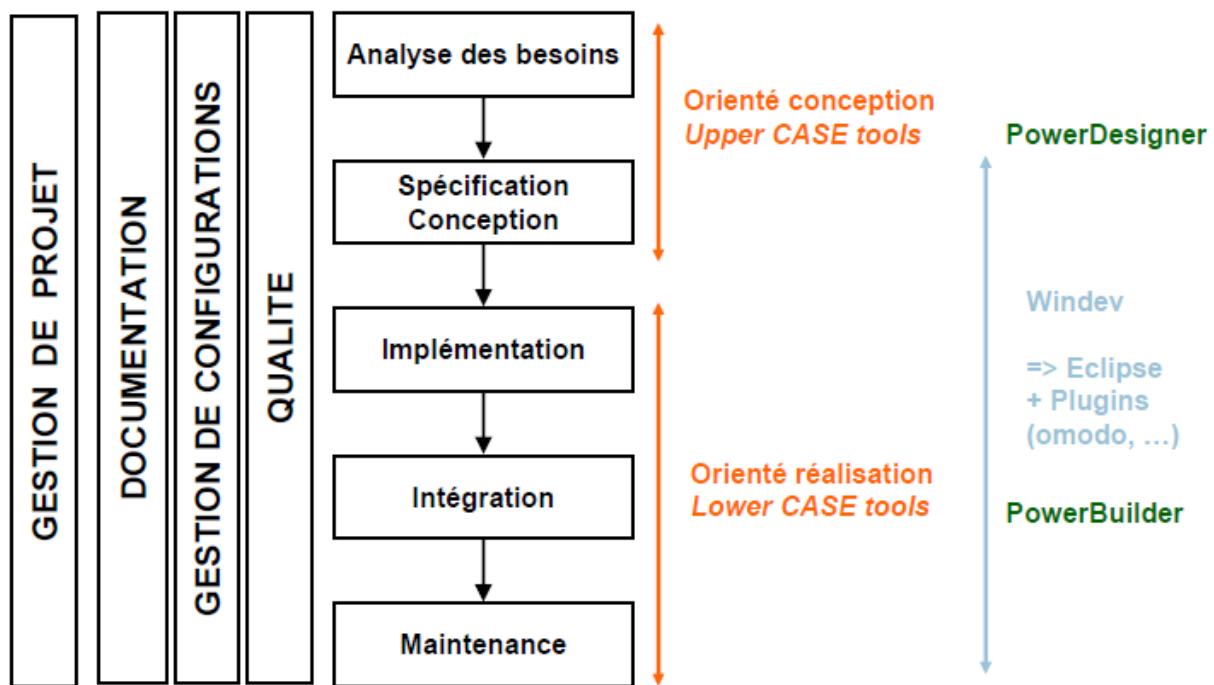
Intégration des activités : un AGL peut gérer le séquencement des appels aux différents outils intégrés, et assurer ainsi un enchainement cohérent des différentes phases du processus logiciel. Cet aspect implique que l'on dispose d'un modèle du processus de développement bien accepté (ce qui relève un peu de l'utopie !!!).

Catégories d'AGL

Les AGL peuvent être classés selon plusieurs aspects :

- ❖ Richesse du support : ensemble d'outils, outils intégrés, aide à la démarche.
- ❖ Type de problèmes : logiciels embarqués, temps réel, ...
- ❖ Type de projet d'ingénierie logicielle : développement logiciel (cf. cycle de vie), intégration de systèmes.
- ❖ Ampleur du projet : complexité, nombres de participants, durée ...
- ❖ Gestion des ressources du projet : les considérations managériales des ressources mises en œuvre dans le projet sont-elles prises en compte ? (planification, ordonnancement, ...).
- ❖ Phase du cycle de développement prises en compte : conception et/ou développement.

Classification basée sur le cycle de développement



On distingue essentiellement deux types d'AGL selon la nature des outils intégrés :

Les environnements de conception (upper-case) : ces ateliers s'intéressent plus particulièrement aux phases d'analyse et de conception du processus logiciel. Ils intègrent généralement des outils pour l'édition de diagrammes (avec vérification syntaxique), des dictionnaires de données, des outils pour l'édition de rapports, des générateurs de (squelettes de) code, des outils pour le prototypage, ... Ces ateliers sont généralement basés sur une méthode d'analyse et de conception (JSD, Yourdon, Merise, ...) et utilisés pour l'analyse et la conception des systèmes d'information.

TRAMIS est un environnement de conception qui intègre notamment un éditeur de diagrammes (TRAMIS View), un générateur de prototypes (TRAMIS Dialog), ...

Les environnements de développement (lower-case) : ces ateliers s'intéressent plus particulièrement aux phases d'implémentation et de test du processus logiciel. Ils intègrent généralement des éditeurs (éventuellement dirigés par la syntaxe), des générateurs d'interfaces homme/machine, des SGBD, des compilateurs, optimiseurs, pretty-printers, debuggers, ...

WinDev est un environnement de développement.

Un autre exemple d'environnement de développement est Unix qui intègre différents outils pour la programmation et le test. L'intégration des données est faite par l'intermédiaire des fichiers Unix, la gestion (limitée) de configurations est faite par make...

Cours d'Ateliers de Génie Logiciel

Certains environnements, plus évolués, sont dédiés à un langage particulier. Il existe par exemple des environnements dédiés à InterLisp, Smalltalk, Loops (l'environnement Loops fonctionne sur une machine dédiée à Loops), Oz ... Ces différents environnements proposent des bibliothèques de composants, une interface graphique, des éditeurs dédiés au langage, des interprètes, debuggers, ... Ces environnements permettent un développement rapide et convivial. En revanche, l'application développée est intégrée dans (et généralement inséparable de) l'environnement, ce qui peut poser des problèmes de portabilité et de coût.

Enfin, il existe des générateurs d'environnements de programmation : Mentor, Gandalf, Synthesizer Generator, ... A partir de la description formelle d'un langage (à l'aide de grammaires attribuées, de la logique), ces différents systèmes génèrent un environnement de programmation dédié au langage, contenant un éditeur dédié au langage, un pretty-printer, un debugger, un interpréteur, ...

Critères d'adoption d'un AGL

Choisir d'utiliser un AGL pose certains questionnements :

Investissement de ressources

→Coût d'adoption d'une technologie AGL.

Aide et Support technique disponible

→évaluation à long terme de l'exploitation du logiciel

Méthodes et processus de GL existants dans l'entreprise

→adéquation entre ce qui est fait par les 'acteurs' et ce qui est proposé par les outils

Montée en charge

→aussi bien en terme d'ampleur du projet que de la performance des applications générées avec l'outil.

Evaluation de la valeur réelle

→ écart plus ou moins grand avec les arguments commerciaux/marketing.

Variété des standards

→ problème de la sélection et de la comparabilité des produits.

Complexité de l'adoption du produit

→ en terme d'utilisation mais aussi en terme de déploiement dans l'entreprise.

Quelques exemples d'AGL

- ✓ AD-Cycle d'IBM
- ✓ AGL Merise/2
- ✓ ArgoUML
- ✓ ARIS d'IDS-Scheer
- ✓ ASA & GODE
- ✓ Advantage Plex
- ✓ Adélia
- ✓ Application Builder d'Enablon
- ✓ BOUML
- ✓ Case*dictionnaire d'Oracle
- ✓ Clarion
- ✓ CodeFluent Entities de SoftFluent
- ✓ Corporate Modeler de Casewise
- ✓ DB-MAIN
- ✓ Dalyo
- ✓ DoMIS
- ✓ Enterprise Architect de Sparx Systems
- ✓ Envision SART
- ✓ Excelerator
- ✓ IntelliJ IDEA
- ✓ KDevelop
- ✓ Mega
- ✓ Method/1
- ✓ Modelio
- ✓ NATSTAR de Nat System
- ✓ Net Express de Micro Focus
- ✓ NetBeans
- ✓ NSDK de Nat System
- ✓ Objecteering
- ✓ Pacbase
- ✓ PACKflow de Mica-Systems
- ✓ Perceptory
- ✓ PowerAMC de Sybase
- ✓ Rational Rose d'IBM
- ✓ (en) Synon
- ✓ UNIFACE
- ✓ Visual Paradigm
- ✓ Visual Studio

- ✓ WEBDEV
- ✓ WINDEV
- ✓ WINDEV Mobile

II.4. Les SGBD avancés

Un **Système de Gestion de Base de Données** (SGBD) est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données.

Parmi les logiciels les plus connus il est possible de citer : MySQL, PostgreSQL, SQLite, Oracle Database, Microsoft SQL Server, Firebird ou Ingres.

A. Les Bases de Données réparties

1. Principe

Dans une base de données centralisée, il y a un seul SGBD, un seul stockage physique, une seule unité de traitement. Dans une base de données répartie, il peut y avoir plusieurs SGBD, plusieurs sites de stockage et plusieurs unités de traitement.

Les raisons qui président à la création des bases de données réparties sont multiples :

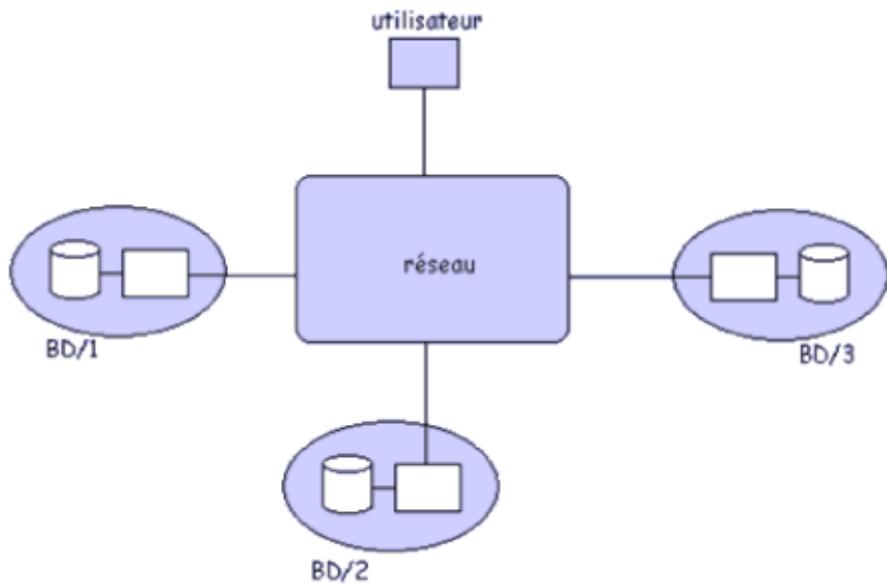
- ✓ limiter le transfert d'informations
- ✓ répartir la charge de travail entre plusieurs unités de traitement ou de stockage
- ✓ augmenter la fiabilité
- ✓ augmenter la disponibilité
- ✓ faciliter l'interopérabilité

Définition

Une base de données répartie est une base de données dont les différentes parties sont stockées sur des sites distants reliés par réseau.

Aspects caractéristiques

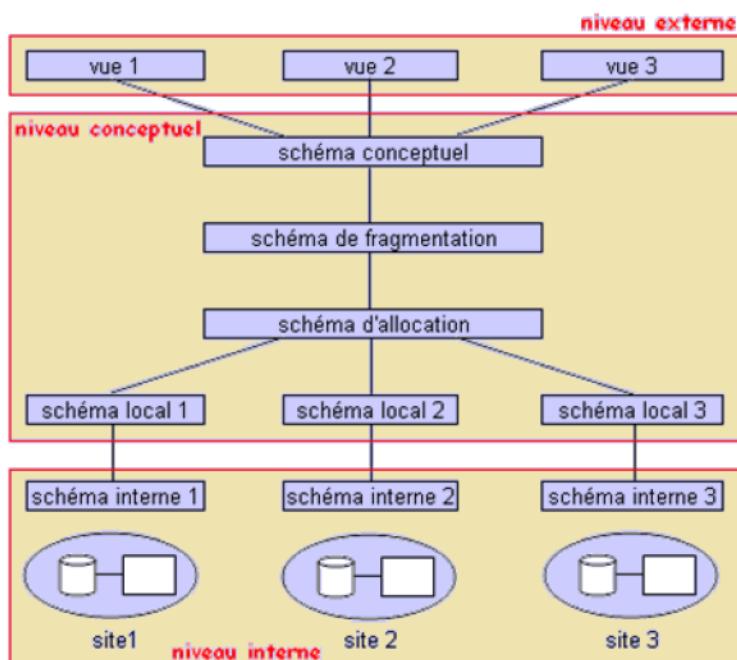
On peut représenter très schématiquement une base de données répartie par l'image suivante:



Cette base doit obéir à 3 principes de "transparence" :

- ✓ transparence de localisation : l'utilisateur accède au schéma conceptuel via des vues ; il ne sait pas sur quel site se trouvent physiquement les données
- ✓ transparence de partitionnement : l'utilisateur ne sait pas comment la base est partitionnée
- ✓ transparence de duplication : l'utilisateur ne sait pas s'il existe des copies des informations, ni où elles se trouvent si elles existent

Une base de données est usuellement modélisée en 3 niveaux : externe, conceptuel, interne.
Pour une base de données répartie, la répartition a lieu dans les trois niveaux :



- ✓ les vues des utilisateurs sont présentées sur leur site (site utilisateur) ; elles correspondent au niveau externe. Il y a donc répartition des vues.
- ✓ le schéma conceptuel (global) est associé aux schémas locaux des sites physiques via un schéma de fragmentation (la manière dont la base est découpée) et un schéma d'allocation (la manière dont les fragments sont répartis).
- ✓ il n'y a pas de schéma interne global, mais des schémas locaux internes.

Démarche de conception

Pour concevoir une base de données répartie, il faut commencer par les deux étapes standards de conception des bases de données centralisées.

- 1) Recueillir l'expression des besoins des utilisateurs et en déduire les vues externes à prévoir.
- 2) Intégrer ces vues dans un schéma conceptuel global et unique

Les deux phases suivantes sont des phases critiques et délicates pour des bases de données réparties :

- 3) Création du schéma de fragmentation : la base de données sera découpée en fragments distincts constituant une partition de la base : l'intersection des fragments doit être vide et leur réunion doit redonner le schéma global.
- 4) Création du schéma d'allocation : les fragments doivent être distribués "au mieux" entre les différents sites.

Les deux dernières phases concernent les sites locaux :

- 5) Création d'un schéma local pour chaque site, relatif aux fragments dévolus à ce site.
- 6) Création des schémas internes : implémentation des données des fragments sur les supports physiques de stockage.

2. Fragmentation

Décomposition en fragments

Il s'agit de décomposer la base de données en fragments sans perte d'information. On procède de la manière suivante :

- ✓ la non perte d'informations est vérifiée par reconstitution de la base en utilisant le langage de manipulation de données (SQL par exemple).
- ✓ les fragments doivent être exclusifs (il ne s'agit pas de duplication)

Pour découper la base, il faut descendre à un certain degré de granularisation aboutissant à des unités de fragmentation (considérées comme insécables). On peut procéder de 4 manières (éventuellement combinables) :

- ✓ la répartition par classes d'objets
- ✓ la répartition par occurrences (tuples) ou fragmentation horizontale
- ✓ la répartition par attributs ou fragmentation verticale
- ✓ la répartition par valeurs qui combine les deux précédentes.

Pour définir le schéma de fragmentation, il faudra

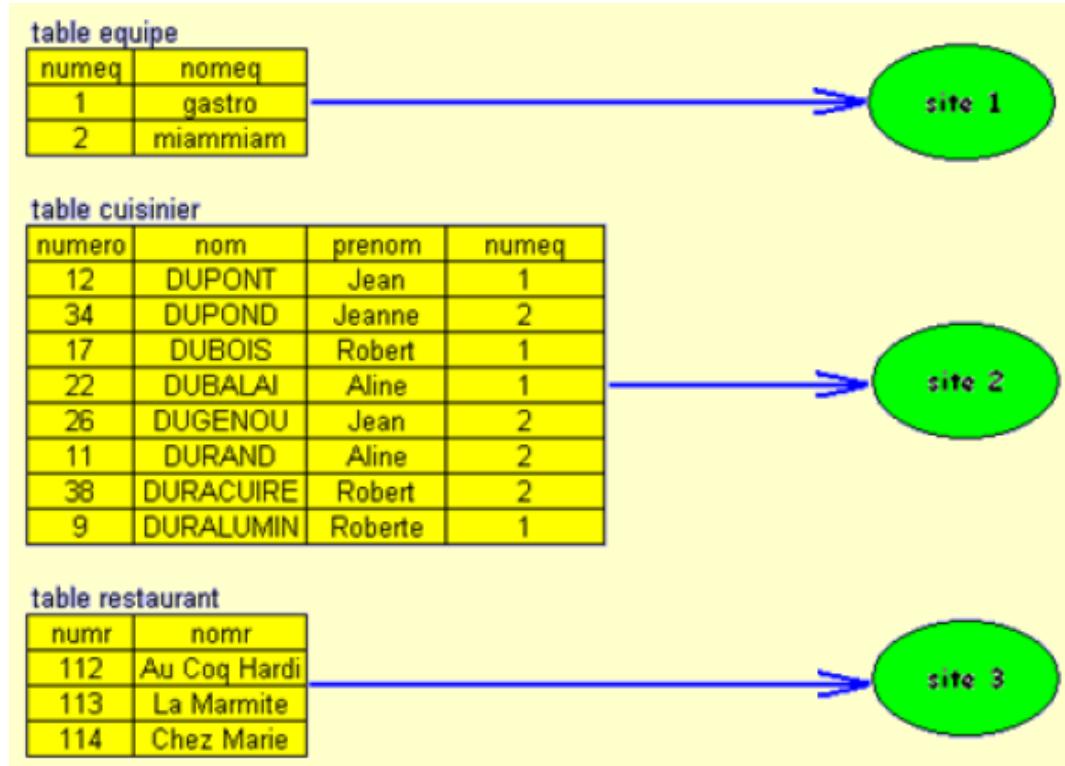
- ✓ définir les fragments, c'est à dire constituer un partitionnement de la base
- ✓ définir la correspondance schéma global --> fragments
- ✓ définir la correspondance fragments --> schéma global (recomposition)

Répartition par classes d'objets

Le mot "classe" est utilisé ici de manière très générique ; il peut désigner une "vraie" classe (modèle objet), une entité (modèle entité-association), une relation ou table (modèle relationnel), ...

Les fragments sont définis à partir des "classes" de la base de données.

Exemple : soit une base de données constituée de 3 tables EQUIPE, CUISINIER, RESTAURANT. Ces trois tables correspondront à trois fragments.



Fragmentation horizontale

Encore appelée fragmentation par occurrences, la fragmentation horizontale est basée sur un découpage des tuples des "classes".

Exemple : Considérons la table CUISINIER. Elle peut être divisée en deux fragments par répartition des tuples en deux catégories :

The diagram illustrates horizontal fragmentation of the CUISINIER table. On the left, the original table is shown with columns: numero, nom, prenom, and numeq. The data consists of 13 tuples. Two blue arrows point from this table to two separate fragments on the right.

table cuisinier

numero	nom	prenom	numeq
12	DUPONT	Jean	1
34	DUPOND	Jeanne	2
17	DUBOIS	Robert	1
22	DUBALAI	Aline	1
26	DUGENOU	Jean	2
11	DURAND	Aline	2
38	DURACUIRE	Robert	2
9	DURALUMIN	Roberte	1
13	DURDUR	Jean	2
20	DURALEX	Jean	1

fragment 1

numero	nom	prenom	numeq
12	DUPONT	Jean	1
17	DUBOIS	Robert	1
22	DUBALAI	Aline	1
9	DURALUMIN	Roberte	1

fragment 2

numero	nom	prenom	numeq
34	DUPOND	Jeanne	2
26	DUGENOU	Jean	2
11	DURAND	Aline	2
38	DURACUIRE	Robert	2

La recomposition s'effectuera par une simple opération d'union.

Fragmentation verticale

Dans la fragmentation verticale ou fragmentation par attributs, les fragments sont construits à partir de quelques attributs d'une "classe".

Exemple : la table CUISINIER peut être répartie en 2 fragments, l'un avec les attributs *numero*, *nom*, *prenom*, l'autre avec les attributs *numero*, *numeq*. On constatera que l'attribut *numero* est commun ; c'est une nécessité car *numero* est la clé de la table et la recomposition n'est possible que grâce à cet attribut (par jointure).

The diagram illustrates vertical fragmentation of the CUISINIER table. On the left, the original table is shown with columns: numero, nom, prenom, and numeq. The data consists of 13 tuples. Two blue arrows point from this table to two separate fragments on the right.

table cuisinier

numero	nom	prenom	numeq
12	DUPONT	Jean	1
34	DUPOND	Jeanne	2
17	DUBOIS	Robert	1
22	DUBALAI	Aline	1
26	DUGENOU	Jean	2
11	DURAND	Aline	2
38	DURACUIRE	Robert	2
9	DURALUMIN	Roberte	1
13	DURDUR	Jean	2
20	DURALEX	Jean	1

fragment 1

numero	nom	prenom
12	DUPONT	Jean
34	DUPOND	Jeanne
17	DUBOIS	Robert
22	DUBALAI	Aline
26	DUGENOU	Jean
11	DURAND	Aline
38	DURACUIRE	Robert
9	DURALUMIN	Roberte
13	DURDUR	Jean
20	DURALEX	Jean

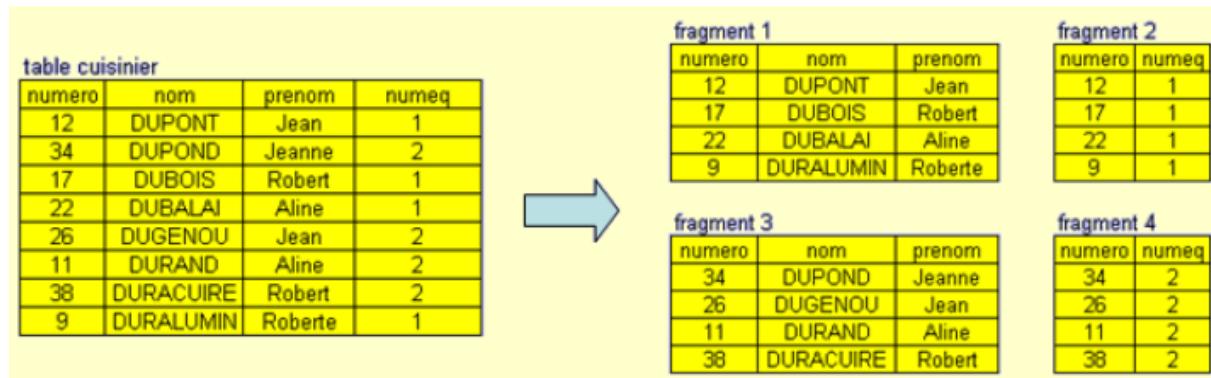
fragment 2

numero	numeq
12	1
34	2
17	1
22	1
26	2
11	2
38	2
9	1
13	2
20	1

Fragmentation par valeurs

Cette fragmentation combine la fragmentation horizontale et verticale.

exemple :



3. Problématique

Une requête sur une base de données répartie nécessite généralement des transferts de données entre sites. Ces transferts sont responsables au premier chef d'un allongement du temps de réponse car les traitements internes sont bien plus rapides que le transport des données.

B. Les Bases de Données Multimédia

1 : Introduction

Multimédia et bases de données

Qu'est-ce qu'une donnée multimédia ? C'est une donnée qui appartient à plusieurs formats d'information :

- ✓ L'image fixe
- ✓ L'image animée
- ✓ Le son
- ✓ La vidéo
- ✓ Le texte alphanumérique.

L'ensemble conjugué de ces différentes données, pour une diffusion vers des utilisateurs, s'appelle...le multimédia.

La problématique des bases de données multimédias est aisément compréhensible si l'on prend l'exemple illustratif suivant. Un hold up a lieu dans une banque et la police recherche activement les voleurs. Pour cela elle dispose

- ✓ d'enregistrement vidéo provenant des caméras de surveillance
- ✓ de conversations téléphoniques provenant de divers suspects
- ✓ de photographies prises par des témoins
- ✓ de photographies issues de fichiers de malfaiteurs

Tous ces documents sont volumineux et "manuellement" la police va mettre beaucoup de temps pour rechercher les corrélations possibles permettant d'identifier les voleurs. Il faut donc

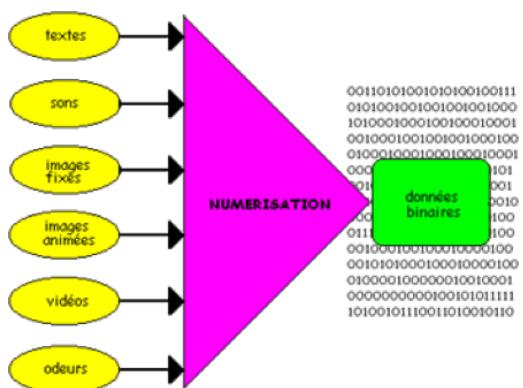
- ✓ pouvoir enregistrer ces documents dans un dispositif unique
- ✓ pouvoir effectuer des requêtes sur cet ensemble de documents pour rechercher par exemple des similarités.

Spécificité des données multimédias

Le multimédia est l'intégration au sein d'un même système d'information de données textuelles, d'images fixes ou animées, de sons, de vidéo. Son histoire est très récente, depuis 1990 environ, et va de pair avec la tendance prononcée à la numérisation de l'information, la production de petits systèmes informatiques à possibilités très performantes et au développement de l'interconnexion de réseaux, notamment dans ce dernier cas, Internet et sa composante "multimédia" : le World Wide Web.

Un premier intérêt du multimédia est l'intégration généralisée des données et des outils de manipulation de ces données. Avant le multimédia, des appareillages divers étaient nécessaires : un ordinateur pour les données informatiques, un magnétophone pour les cassettes de sons, un magnétoscope pour les vidéos, un téléphone pour le transport de la voix, du papier pour les images, ... L'utilisation simultanée de données de natures différentes nécessitait donc une juxtaposition inconfortable de ces outils. L'utilisation de l'ordinateur comme outil unique est un progrès sensible.

Par ailleurs, les données multimédias sont manipulées comme des données informatiques puisque codées sous forme de 0 et de 1 et peuvent donc, en particulier, être stockées de manière unique et transmises de manière unique ce qui constitue un second intérêt



Cette intégration se fait cependant avec quelques problèmes :

- ❖ le volume de données numérisées est généralement important, ce qui nécessite des techniques de compression/décompression.
Les traitements sont plus complexes et nécessitent des matériels et logiciels de plus en plus performants
- ❖ la transmission de données volumineuses nécessite des débits binaires importants sur les réseaux.

Néanmoins cette tendance semble irréversible et entraîne une reconfiguration du paysage autrefois audio-visuel en paysage totalement numérique.

2. Le multimédia dans l'objet-relationnel

SQL3 a défini des types de données "classiques" :

- ✓ CHAR(n) : chaîne de caractères de longueur n
- ✓ CHAR VARYING(n) : chaîne de caractères de longueur maximale n [Oracle nomme ce type VARCHAR2(n)]
- ✓ INTEGER : nombre entier [Oracle nomme ce type NUMBER]
- ✓ NUMERIC(p,q) ; nombre décimal de p chiffres avec q chiffres décimaux [Oracle nomme ce type NUMBER(p,q)]
- ✓ FLOAT: nombre flottant [pas d'équivalent chez Oracle]
- ✓ DATE : date
- ✓ TIME : heure
- ✓ TIMESTAMP : date et heure [pas d'équivalent chez Oracle]

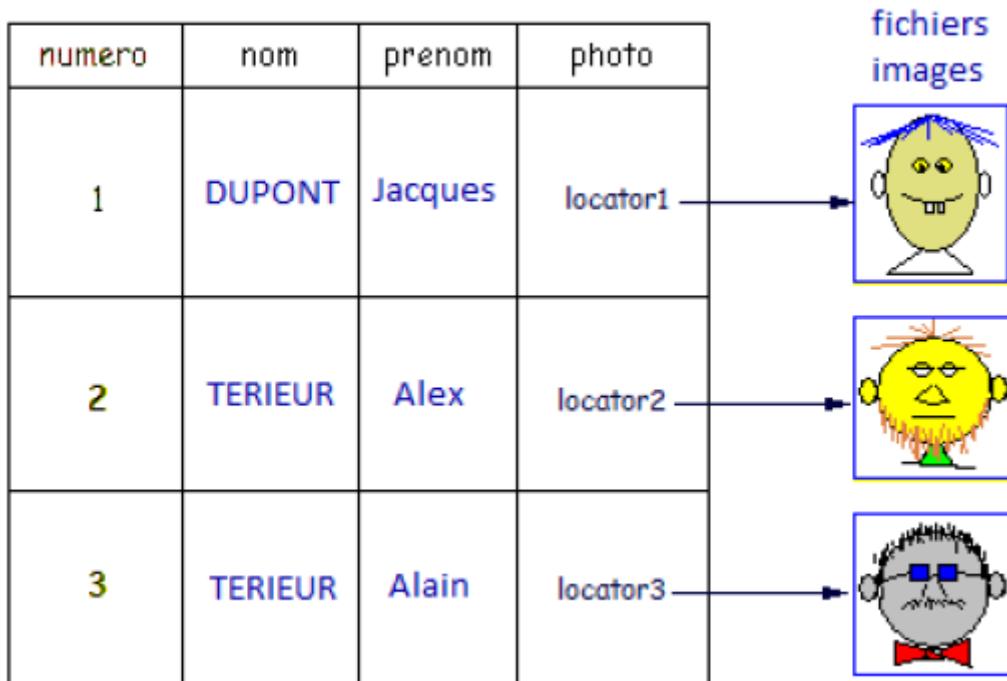
Il a aussi introduit des types de données se référant à des objets multimédias :

- ✓ BLOB : objet codé en binaire (Binary Large Object)
- ✓ CLOB : objet codé en caractères (Character Large Object)
- ✓ BFILE : fichier binaire externe (Binary File)

BLOB, CLOB (les "LOB") correspondent à des documents multimédias. BLOB est utilisé pour des fichiers binaires de type image, audio ou vidéo. CLOB est utilisé pour des données texte volumineuses. Dans tous les cas, le stockage comprend un "LOB-locator", qui est un pointeur vers les données multimédias, et les données multimédias elles-mêmes, "LOB-value". La table ne contient que le LOB-locator et la LOB-value est stockée en dehors de la table (mais dans la base de données cependant). La taille d'un LOB peut aller jusqu'à 128 To, mais tout dépend du SGBD utilisé (4 Go pour Oracle).

A contrario BFILE désigne des fichiers représentant des objets multimédias stockés à l'extérieur de la base de données. Dans la base de données BFILE correspond à un pointeur (BFILE-locator) permettant de trouver l'objet extérieur à la base de données (sous forme d'un fichier binaire).

Exemple 1 : on souhaite créer une table répertoriant des personnes avec leur nom, leur prénom et leur photographie. On ajoutera un numéro identificateur pour distinguer les personnes. On a donc une relation PERSONNE (numero, nom, prenom, photo) dont la vue conceptuelle est la suivante :



```
CREATE TABLE personne
(
    numero NUMBER PRIMARY KEY,
    nom VARCHAR2(20),
    prenom VARCHAR2(20),
    photo BLOB
);
```

On a ici choisi de stocker les photographies dans la base de données. Bien entendu, dans la table, on n'a pas les photos directement mais en général des LOB-locators.

C. Les Bases de Données Orientées Objet

1. Concepts de base

La séparation des traitements et des données a permis de fonder les bases d'une méthodologie de conception dont Merise est l'exemple et de développer la théorie des bases de données. Toutefois cette séparation pose un problème évident : il n'est pas toujours possible de séparer complètement les données et les traitements car ces derniers sont les manipulateurs des premières (et on ne peut "manipuler" que si les données concernées sont présentes et correctement définies). La conception des systèmes d'information suivant Merise suppose d'ailleurs une confrontation entre le modèle conceptuel des traitements et le modèle conceptuel des données pour validation du dernier par le premier.

Le modèle objet suppose à l'inverse que données et traitements soient réunis, mais sous une forme "granulaire" : l'objet.

Le mot objet n'est pas choisi au hasard ; en effet, dans toute modélisation, on cherche à représenter la réalité au plus près et notamment les différents "objets" réels. On prend donc le même mot, objet, pour désigner la modélisation d'un objet réel.

Prenons comme support explicatif l'exemple d'une bibliothèque. Pour modéliser la gestion d'une bibliothèque, on identifie les objets réels : livres, lecteurs, auteurs, œuvres, rayons, prêts, achats,... Certains de ces objets ont une réalité matérielle (livres, lecteurs, auteurs, rayons), d'autres ont une réalité abstraite (œuvres, prêts, achats). On fera, dans un premier temps, correspondre à ces objets "réels" des objets du modèle que l'on pourra d'ailleurs appeler du même nom. Ainsi l'objet "livre" peut être décrit par des attributs : numéro d'inventaire, titre, auteurs, date de publication, nombre de pages, éditeur,... et par des opérations : acheter, prêter, ranger.

Retenant cet exemple, il existe beaucoup d'autres livres dans une bibliothèque, qui possèdent les mêmes attributs et les mêmes méthodes ; on dit qu'ils appartiennent à une classe, en l'occurrence ici, la classe "livre" dont l'objet livre1 est une instanciation. La classe se définira par les attributs et les méthodes (communs à tous les objets de la classe) ; elle définit une structure d'objet. La classe LIVRE se définit donc comme suit :

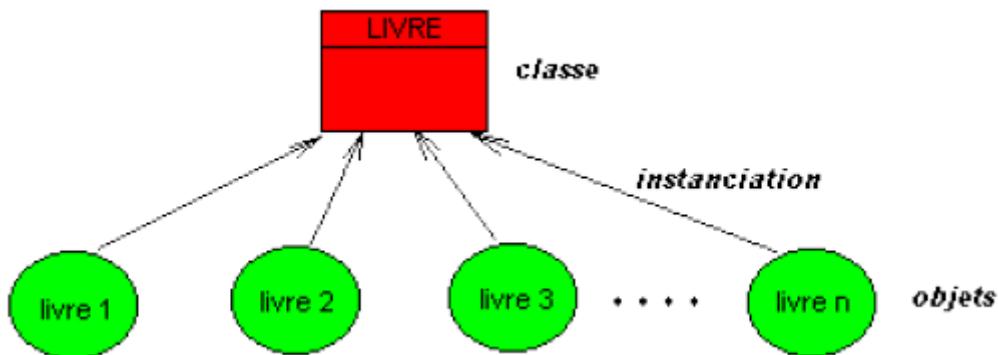
classe : LIVRE

attributs :

- numinv
- auteurs
- éditeur
- date_édition
- nbpages

méthodes :

acheter(date, fournisseur, facture)
prêter(date, lecteur)
ranger(rayon)



2. SGBD Orienté Objet

Les bases de données "Orientées Objet" sont effectivement conçues pour stocker et manipuler des objets. La première base de données de cette catégorie semble être Gemstone (1983) élaborée à partir du langage de programmation Smalltalk. Il existe actuellement une assez grande variété de bases de données "orientées objet" dans le domaine expérimental et dans le commerce.

Parmi les plus connues, on peut citer les prototypes expérimentaux :

- ✓ ORION (Microelectronics and Computer Technology Corporation)
- ✓ ONTOS (Ontos)
- ✓ OPENOODB (Texas Instruments)
- ✓ ENCORE/Observer (Brown University)
- ✓ ODE (AT&T Bell Labs, puis Lucent Technologies)

et les produits commerciaux :

- ✓ O2 (O2 Technology), puis ARDENT (ARDENT Software), puis n'est plus commercialisé
- ✓ Gemstone/OPAL (Servio Logic)
- ✓ Ontos (Ontologic)
- ✓ Statice (Symbolics)
- ✓ Gbase (Object Databases)
- ✓ Vision (Innovative Systems)
- ✓ Objectstore (Object Design)
- ✓ Objectivity/DB (Objectivity)
- ✓ Versant (Versant Object Technology)
- ✓ Orion (Itasca Systems)
- ✓ Illustra (Illustra Inc.)

- ✓ SQLX (UniSQL)

Ces bases sont nées de la conjonction de deux domaines : les bases de données (et plus spécialement relationnelles) et les langages de programmation orientés objet (Simula, Smalltalk, Eiffel, C++, ...).

Signalons les travaux de l'ODMG (Object Database Management Group), fondé en 1991 par 5 éditeurs (O2 Technology, Objectivity, Object Design, Ontos, Versant) qui s'était donné comme tâche la spécification des bases de données orientées objet (il s'est dissout en 2001).

3. Généralités sur l'objet-relationnel

Introduction

Il y a deux limites principales au modèle relationnel :

- ❖ l'approche orientée objet n'est pas prévue : les concepts d'objet, d'héritage, d'encapsulation, lui sont étrangers.
- ❖ le modèle relationnel ne manipule que des données classiques. Il n'existe pas de structures de données. Par ailleurs, les types de données sont prédéfinis. Aucune possibilité de créer des types nouveaux.

A partir de ce constat, on peut envisager deux types de solutions :

1ère possibilité : abandonner le modèle relationnel et construire un nouveau modèle "objet" de bases de données. De tels modèles existent (presque) mais leur commercialisation sous forme de SGBD est lente et ils ne sont vraiment utilisés que dans des créneaux très spécialisés. Il faut dire que les éditeurs de SGBD relationnels ne sont pas enclins à passer brutalement à des modèles objets pour des raisons économiques très compréhensibles.

2ème possibilité : étendre le modèle relationnel vers un modèle objet-relationnel (il serait plus exact de dire "relationnel-objet"). Il s'agit de rendre le modèle relationnel compatible avec les "règles" objet. De fait, il y a trois extensions essentielles par rapport au modèle relationnel standard :

- ✓ l'incorporation de structures de données complexes : les types abstraits de données (TAD en français, ADT en anglais).
- ✓ l'introduction des objets, notamment via leur identifiant (appelé OID : object identifier)
- ✓ la modélisation des associations à partir des deux premières extensions : les structures de données complexes (tables imbriquées) et l'identité des objets (usages de pointeurs)

Les SGBD commerciaux, basés sur le modèle Objet-relationnel sont en général issus du modèle Relationnel. Citons, parmi les plus connus :

Cours d'Ateliers de Génie Logiciel

- ✓ Oracle (versions > 8)
- ✓ Sybase
- ✓ DB2
- ✓ Informix
- ✓ PostgreSQL (open source)

Les types abstraits de données

Dans le modèle relationnel, les types de données sont prédéfinis (nombre, caractères, date, ...). Dans le modèle objet-relationnel, les types abstraits de données (TAD) sont des types d'attributs définis par l'utilisateur. On retrouve évidemment les types de données prédéfinis du modèle relationnel. Les TAD peuvent aussi être définis à partir d'autres TAD.

Dans ce qui suit, on utilisera une syntaxe de type SQL3 pour la description, la manipulation et l'interrogation de données.

Du point de vue conceptuel, on peut considérer que dans le modèle objet-relationnel, les TAD correspondent aux classes de la modélisation objet.

exemple :

- création du type T_adresse :

```
CREATE TYPE T_adresse ( num NUMBER, voie CHAR(30), ville(30), cp (CHAR(10));
```

- création du type T_personne :

```
CREATE TYPE T_personne (nom CHAR(30), prenom CHAR(30), adr T_adresse);
```

- création d'une table ANNUAIRE :

```
CREATE TABLE ANNUAIRE OF T_personne
```

nom	prenom	adr			
		num	voie	ville	cp

Sur cet exemple, on voit que l'on a incorporé dans une table un objet complexe adr (ce qui est contraire à la 1ère forme normale)

Les objets

Dans le modèle objet-relationnel, on peut définir des objets (en fait des classes avec les TAD). Un objet de la base est représenté par son identifiant OID. L'utilisateur n'a pas à s'en soucier. En revanche c'est le SGBD qui l'utilise, de manière transparente, pour rechercher et trouver

Cours d'Ateliers de Génie Logiciel

les objets via leur OID. L'utilisation des OID permet de chaîner les objets entre eux. En particulier le modèle objet-relationnel introduit les pointeurs comme lien entre deux objets.

exemple :

```
CREATE TYPE T_adresse (num NUMBER, voie CHAR(30), ville CHAR(30), cp  
CHAR(10);  
CREATE TYPE T_personne(nom CHAR(30), pnom CHAR(30), adr REF T_adresse);  
CREATE TABLE ADRESSE OF T_adresse;  
CREATE TABLE PERSONNE OF T_personne;
```

nom	prenom	adresse
DUPONT	Jacques	
TERIEUR	Alex	
TERIEUR	Alain	

TABLE PERSONNE

numero	voie	ville	code postal
3	rue des Jonquilles	Bordeaux	33000
54	avenue des Tomates	Montpellier	34000
78	Cours de la Libération	Nice	06000

TABLE ADRESSE

Pour montrer la différence avec le modèle relationnel examinons quelques exemples de requêtes.

exemple : une cuisine pour collectivités est organisée en équipes de cuisiniers ; chaque équipe est dirigée par un chef d'équipe.

Dans le modèle relationnel, on peut représenter la situation par une table CUISINIER (numero, nom, prenom, chef)

numero	nom	prenom	chef
12	DUPONT	Jean	17
34	DUPOND	Jeanne	11
17	DUBOIS	Alain	NULL
21	DUBALAI	Aline	11
26	DUGENOU	Pierre	17
11	DURAND	Pierrette	NULL
38	DURACUIRE	Robert	17
9	DURALUMIN	Roberte	11

Considérons les deux requêtes suivantes :

R1 : Quel est le chef du cuisinier 26 ?

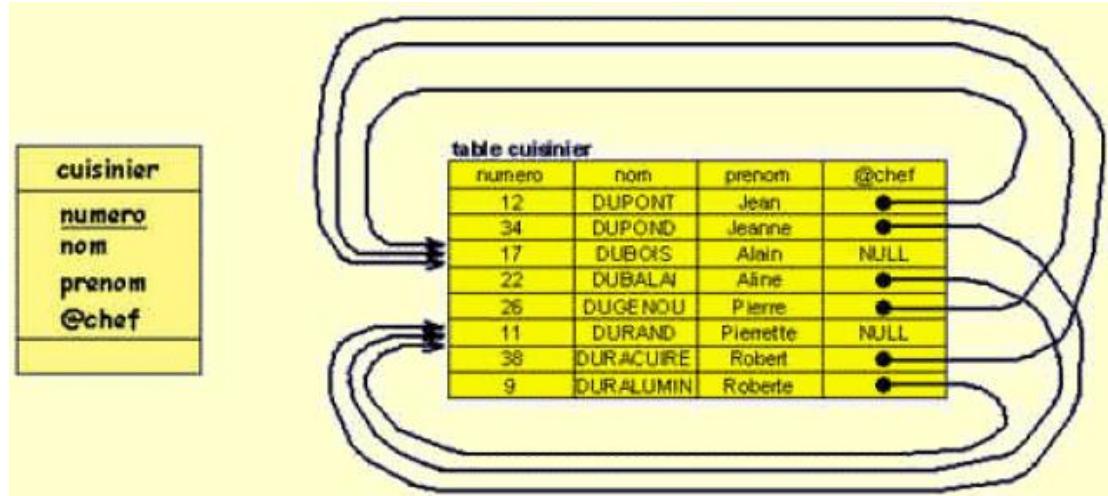
R2 : Quels sont les cuisiniers de l'équipe dirigée par Alain DUBOIS ?

La solution, avec SQL2, peut être :

R1 : SELECT numero, nom, prenom FROM CUISINIER WHERE numero = (SELECT chef FROM CUISINIER WHERE numero=26);

R2 : SELECT numero, nom, prenom FROM CUISINIER WHERE chef = (SELECT numero FROM CUISINIER WHERE nom='DUBOIS' AND prenom = 'Alain');

Dans le modèle objet-relationnel, une occurrence de CUISINIER peut être considérée comme un objet (qui sera repéré par son OID).



On pourra formuler les requêtes de la manière suivantes (dans une syntaxe de type SQL3) :

R1 : SELECT c.chef.numero, c.chef.nom, c.chef.prenom
FROM CUISINIER c
WHERE c.numero=26;

R2 : SELECT c.numero, c.nom, c.prenom
FROM CUISINIER c
WHERE c.chef.nom = "DUBOIS" AND
c.chef.prenom = 'Alain';

On notera l'écriture qui fait usage d'un alias. Un objet pointé s'écrit ainsi "alias.pointeur.objet_pointé".

4. Oracle et l'objet-relationnel

On considère parfois Oracle comme le premier SGBD Relationnel (1979). Toujours est-il qu'il s'est maintenu jusqu'à aujourd'hui à la première place des systèmes commercialisés. Il a su évoluer du relationnel vers l'Objet-Relationnel depuis la version 8 complétée par la version 8i qui intègre des fonctionnalités Web et multimédia. La version actuelle est 18c. On peut utiliser gratuitement une version d'Oracle appelée Oracle XE (Express Edition) ; nous l'utiliserons dans la partie des travaux pratiques.

Il possède l'intérêt majeur d'être implantable sur les deux principaux systèmes d'exploitation Unix (et Linux) et Windows. Il est évidemment concurrencé aujourd'hui par des SGBD libres et OpenSource.

Le type abstrait de données OBJET

Oracle 8 permet la création de TAD spécifiques que SQL3 ne prévoit pas :

- ✓ le type OBJET
- ✓ les types du genre " COLLECTION " : VARRAY et NESTED TABLE

Le type OBJET

Il est défini de la façon suivante :

```
CREATE TYPE <nom du type> AS OBJECT (
    attribut 1 DATATYPE,
    attribut 2 DATATYPE,
    -----
    ,
    attribut n DATATYPE,
    [définition de méthode]
);
```

Les méthodes sont des fonctions ou des procédures qui sont stockées dans la base (PL/SQL) ou à l'extérieur de la base (langage C).

exemples :

```
CREATE TYPE T_personne AS OBJECT (
    nom VARCHAR2(30),
    telephone VARCHAR2(20));
```

```
CREATE TYPE T_article AS OBJECT (code VARCHAR2(30),
    qte NUMBER,
    pu NUMBER(10,2)
);
```

On peut ensuite utiliser le mot clé OF pour utiliser ce type :

```
CREATE TABLE Personne OF T_personne;
```

```
CREATE TABLE Article OF T_article;
```

Il faut faire une distinction entre les tables typées et les tables non typées. Une table typée est déclarée à l'aide d'un TAD comme dans l'exemple précédent (CREATE TABLE <nom_table> OF <nom_type>). Un table non typée est définie directement (CREATE TABLE <nom_table> (.....attributs.....)). Dans les tables typées, les lignes de la table sont des objets possédant un OID. Dans les tables non typées, les lignes de la table ne sont pas des objets (et ne possèdent pas d'OID).

Cours d'Ateliers de Génie Logiciel

CREATE TABLE machin (numP NUMBER, nomP VARCHAR2(40)); donnera une table comportant des enregistrements qui ne sont pas des objets. C'est le cas du modèle "simplement" relationnel.

```
CREATE TYPE T_truc AS OBJECT (numP NUMBER, nomP VARCHAR2(40));
```

CREATE TABLE machin OF T_truc; donnera une table dont les enregistrements sont des objects.

Les types abstraits de données VARRAY et NESTED TABLE

Il permet la création d'attributs et correspond aux deux possibilités VARRAY et NESTED TABLE.

VARRAY

Tableau à 1 dimension d'éléments du même type ; le tableau a une taille variable.

```
CREATE TYPE <nom1 du type> AS VARRAY(nmax) OF <nom2 de type>
```

exemple : répertoire de clients pouvant avoir chacun 3 adresses au maximum

```
CREATE TYPE T_client_adr AS OBJECT (  
    num NUMBER,  
    voie VARCHAR2(30),  
    ville VARCHAR2(30),  
    cp NUMBER  
) ;
```

```
CREATE TYPE T_adresses AS VARRAY(3) OF T_client_adr;
```

```
CREATE TYPE T_client AS OBJECT (  
    nomclient VARCHAR2(30),  
    adresses T_adresses  
) ;
```

```
CREATE TABLE Client OF T_client;
```

NESTED TABLE

Ce type correspond à un ensemble non ordonné et non limité d'éléments de même type. Ici le type désigne une table.

```
CREATE TYPE <nom1 de type> AS TABLE OF <nom2 de type>
```

exemple : description d'un stock d'articles

```
CREATE TYPE T_article AS OBJECT (  
    nomart VARCHAR2(30),
```

Cours d'Ateliers de Génie Logiciel

```
        qte NUMBER,  
        pu NUMBER(10,2)  
);  
  
CREATE TYPE T_articles AS TABLE OF T_article ;  
  
CREATE TYPE T_stock AS OBJECT  
    categorie VARCHAR2(30),  
    articles T_articles  
);  
  
CREATE TABLE Stock OF T_stock NESTED TABLE articles STORE AS LISTE;  
  
NESTED TABLE articles indique au SGBD que articles est une table imbriquée et STORE AS LISTE nomme la structure interne pour stocker cette table imbriquée (on l'appelle LISTE ou n'importe quoi d'autre).
```

exemple : description d'une commande

```
CREATE TYPE T_article AS OBJECT (  
    numart INTEGER;  
    qtecom INTEGER  
);  
  
CREATE type T_com_article AS TABLE OF T_article;  
  
CREATE TYPE T_ligne_com AS OBJECT (  
    numcom INTEGER,  
    datecom DATE,  
    numcli INTEGER,  
    articles T_com_article  
);
```

```
CREATE TABLE Commande OF T_ligne_com NESTED TABLE articles STORE AS ENCOM;
```

Manipulation de données

Les instructions de manipulation restent classiques : INSERT (pour ajouter un enregistrement), UPDATE (pour modifier un enregistrement), DELETE (pour supprimer un enregistrement).

Tables relationnelles

Considérons l'exemple suivant avec les tables EMPLOYEE, SERVICE définies par

```
CREATE T_service AS OBJECT (numS NUMBER, nomS VARCHAR2(40));  
  
CREATE T_employe AS OBJECT (numE NUMBER, nomE VARCHAR2(40), numS NUMBER);
```

Cours d'Ateliers de Génie Logiciel

CREATE TABLE employe OF T_employe;

CREATE TABLE service OF T_service;

EMPLOYEE

numE	nomE	numS

SERVICE

numS	nomS

Ajoutons les clés primaires et étrangères :

ALTER TABLE service ADD CONSTRAINT PK_SERVICE PRIMARY KEY(numS);

ALTER TABLE employe ADD CONSTRAINT PK_EMPLOYE PRIMARY KEY(numE); ALTER TABLE employe ADD CONSTRAINT FK_EMPLOYE PRIMARY KEY (numS);

Noter que l'on aurait pu écrire directement

CREATE TABLE service OF T_service (CONSTRAINT PK_SERVICE PRIMARY KEY (numS));

CREATE TABLE employe OF T_employe (CONSTRAINT PK_EMPLOYE PRIMARY KEY (numE), CONSTRAINT FK_EMPLOYE FOREIGN KEY (numS) REFERENCES service (numS));

Insérons un enregistrement dans service :

INSERT INTO service VALUES (1, 'achats');

En fait, on devrait écrire la forme plus correcte (mais il y a de la tolérance) INSERT INTO service VALUES (T_service (1, 'achats')); pour rappeler la structure des objets de la table.

Quand on a plusieurs enregistrements à ajouter il est commode de travailler en mode PL/SQL (voir plus loin) :

```
BEGIN
    INSERT INTO service VALUES (2, 'comptabilité');
    INSERT INTO service VALUES (3, 'production');
    INSERT INTO service VALUES (4, 'ventes');
END;
```

De même :

```
BEGIN
    INSERT INTO employe VALUES (1, 'DUPONT',2);
    INSERT INTO employe VALUES (2, 'DURAND',4);
```

Cours d'Ateliers de Génie Logiciel

```
INSERT INTO employe VALUES (3, 'DUPUIS',1);
INSERT INTO employe VALUES (4, 'DUVAL',3);
INSERT INTO employe VALUES (5, 'DUCHEMIN',3);
INSERT INTO employe VALUES (6, 'DUMOULIN',2);
INSERT INTO employe VALUES (7, 'DUMONT',1);

END;
```

Changeons de service l'employé de numéro numE = 6 qui passe du service 2 au service 4 :

```
UPDATE employe SET numS = 4 WHERE numE = 6;
```

Supprimons maintenant l'employé 'DUMONT' :

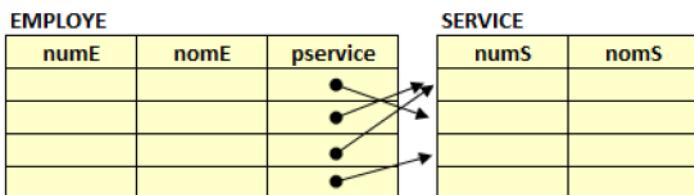
```
DELETE FROM employe WHERE nomE = 'DUMONT';
```

Tables avec pointeur

Tout ce qui précède est bien classique. Supposons maintenant que l'on remplace la clé étrangère numS par un pointeur sur un enregistrement de la table service. Il faut modifier la structure de la table employe que l'on appellera maintenant employe2.

```
CREATE TYPE T_employe2 AS OBJECT (numE NUMBER, nomE VARCHAR2(40), pservice REF T_service);
```

```
CREATE TABLE employe2 OF T_employe2;
```



Insérons un enregistrement dans la table employe2

```
INSERT INTO employe2 VALUES (1, 'DUPONT', (SELECT REF(s) FROM service s WHERE s.numS = 1));
```

Modifions un enregistrement de la table employe2 : changement d'affection de l'employé de numéro numE = 6 qui passe du service 2 au service 4 :

```
UPDATE employe2 e SET e.pservice = (SELECT REF(s) FROM service s WHERE s.numS = 4)
WHERE e.numE=6
```

Enfin supprimons les employés qui travaillent dans le service numS = 1

```
DELETE FROM employe2 e WHERE e.pservice = (SELECT REF(s) FROM service s WHERE s.numS = 1);
```

Tables imbriquées

On veut maintenant réunir les deux tables service et employe en une seule table dont la structure est la suivante :

ENTREPRISE		employes	
numS	nomS	numE	nomE

```
CREATE TYPE T_employe3 AS OBJECT (numE NUMBER, nomE VARCHAR2(40));
```

```
CREATE TYPE T_employes AS TABLE OF T_employe3;
```

```
CREATE TABLE entreprise (numS NUMBER, nomS VARCHAR2(40), employes T_employes
NESTED TABLE employes STORE AS PERSONNEL);
```

Insérons le service numS = 1, nomS = 'achats' avec les employés (1, 'DUPONT'), (3, 'DUPUIS'), (4, 'DUVAL') :

```
INSERT INTO entreprise VALUES (1, 'achats', T_employes (
```

```
    T_employe3(1, 'DUPONT'),
    T_employe3(3, 'DUPUIS'),
    T_employe3(4, 'DUVAL')
)
```

```
);
```

Effectuons maintenant une mise à jour en changeant en 10 le numéro numE de DUMOULIN qui est dans le service numS = 3. Ici les choses se compliquent et on doit utiliser le mot clé THE qui désigne une table imbriquée.

```
UPDATE THE (SELECT f.employes FROM entreprise f WHERE f.numS=3) t SET t.numE=10
WHERE t.nomE='DUMOULIN';
```

Pour faire le changement d'affectation vu précédemment, il faut en fait créer un enregistrement et en détruire un autre. Par exemple pour changer d'affectation DUPUIS du service numS = 1 au service numS = 2, il faudra effectuer successivement

```
INSERT INTO THE (SELECT f.employes FROM entreprise f WHERE f.numS = 2) VALUES (3,
'DUPUIS');
```

Cours d'Ateliers de Génie Logiciel

```
DELETE FROM THE (SELECT f.employes FROM entreprise f WHERE f.numS = 1) t WHERE t.nomE = 'DUPUIS';
```

Requêtes

On reprend ici les trois exemples précédents : tables relationnelles, tables avec pointeurs, tables imbriquées.

Tables relationnelles

Le SELECT ne pose pas de problème avec les tables relationnelles. SQL3 s'applique simplement avec les alias. Pour trouver le service où travaille 'DUMOULIN', on écrira :

```
SELECT s.numS, s.nomS FROM service s WHERE s.numS = (SELECT e.numS FROM employe e WHERE e.nomE = 'DUMOULIN');
```

ou bien

```
SELECT s.numS, s.nomS FROM service s, employe e WHERE e.numS = s.numS AND e.nomE = 'DUMOULIN';
```

Tables avec pointeur

Effectuons la même recherche que précédemment

```
SELECT s.numS, s.nomS FROM service s WHERE s.numS = (SELECT e.pservice.numS FROM employe2 e WHERE e.nomE = 'DUMOULIN');
```

ou bien

```
SELECT s.numS, s.nomS FROM service s, employe2 e WHERE e.pservice.numS = s.numS AND e.nomE = 'DUMOULIN';
```

Tables imbriquées

Toujours la même requête. Employons THE pour désigner la table imbriquée

```
SELECT f.numS, f.nomS FROM entreprise f, THE (SELECT g.employes FROM entreprise g WHERE g.numS = f.numS) t WHERE t.nomE='DUMOULIN';
```

On peut aussi utiliser TABLE(f.employes) pour désigner la table imbriquée de l'enregistrement f de entreprise :

```
SELECT f.numS, f.nomS FROM entreprise f, TABLE(f.employes) e WHERE e.nomE='DUMOULIN';
```

Chapitre III : Travaux Pratiques avec les AGL

III.1. Travaux Pratiques avec Oracle

Objectif

Imaginons une configuration qui nous conduirait à une table composite comme la suivante :

ECOLE1

nomC	numE	nomE	numP	nomP	nbh
CM1	10	FRICOTIN	P1	DUPONT	12
	14	TARZAN	P5	DUBALAI	18
	18	FANTASIO	P6	DUVAL	25
	23	BILL			
CM2	11	FILOCHARD	P1	DUPONT	25
	15	TINTIN	P3	DUGENOU	28
	19	MICKEY			
CE1	12	CROQUIGNOL	P2	DUBOIS	18
	16	MILOU	P4	DURACUIRE	15
	20	DONALD	P5	DUBALAI	32
CE2	13	RIBOULDINGUE	P3	DUGENOU	30
	17	SPIROU	P4	DURACUIRE	30
	22	BOULE	P6	DUVAL	28

Définition de types abstraits de données

Nous voyons que cette table implique des tables imbriquées. Pour définir une telle structure, on est conduit à définir des TAD comme le type T_eleve composé de numE et de nomE et le type T_professeur composé de numP, nomP et nbh.

Commençons par définir ces deux types avec Oracle XE en utilisant les commandes SQL :

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The menu bar has "Fichier", "Accès", "SQL", "Paramètres", "Aide", and "Aide en ligne". The main window has a toolbar with "Enregistrer" and "Exécuter". The SQL command window contains the following code:

```
CREATE TYPE eleve_T AS OBJECT (
    numE Numberic,
    nomE varchar2(30));

```

The status bar at the bottom says "Saisissez l'instruction SQL ou la commande PL/SQL, puis cliquez sur Exécuter pour voir les résultats."

Faisons de même avec le type T_professeur :

```
CREATE TYPE T_professeur AS OBJECT (
    numP Numeric,
    nomP varchar2(30),
```

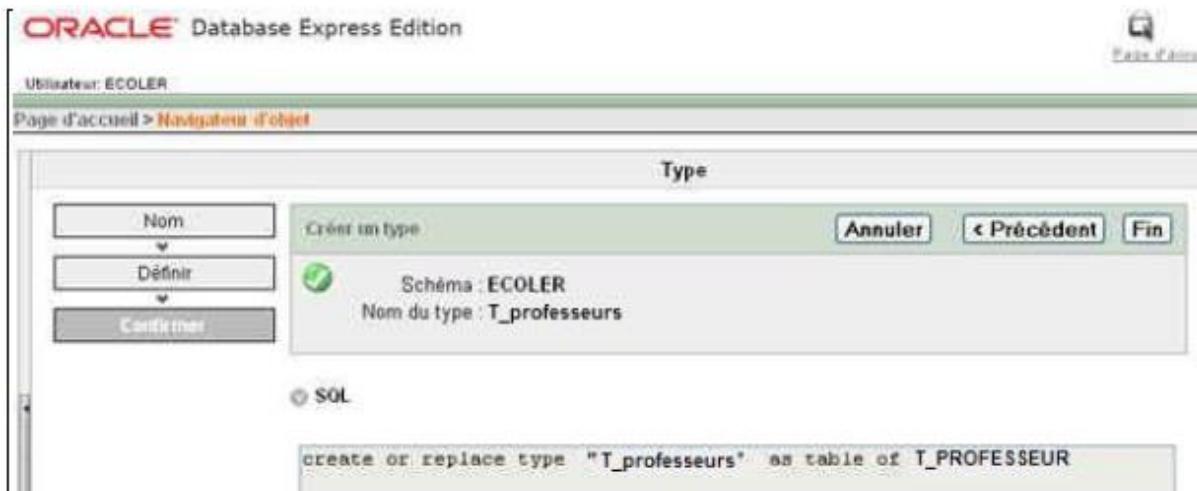
Cours d'Ateliers de Génie Logiciel

```
nbh Numeric  
);
```

Créons maintenant les tables imbriquées.

La vue SQL permet de voir comment la création est formulée.

On agit de même pour créer le type T_professeurs :



Création de tables comportant des tables imbriquées

Il reste simplement à créer la table ECOLE1 considérée comme une table d'objets de type T_ecole1 :

```
CREATE TYPE T_ecole1 AS OBJECT(  
    nomC varchar2(30),  
    eleves T_eleves,  
    professeurs T_professeurs  
);
```

Nous utilisons une commande SQL pour créer la table :

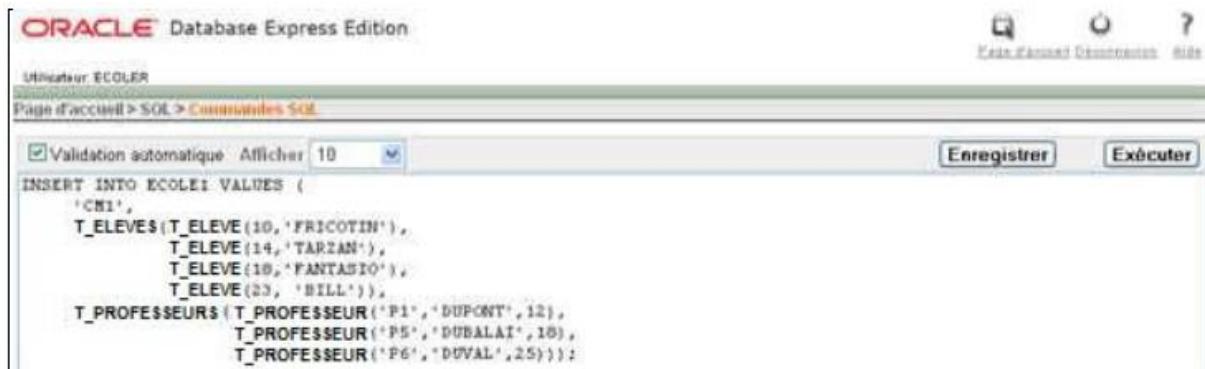
```
CREATE TABLE ECOLE1 OF T_ecole1  
(PRIMARY KEY(nomC) )  
    NESTED TABLE eleves STORE AS ELEV,  
    NESTED TABLE professeurs STORE AS PROF;
```

Insertion de données avec des nested tables

Il faut maintenant remplir la table. On utilisera une commande SQL pour montrer la syntaxe employée dans le cas de tables imbriquées :

Cours d'Ateliers de Génie Logiciel

Rentrerons d'abord ce qui correspond à nomC = 'CM1' :



```
ORACLE Database Express Edition
Utilisateur: ECOLER
Page d'accueil > SQL > Commandes SQL
Validation automatique : Afficher 10 Enregistrer Exécuter
INSERT INTO ECOLE1 VALUES (
    'CM1',
    T_ELEVES(T_ELEVE(10,'FRICOTIN'),
              T_ELEVE(14,'TARZAN'),
              T_ELEVE(10,'FANTASIO'),
              T_ELEVE(23,'BILL')),
    T_PROFESSEURS(T_PROFESSEUR('P1','DUPONT',12),
                  T_PROFESSEUR('P5','DUBALAI',18),
                  T_PROFESSEUR('P6','DUVAL',25)));
```

et faisons de même pour les autres lignes :

```
INSERT INTO ECOLE1 VALUES (
    'CM2',
    T_ELEVES(T_ELEVE(11,'FILOCHARD'),
              T_ELEVE(15,'TINTIN'),
              T_ELEVE(19,'MICKEY')),
    T_PROFESSEURS (T_PROFESSEUR ('P1','DUPONT',25),
                  T_PROFESSEUR('P3','DUGENOU',28))
);

INSERT INTO ECOLE1 VALUES (
    'CE1',
    T_ELEVES(T_ELEVE(12,'CROQUIGNOL'),
              T_ELEVE(16,'MILOU'),
              T_ELEVE(20,'DONALD')),
    T_PROFESSEURS(T_PROFESSEUR('P2','DUBOIS',18),
                  T_PROFESSEUR('P4','DURACUIRE',15),
                  T_PROFESSEUR ('P5','DUBALAI',32))
);

INSERT INTO ECOLE1 VALUES (
    'CE2',
    T_ELEVES(T_ELEVE(13,'RIBOULDINGUE'),
              T_ELEVE(17,'SPIROU'),
              T_ELEVE(22,'BOULE')),
    T_PROFESSEURS (T_PROFESSEUR('P3','DUGENOU',30),
                  T_PROFESSEUR('P4','DURACUIRE',30),
                  T_PROFESSEUR('P6','DUVAL',28))
);
```

Interrogation de données avec des nested tables

Rq1 : Donner la liste des élèves de la classe 'CM1'

Rq2 : Donner la liste des élèves qui ont 'DUBALAI' comme professeur

Rq3 : Donner le nombre d'heures effectuées par le professeur 'DUBALAI'

La différence avec le modèle relationnel vient notamment du fait que l'on manipule des objets, ce qui influe sur le langage d'interrogation.

La requête 1 peut être formulée de la manière suivante :

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The menu bar includes "Page d'accueil", "Déconnexion", and "Aide". The main area shows a SQL command window with the following content:

```
Validation automatique Afficher 10 Enregistrer Exécuter
SELECT e.numE, e.nomE FROM THE (SELECT ee.eleves FROM ECOLE1 ee WHERE ee.nomC = 'CM1')e;
```

Below the command window, there are tabs for "Résultats", "Expliquer", "Décrire", "SQL enregistré", and "Historique". The "Résultats" tab is selected. The results table has columns "NUME" and "NOME". The data is:

NUME	NOME
10	FRICOTIN
14	TARZAN
18	FANTASIO
23	BILL

4 lignes renvoyées en 0,08 secondes Export.CSV

Le mot THE désigne la table imbriquée (eleves) dans la table Ecole1.

La deuxième requête est un peu plus compliquée car on doit rechercher des informations dans une table imbriquée avec un critère s'appliquant à une autre table imbriquée. Voici une solution possible :

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The menu bar includes "Page d'accueil", "Déconnexion", and "Aide". The main area shows a SQL command window with the following content:

```
Validation automatique Afficher 20 Enregistrer Exécuter
SELECT e.numE, e.nomE FROM Ecole1 q, TABLE(q.eleves)e, TABLE(q.professeurs) p
WHERE p.nomP='DUBALAI' ORDER BY q.nomC, e.numE;
```

Below the command window, there are tabs for "Résultats", "Expliquer", "Décrire", "SQL enregistré", and "Historique". The "Résultats" tab is selected. The results table has columns "NUME" and "NOME". The data is:

NUME	NOME
10	FRICOTIN

Cours d'Ateliers de Génie Logiciel

où la fonction TABLE() permet de traiter dans une requête comme une table une nested table.

La troisième requête pose moins de difficultés :

The screenshot shows the Oracle Database Express Edition interface. The title bar reads "ORACLE Database Express Edition". The menu bar includes "Page d'accueil", "Déconnecter", and "Aide". The top navigation bar shows "Utilisateur: ECOLER" and "Page d'accueil > SQL > Commandes SQL". The main area contains a SQL query window with the following content:

```
Validation automatique Afficher 20 Enregistrer Exécuter
SELECT sum(p.nbh) FROM Ecole1 q, TABLE(q.professeurs) p WHERE p.nomP='DUBALAI';
```

Below the query window, there are tabs for "Résultats", "Expliquer", "Décrire", "SQL enregistré", and "Historique". The "Résultats" tab is selected. A result table is displayed with one row:

SUM(P.NBH)
50

Below the table, it says "1 lignes renvoyées en 0,00 secondes" and "Export CSV".

Utilisation de VARRAY

Au lieu d'utiliser des NESTED TABLE, on peut utiliser des VARRAY. On supposera qu'une classe ne peut contenir que 25 élèves et que 10 professeurs au maximum interviennent par classe. Créons alors la table Ecole2, identique à Ecole1 sauf que les tables imbriquées sont remplacées par des Varray.

```
CREATE TYPE T_Velevs AS VARRAY(25) OF T_ELEVE;
CREATE TYPE T_Vprofesseurs AS VARRAY(10) OF T_PROFESSEUR;
CREATE TYPE T_Ecole2 AS OBJECT (
    nomC varchar2(30),
    Velevs T_Velevs,
    Vprofesseurs T_Vprofesseurs
);
CREATE TABLE Ecole2 OF T_Ecole2;
```

Pour remplir la table, on procède de manière analogue aux tables imbriquées :

```
INSERT INTO Ecole2 VALUES (
    'CM1',
    T_Velevs (T_ELEVE(10,'FRICOTIN'),
              T_ELEVE(14,'TARZAN'),
              T_ELEVE(18,'FANTASIO'),
              T_ELEVE(23,'BILL')),
    T_Vprofesseurs(T_PROFESSEUR('P1','DUPONT',12),
                  T_PROFESSEUR('P5','DUBALAI',18),
```

Cours d'Ateliers de Génie Logiciel

```
T_PROFESSEUR('P6','DUVAL',25))  
);  
  
INSERT INTO Ecole2 VALUES (  
    'CM2',  
    T_Velevés(T_ELEVE(11,'FILOCHARD'),  
              T_ELEVE(15,'TINTIN'),  
              T_ELEVE(19,'MICKEY')),  
    T_Vprofesseurs(T_PROFESSEUR('P1','DUPONT',25),  
                    T_PROFESSEUR('P3','DUGENOU',28))  
);  
  
INSERT INTO Ecole2 VALUES (  
    'CE1',  
    T_Velevés(T_ELEVE(12,'CROQUIGNOL'),  
              T_ELEVE(16,'MILOU'),  
              T_ELEVE(20,'DONALD')),  
    T_Vprofesseurs(T_PROFESSEUR('P2','DUBOIS',18),  
                    T_PROFESSEUR('P4','DURACUIRE',15),  
                    T_PROFESSEUR('P5','DUBALAI',32))  
);  
  
INSERT INTO Ecole2 VALUES (  
    'CE2',  
    T_Velevés(T_ELEVE(13,'RIBOULDINGUE'),  
              T_ELEVE(17,'SPIROU'),  
              T_ELEVE(22,'BOULE')),  
    T_Vprofesseurs(T_PROFESSEUR('P3','DUGENOU',30),  
                    T_PROFESSEUR('P4','DURACUIRE',30),  
                    T_PROFESSEUR('P6','DUVAL',28))  
);
```

Reprendons maintenant les requêtes précédentes :

Rq1 : Donner la liste des élèves de la classe 'CM1'

Rq2 : Donner la liste des élèves qui ont 'DUBALAI' comme professeur

Rq3 : Donner le nombre d'heures effectuées par le professeur 'DUBALAI'

La requête Rq1 peut être formulée de manière analogue à la méthode employée avec les tables imbriquées :

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The top menu has icons for Home, Logout, and Help. Below the menu, it says "Utilisateur: ECOLER". The main area shows a SQL command window with the following code:

```
SELECT e.numE, e.nomE FROM THE (SELECT ee.Velevres FROM Ecole2 ee  
WHERE ee.nomC = 'CM1')e;
```

Below the code, there are tabs for "Résultats", "Expliquer", "Décrire", "SQL enregistré", and "Historique".

La deuxième requête Rq2 est également analogue à celle des tables imbriquées :

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The top menu has icons for Home, Logout, and Help. Below the menu, it says "Utilisateur: ECOLER". The main area shows a SQL command window with the following code:

```
SELECT e.numE, e.nomE FROM Ecole2 q, TABLE(q.Velevres) e,  
TABLE(q.Vprofesseurs) p WHERE p.nomP='DUBALAI' ORDER BY  
q.nomC,e.numE
```

Below the code, there are tabs for "Résultats", "Expliquer", "Décrire", "SQL enregistré", and "Historique".

Enfin, la troisième requête Rq3 se traite aussi de manière analogue :

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The top menu has icons for Home, Logout, and Help. Below the menu, it says "Utilisateur: ECOLER". The main area shows a SQL command window with the following code:

```
SELECT SUM(p.nbh) FROM Ecole2 q, TABLE(q.Vprofesseurs) p WHERE  
p.nomP='DUBALAI';
```

Below the code, there are tabs for "Résultats", "Expliquer", "Décrire", "SQL enregistré", and "Historique".

The results section shows a table with one row:

SUM(P.NBH)
50

Below the table, it says "1 lignes renvoyées en 0,12 secondes" and "Export CSV".

Utilisation de pointeurs

Imaginons maintenant que l'on utilise des pointeurs dans les tables et considérons la situation suivante :

ELEVEP			CLASSEP		PROFESSEURP		SERVICEP		
numE	nomE	@nomC	nomC		numP	nomP	@numP	@nomC	nbh
10	FRICOTIN	@CM1	CM1		P1	DUPONT	@P1	@CM1	12
11	FILOCHARD	@CM2	CM2		P2	DUBOIS	@P1	@CM2	25
12	CROQUIGNOL	@CE1	CE1		P3	DUGENOU	@P2	@CE1	18
13	RIBOULDINGUE	@CE2	CE2		P4	DURACUIRE	@P3	@CE2	30
14	TARZAN	@CM1			P5	DUBALAI	@P3	@CM2	28
15	TINTIN	@CM2			P6	DUVAL	@P4	@CE1	15
16	MILOU	@CE1					@P4	@CE2	30
17	SPIROU	@CE2					@P5	@CE1	32
18	FANTASIO	@CM1					@P5	@CM1	18
19	MICKEY	@CM2					@P6	@CM1	25
20	DONALD	@CE1					@P6	@CE2	28
22	BOULE	@CE2							
23	BILL	@CM1							

où le signe @truc indique un pointeur vers truc.

Commençons par créer les tables

```

CREATE TYPE T_CLASSEP AS OBJECT (nomC varchar2(10));
CREATE TABLE CLASSEP OF T_CLASSEP (PRIMARY KEY (nomC));
CREATE TYPE T_PROFESSEURP AS OBJECT (numP VARCHAR2(2),nomP
VARCHAR2(30));
CREATE TABLE PROFESSEURP OF T_PROFESSEURP (PRIMARY KEY (numP));
CREATE TYPE T_ELEVEP AS OBJECT (numE INTEGER,nomE VARCHAR2(30),refnomC
REF T_CLASSEP);
CREATE TABLE ELEVEP OF T_ELEVEP (PRIMARY KEY (numE));
CREATE TYPE T_SERVICEP AS OBJECT (refnumP REFT_ PROFESSEURP,refnomC REF
T_CLASSEP,nbh INTEGER);
CREATE TABLE SERVICEP OF T_SERVICEP (PRIMARY KEY (refnumP,refnomC));

```

Insérons les données non pointeurs avec la méthode du copier/coller. Pour les pointeurs, on peut utiliser UPDATE :

ORACLE Database Express Edition

Utilisateur: ECOLER

Page d'accueil > SQL > **Commandes SQL**

Validation automatique Afficher 10 Enregistrer Exécuter

```
UPDATE ELEVEP e
SET e.refnomC = (SELECT REF(c) FROM CLASSEP c WHERE c.nomC='CM1')
WHERE e.numE=10;
```

Résultats Expliquer Décrire SOL enregistré Historique

1 ligne(s) mise(s) à jour.

Il faut agir de même pour toutes les lignes de la table ELEVEP. Pour remplir la table SERVICEP, on peut utiliser INSERT :

ORACLE Database Express Edition

Utilisateur: ECOLER

Page d'accueil > SQL > **Commandes SQL**

Validation automatique Afficher 10 Enregistrer Exécuter

```
INSERT INTO SERVICEP VALUES
(
    (SELECT REF(p) FROM PROFESSEURP p WHERE p.numP='P1'),
    (SELECT REF(c) FROM CLASSEP c WHERE c.nomC='CM1'),
    12
);
```

Résultats Expliquer Décrire SOL enregistré Historique

1 ligne(s) insérée(s).

Il faut faire de même pour toutes les lignes.

Examinons maintenant les requêtes.

Rq1 : Donner la liste des élèves de la classe 'CM1'

ORACLE® Database Express Edition

Utilisateur: ECOLER

Page d'accueil > SQL > Commandes SQL

Validation automatique Afficher 10

```
SELECT e.numE, e.nomE FROM ELEVEP e WHERE e.refnomC.nomC='CM1';
```

Résultats Expliquer Décrire SQL enregistré Historique

HUME	NOME
10	FRICOTIN
14	TARZAN
18	FANTASIO
23	BILL

4 lignes renvoyées en 0,03 secondes [Export CSV](#)

Rq2 : Donner la liste des élèves qui ont 'DUBALAI' comme professeur

ORACLE® Database Express Edition

Utilisateur: ECOLER

Page d'accueil > SQL > Commandes SQL

Validation automatique Afficher 10

```
SELECT e.numE, e.nomE FROM ELEVEP e, SERVICEP s WHERE e.refnomC.nomC=s.refnomC.nomC AND s.refnumP.nomP='DUBALAI';
```

Résultats Expliquer Décrire SQL enregistré Historique

HUME	NOME
10	FRICOTIN

Rq3 : Donner le nombre d'heures effectuées par le professeur 'DUBALAI'

ORACLE® Database Express Edition

Utilisateur: ECOLER

Page d'accueil > SQL > Commandes SQL

Validation automatique Afficher 10

```
SELECT SUM(s.nbh) FROM SERVICEP s WHERE s.refnumP.nomP='DUBALAI';
```

6. Exercices

Exercice 1

On considère une base de données relationnelle constituée de trois tables :

LIVRE (numL, auteur, titre, prix)

MAGASIN (numM, ville, cp)

STOCK (numM, numL, qte)

où numL est un numéro de livre, auteur est l'auteur principal, cp est un code postal, numM est un numéro de magasin, qte est la quantité en stock d'un livre donné dans un magasin donné.

On désire utiliser le modèle objet-relationnel d'Oracle>8i dans lequel on considérera une table unique dont la structure est donnée graphiquement ci-dessous :

LIVRES							
numL	auteur	titre	prix	numM	ville	cp	qte
100	Jules Verne	Cinq semaines en ballon	12	M1	Paris	75005	50
				M2	Lyon	69000	30
				M3	Marseille	13000	40
				M4	Bordeaux	33000	45
101	Alexandre Dumas	Les trois mousquetaires	15	M1	Paris	75005	25
				M2	Lyon	69000	35
				M3	Marseille	13000	50
				M4	Bordeaux	33000	60
102	Molière	Le bourgeois gentilhomme	8	M1	Paris	75005	30
				M2	Lyon	69000	40
				M3	Marseille	13000	50
				M4	Bordeaux	33000	30

1) On souhaite réaliser cette structure avec une table imbriquée (NESTED TABLE)

- a) Créer la table en SQL3/Oracle
- b) Insérer avec SQL3/Oracle les données ci-dessus
- c) Formuler en SQL3/Oracle les requêtes suivantes :

Rq1 : Donner la liste des livres proposés par le magasin M1 et leur quantité en stock.

Rq2 : Donner le stock total pour le livre 100.

2) On désire réaliser cette structure avec des VARRAY. Répondre aux mêmes questions que ci-dessus.

Exercice 2

On considère une base de données relationnelle composée de deux tables et représentant des équipes de joueurs.

JOUEUR(numJ, nomJ, #numE)

Cours d'Ateliers de Génie Logiciel

EQUIPE (numE, nomE, eff)

Où numJ et numE sont des clés, nomJ le nom d'un joueur, numE, le nom d'une équipe, eff, l'effectif d'une équipe.

JOUEUR			EQUIPE	
numJ	nomJ	numE	numE	nomE
J1	DUBOIS	E1	E1	Les as du ballon
J2	DUPONT	E2	E2	Les rois de la pelouse
J3	DUVAL	E1		
J4	DUGENOU	E2		
J5	DURACUIRE	E1		
J6	DUPUIS	E2		
J7	DUBOUT	E1		
J8	DUPRE	E2		

- 1) On désire représenter cette base de données dans le modèle objet-relationnel en utilisant les types abstraits de données. Indiquer comment en SQL3/Oracle on peut définir la structure de la base de données.
- 2) Indiquer comment insérer dans la base de données objet relationnel la 1ère ligne de la table JOUEUR et la 1ère ligne de la table EQUIPE.
- 3) On souhaite rassembler les deux tables précédentes en une table unique TEAM ayant la structure suivante (méthode des tables imbriquées) :

TEAM				
numE	nomE	eff	joueurs	
			numJ	nomJ
E1	Les as du ballon	4	J1	DUBOIS
			J3	DUVAL
			J5	DURACUIRE
			J7	DUBOUT
E2	Les rois de la pelouse	4	J2	DUPONT
			J4	DUGENOU
			J6	DUPUIS
			J8	DUPRE

Indiquer comment créer la structure de cette table avec des instructions SQL3/Oracle.

5) Insérer avec SQL3/Oracle dans cette table les données relatives à l'équipe E1.

6) Formuler en SQL3/Oracle la requête suivante :

- a) Quels sont les joueurs (numJ, nomJ) de l'équipe E1 ?
- b) A quelle équipe appartient le joueur "DURACUIRE" ?
- c) Vérifier que les effectifs sont exacts.

Exercice 3

Des composants entrent dans la fabrication de produits selon les règles suivantes

Un produit est élaboré avec une quantité donnée de chaque composant entrant dans sa fabrication

Un composant peut entrer dans la fabrication de produits différents.

La base de données relationnelle correspondante comprend 3 tables :

COMPOSANT (numC, nomC, uniteC)

PRODUIT (numP, nomP)

COMPOSITION (#numC, #numP, qte)

numC est un identifiant unique d'un composant caractérisé par ailleurs par un nom, nomC, et une unite, uniteC, de comptage (kilo, litre, pièce, ...).

numP est un identifiant unique d'un produit caractérisé aussi par un nom, nomP.

Pour un produit donné (numP), un composant (numC) entre dans sa fabrication selon une quantité (qte) exprimée en uniteC.

Pour fixer les idées, une instanciation partielle de la base est donnée ci-dessous:

COMPOSANT			PRODUIT		COMPOSITION		
numC	nomC	uniteC	numP	nomP	numC	numP	qte
C1	aaaaaa	kg	P1	XXXXXX	C1	P1	3
C2	bbbbbb	pièce	P2	YYYYYY	C3	P1	10
C3	cccccc	pièce	P3	ZZZZZ	C5	P1	15
C4	ddddd	kg	P4	TTTTT	C2	P2	4
C5	eeeeee	douzaine			C3	P2	5
C6	ffffff	pièce			C4	P3	12
					C5	P3	2
					C6	P3	20
					C1	P4	10
					C2	P4	4

1ère partie

On souhaite passer du modèle relationnel au modèle objet-relationnel et utiliser les possibilités de SQL3/Oracle.

1) Indiquer, en précisant les ordres SQL, comment construire la base de données objet-relationnel en utilisant des types abstraits de données.

2ème partie

1) On souhaite remplacer les trois tables par une table unique faisant appel à la technique des tables imbriquées (Nested Tables). La structure de la nouvelle table FABRICATION est donnée ci-dessous en reprenant l'instanciation initiale :

FABRICATION					
numP	nomP	composant			
		numC	nomC	uniteC	qte
P1	XXXXX	C1	aaaaaa	kg	3
		C3	ccccc	kg	10
		C5	eeeeee	douzaine	15
P2	YYYYY	C2	bbbbbb	pièce	4
		C3	cccccc	kg	5
P3	ZZZZZ	C4	ddddd	kg	12
		C5	eeeeee	douzaine	2
		C6	fffffff	pièce	20
P4	TTTTT	C1	aaaaaa	kg	10
		C2	bbbbbb	pièce	4

Indiquer en SQL3/Oracle comment définir la structure d'une telle table.

- 2) On souhaite insérer dans la table FABRICATION le produit P1 (complet). Indiquer en SQL3/Oracle comment procéder.
- 3) Formuler en SQL3/Oracle la requête suivante : Quels sont les composants (numC, nomC, qte) se comptant par kg entrant dans la fabrication du produit P1 ?

Exercice 4

Une certification professionnelle est basée sur un examen en ligne présentant des questions à choix multiples. Ces questions sont enregistrées dans une base de données relationnelle dont le schéma est donné ci-dessous :

QUESTION (numQ, questionQ)

REPONSE (#numQ, numR, reponseR, bonneR)

Le signe "#" indique une clé étrangère. numQ désigne le numéro unique d'une question, questionQ est le libellé de la question. Il peut y avoir pour chaque question des réponses proposées numérotées avec numR. reponse R est le libellé d'une réponse et bonneR est un booléen prenant les valeurs oui ou non suivant que la réponse est bonne ou mauvaise.

I – Passage au modèle objet relationnel

- 1) Créer en SQL3/Oracle les types de données nécessaires
- 2) Créer en SQL3/Oracle la structure des tables QUESTION et REPONSE dans le cadre du modèle objet-relationnel
- 3) Insérer dans la base la question suivante :

numQ = 499

questionQ :" Un ordre " au marché " est exécuté :"

numR = 1 reponseR : " Au meilleur cours de la journée"

bonneR = "non"

Cours d'Ateliers de Génie Logiciel

```
numR = 2    reponseR : " Au plus vite"                bonneR = "oui"  
numR = 3    reponseR : " Au cours du lendemain"      bonneR = "non"  
numR = 4    reponseR : " Une fois tous les autres ordres exécutés"  bonneR = "non"
```

II – On souhaite fusionner les deux tables en une seule table QUESTIONNAIRE dont la structure est décrite ci-dessous :

- 1) Créer en SQL3/Oracle la structure de cette table
 - 2) Insérer les données du I-3

Exercice 5

Le Mira est un jeu de cartes bien connu et qui se joue par équipe. Une équipe peut comporter un nombre indéfini de joueurs. Chaque année, un championnat permet la confrontation des équipes.

Les règles de gestion sont les suivantes :

Un joueur ne peut appartenir qu'à une seule équipe

Un joueur est de nationalité unique

Une équipe ne peut appartenir qu'à un seul pays

Un match se joue entre deux équipes

Un joueur est caractérisé par un numéro unique *numJ*, un nom, *nomJ*, un prénom, *pnomJ*, un pays d'appartenance, *pays*. Une équipe est caractérisée par un numéro unique *numE*, un nom d'équipe, *nomE*, un effectif de joueurs, *effE*, et un pays d'appartenance, *pays*. Un pays est caractérisé de manière unique par son nom, *pays*. Un match est caractérisé par un numéro unique, *numM*, une date, *dateM*, un lieu, *lieuM* et les numéros des deux équipes concernées. Une équipe ayant participé à un match est qualifiée par un score, *score*.

Le modèle relationnel est défini par le schéma suivant :

JOUEUR(numJ, nomJ, pnomJ, pays, numE)

EQUIPE (numE, nomE, effE, pays)

Cours d'Ateliers de Génie Logiciel

MATCH (numM, dateM, lieuM, numE1, numE2)

RESULTAT(numM, numE, score)

Dans la relation MATCH, numE1 et numE2 sont les numéros des équipes ayant participé au match de numéro numM. Pour illustration on donne ci-dessous l'état des tables EQUIPE et JOUEUR :

EQUIPE				JOUEUR				
numE	nomE	effE	pays	numJ	nomJ	pronomJ	pays	numE
E1	Pluton	5	Allemagne	J1	WOLF	Johann	Allemagne	E1
E2	Neptune	4	Espagne	J2	LOUP	Jean	France	E3
E3	Saturne	4	France	J3	LOBO	Juan	Espagne	E2
E4	Uranus	4	Italie	J4	WILK	Jon	Pologne	E2
				J5	VARG	Oleg	Suède	E4
				J6	ULVEN	Erich	Danemark	E1
				J7	LUPO	Arturo	Italie	E4
				J8	RENARD	Joseph	France	E3
				J9	VOLPE	Benito	Italie	E4
				J10	FUCHS	Heinrich	Allemagne	E1
				J11	ZORRO	Carlos	Espagne	E2
				J12	ROKA	Wilfrid	Hongrie	E4
				J13	TILKI	Kemal	Turquie	E3
				J14	LISKA	Jos	Slovaquie	E1
				J15	VLK	Karl	Slovaquie	E2
				J16	FARKAS	Wladimir	Hongrie	E1
				J17	RAEVEN	Stan	Danemark	E3

On souhaite passer du modèle relationnel au modèle objet-relationnel et utiliser les possibilités de SQL3/Oracle.

1ère partie

- 1) Indiquer, en précisant les ordres SQL, comment construire la base de données objet-relationnel en utilisant des types abstraits de données.
- 2) Indiquer en SQL3/Oracle comment remplir la table EQUIPE
- 3) On souhaite remplacer dans la table JOUEUR, le champ numE par un pointeur vers un enregistrement de la table EQUIPE. Indiquer comment en SQL3/Oracle cela peut être déclaré.

2ème partie

- 1) On souhaite remplacer les deux tables JOUEUR et EQUIPE par une table unique Z faisant appel à la technique des tables imbriquées (Nested Tables). La structure de la table Z est donnée ci-dessous :

Z				joueurs			
numE	nomE	effE	pays	numJ	nomJ	pnomJ	pays
E1	Pluton	5	Allemagne	J1	WOLF	Johann	Allemagne
				J6	ULVEN	Erich	Danemark
				J10	FUCHS	Heinrich	Allemagne
				J14	LISKA	Jos	Slovaquie
				J16	FARKAS	Wladimir	Hongrie
E2	Neptune	4	Espagne	J3	LOBO	Juan	Espagne
				J4	WILK	Jon	Pologne
				J11	ZORRO	Carlos	Espagne
				J15	VLK	Karl	Slovaquie
E3	Saturne	4	France	J2	LOUP	Jean	France
				J8	RENARD	Joseph	France
				J13	TILKI	Kemal	Turquie
				J17	RAEVEN	Stan	Danemark
E4	Uranus	4	Italie	J5	VARG	Oleg	Suède
				J7	LUPO	Arturo	Italie
				J9	VOLPE	Benito	Italie
				J12	ROKA	Wilfrid	Hongrie

Indiquer en SQL3/Oracle comment définir la structure d'une telle table

2) On souhaite remplir la table Z avec les données de l'équipe E1 et des joueurs correspondants. Indiquer en SDQL3/Oracle comment procéder.

Exercice 6

La direction des ressources humaines d'une entreprise modélise les salariés de la manière suivante :

L'entreprise est composée de services. Chaque service est caractérisé par un numéro unique, *numS*, (achats, comptabilité, marketing, bureau d'études, ...), un nom, *nomS*, et un effectif, *effS* (nombre de personnes travaillant dans ce service à temps plein ou partiel). Un personnel est caractérisé par un identifiant unique, *numP*, un nom, *nomP*, un numéro de téléphone, *telP*.

Chaque personnel appartient à un type, *type*, (ingénieur, ouvrier, comptable, ...) et un seul. Chaque personnel peut être au cours d'une année être affecté à plusieurs services dans lequel il effectue un nombre d'heures par semaine, *nbh*, dans chaque service.

Tout le problème est à traiter dans le formalisme objet-relationnel de manière concrète (voir données ci-dessous), c'est-à-dire en utilisant Oracle XE. Pour chaque question, on donnera des copies d'écran pour prouver que la manipulation a bien été effectuée.

Tables « classiques » en relationnel :

PERSONNEL			SERVICE			AFFECTATION		
numP	nomP	telP	numS	noms	effS	numP	numS	nbh
100	DURAND	0123456789	S1	marketing	2	100	S1	24
101	DUGENOU	1234567890	S2	ventes	2	100	S2	11
102	DUPUIS	2345678901	S3	achats	1	101	S3	35
103	DUVAL	3456789012	S4	bureau d'études	2	102	S7	35
104	DUPONT	4567890123	S5	production	3	103	S8	35
105	DURACUIRE	5678901234	S6	comptabilité	1	104	S2	10
106	DURALUMIN	6789012345	S7	stocks	2	104	S6	25
107	DUBUSSON	7890123456	S8	sécurité	2	105	S4	22
108	DUBOEUF	8901234567				105	S5	13
109	DUVEAU	9012345678				106	S5	35
110	DUMOUTON	0987654321				107	S5	35
111	DUMESNIL	9876543210				108	S4	35
						109	S1	35
						110	S8	35
						111	S7	35

1ère partie

- 1) Créer, avec SQL3/Oracle, la base de données et les tables correspondant à la situation décrite plus haut.
- 2) Remplir les tables avec les données fournies ci-dessus.
- 3) Formuler et exécuter en SQL3/Oracle les requêtes suivantes

R1 : Quels sont les personnels travaillant dans le service « bureau d'études » et quel est leur service en heures ?

R2 : Quel est le nombre d'heures total du personnel « DUPONT » ?

R3 : Quels sont les types de personnel employés par le service ventes ?

R4 : Dans quel(s) service(s) se trouve le personnel dont le numéro de téléphone est « 3456789012 » ?

2ème partie

On envisage de remplacer les deux tables PERSONNEL, AFFECTATION par une table unique SALARIE en employant la méthode des tables imbriquées :

SALARIES				
Personnel			Affectation	
numP	nomP	telP	numS	nbh
100	DURAND	0123456789	S1	24
			S2	11
101	DUGENOU	1234567890	S3	35
102	DUPUIS	2345678901	S7	35
103	DUVAL	3456789012	S8	35
104	DUPONT	4567890123	S2	10
			S6	25
105	DURACUIRE	5678901234	S4	22
			S5	13
106	DURALUMIN	6789012345	S5	35
107	DUBUSSON	7890123456	S5	35
108	DUBOEUF	8901234567	S4	35
109	DUVEAU	9012345678	S1	35
110	DUMOUTON	0987654321	S8	35
111	DUMESNIL	9876543210	S7	35

- 1) Créer la structure de la table SALARIES
- 2) Remplir cette table avec les données ci-dessus
- 3) On embauche un nouveau personnel, « TARTEMPION », d'identifiant numP = 113, que l'on affecte au service « marketing » pour 8h et au service « ventes » pour 27h. Exprimer cet ajout en SQL3/Oracle.
- 4) Calculer avec une requête SQL3/Oracle le nombre d'heures totales hebdomadaires effectuées dans l'entreprise.
- 5) Calculer avec une requête SQL3/Oracle le nombre d'heures totales hebdomadaires effectuées dans le service « marketing »
- 6) Donner, avec une requête SQL3/Oracle la liste des personnels travaillant dans le service « marketing ».

Exercice 7

On considère une base de données concernant le relevé des températures de villes dans le monde. On s'appuie sur le schéma relationnel suivant :

TEMPS (#ville, date, temperature)

LOCALITE (ville, nbh, #pays)

ETAT (pays, superficie)

où les clés primaires sont soulignées et le signe # indique une clé étrangère.

avec ville : nom de ville,

nbh : nombre d'habitants d'une ville en milliers d'individus,

Cours d'Ateliers de Génie Logiciel

pays : nom d'un pays

superficie : superficie totale du pays en km²,

date : date du relevé de température

température : température relevée dans une ville à une date donnée.

On admettra que, parmi les villes considérées, il n'est pas possible de trouver le même nom de ville pour des pays différents. On admet aussi que deux pays différents ne peuvent pas porter le même nom.

On désire exprimer ce schéma dans le modèle objet-relationnel, notamment en définissant des types abstraits de données pour représenter des objets.

1) Indiquer en SQL3/Oracle comment définir les trois tables correspondant aux relations précédentes.

2) On souhaite rassembler les deux tables LOCALITE et ETAT en une seule en utilisant une table imbriquée comme l'indique le schéma ci-dessous.

LIEU			
ETAT		LOCALITE	
pays	superficie	ville	nbh
France	500 000	Paris	6 000
		Strasbourg	250
		Toulouse	300
Suisse	10 500	Genève	320
		Zurich	175
Vénézuela	435 000	Caracas	4 000
Burkina Faso	150 000	Ouagadougou	5 000
		Bobo Dioulasso	1 250

Indiquer en SQL3/Oracle comment construire cette table LIEU.

3) En SQL3/Oracle, insérer dans la table LIEU les données du tableau précédent.

4) On désire effectuer sur la base de données objet-relationnelle les requêtes suivantes que l'on formulera en SQL3/Oracle.

- Déterminer les villes de la base de données situées dans des pays de superficie supérieure à 150 000 km².
- Déterminer le nombre total d'habitants des villes dont la température est inférieure à 15°C du 1er janvier 2009 au 30 juin 2009
- Déterminer les températures maximales de chaque pays au cours de l'année.

Exercice 8

On considère une base de données relationnelle relative à un grand magasin constitué de rayons dans lesquels travaillent des vendeurs ; Le schéma relationnel est

Cours d'Ateliers de Génie Logiciel

RAYON (numR, nomR)

VENDEUR (numV, nomV, #numR)

avec numR : numéro de rayon

nomR : nom de rayon

numV : numéro de vendeur

nomV : nom de vendeur

Le symbole # indique une clé étrangère et les clés primaires sont soulignées.

- 1) On souhaite utiliser les possibilités du modèle objet-relationnel et n'utiliser qu'une seule table contenant une table imbriquée.

MAGASIN			
RAYONS		VENDEURS	
numR	nomR	numV	nomV
1	Librairie	10	DUGENOU
		11	DUBOIS
2	Vêtements homme	12	DUPONT
		13	DURACUIRE
3	Vêtements femme	14	DUVAL
		15	DUBALAI
		16	DUMONT
4	Cuisine	17	DUMESNIL
		18	DUPUIS
		19	DURAND
5	Produits entretien	20	DURALUMIN

En SQL3/Oracle

- a) définir les types abstraits de données nécessaires et créer la table
- b) insérer le rayon ('1', 'Librairie') et les vendeurs associés ('10', 'DUGENOU') et ('11', 'DUBOIS')

III.2. Travaux Pratiques avec WinDev

WinDev est un atelier de génie logiciel (AGL) conçu pour le développement d'applications. L'entreprise française PC SOFT a édité la première version de WinDev en 1993. Depuis lors, d'autres versions ont vu le jour afin de mettre à jour le logiciel. La 25^e version est la dernière mise à jour de l'AGL et est sortie le 10 décembre 2019. De plus, WinDev possède son propre langage : le Wlanguage. Cet atelier de génie logiciel est principalement conçu pour Windows, Linux, .Net et Java. Par ailleurs, WinDev est surtout répandu en France.

Cet atelier de génie logiciel possède également des dérivées comme le WinDev Mobile et le WebDev. Le premier est l'environnement de développement dédié aux solutions mobile sous iOS et Android. Le second est l'environnement de développement dédié aux solutions web.

Le W-language est un langage :

- ✓ Procédural
- ✓ Orienté Objet
- ✓ de 4ème génération (5 depuis Windev 7.5)
- ✓ qui propose une syntaxe simplifiée (proche du langage naturel)
- ✓ qui éloigne certains détails techniques (gestion mémoire)

Exemple de code en W-Language :

```
A est un entier //Commentaire  
S est une Chaîne = "Bonjour Monde"  
Si OuiNon("Voulez vous quitter ?")  
    Ferme()  
fin
```

A. Présentation et premier développement guidé

L'objectif de ce premier TP est de vous faire programmer de façon simple tout en vous accompagnant pour que vous preniez confiance en vous !

Pour ce premier exercice, nous allons créer un convertisseur Franc / Euro. Lancez WinDev.

La fenêtre d'accueil apparaît :

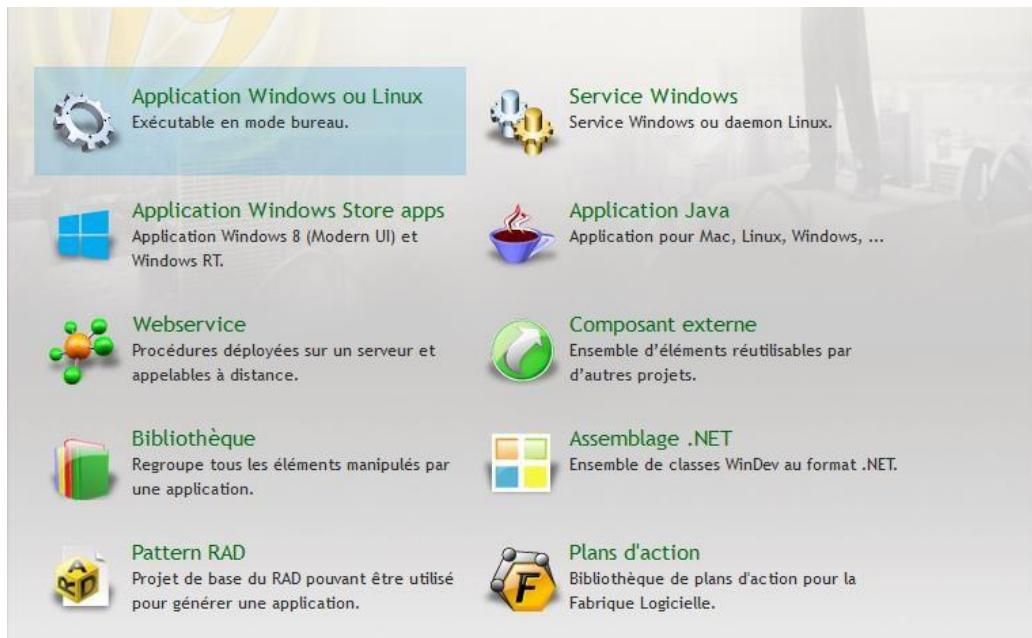


Cliquez sur Créeer un projet.

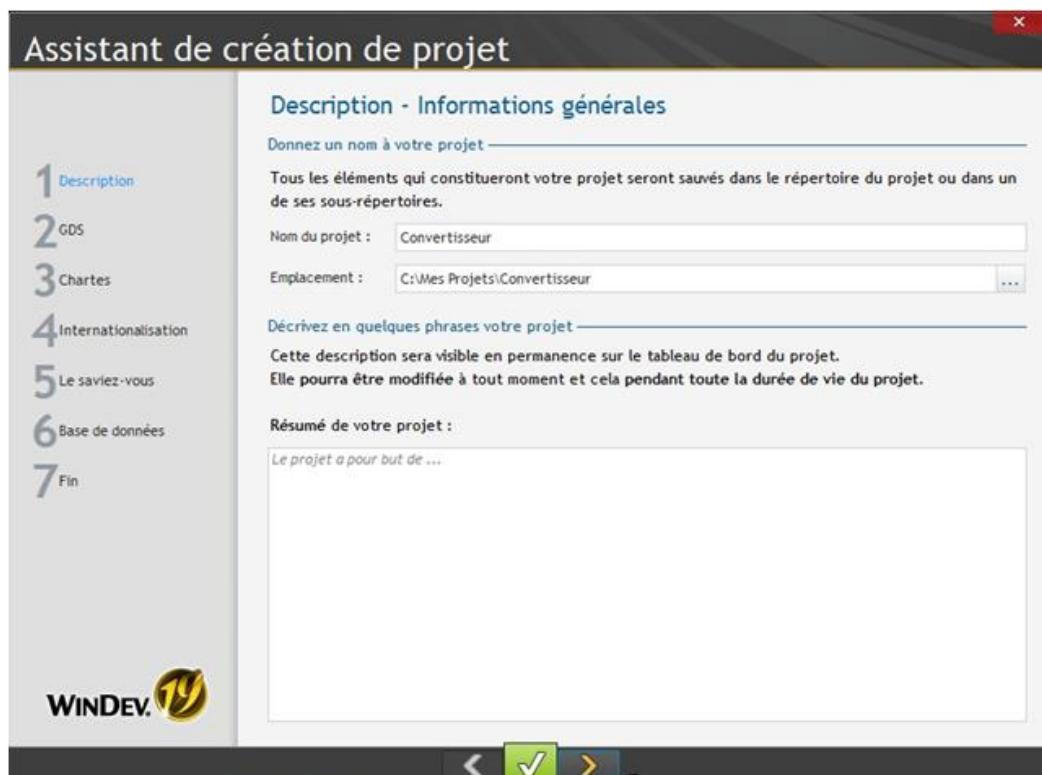
Cours d'Ateliers de Génie Logiciel

A partir de maintenant, vous allez être pris en main par un assistant qui va vous aider à définir les grandes options de votre projet de développement.

La fenêtre suivante doit apparaître :



Comme vous le voyez avec Windev, vous pouvez créer une multitude de types d'applications. Nous allons choisir « Application Windows ou Linux ». L'assistant de création de projet se met en œuvre.



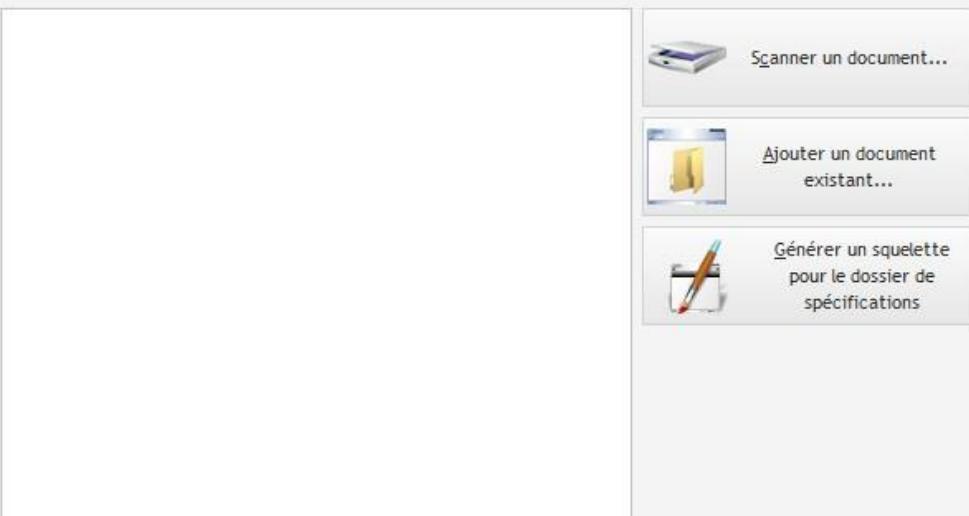
Cours d'Ateliers de Génie Logiciel

A l'emplacement Nom du projet saisissez « Convertisseur », l'emplacement va se définir par défaut. Vous pouvez rajouter un petit résumé caractérisant votre applicatif. Ensuite cliquez sur l'icône permettant d'aller sur l'écran suivant.

Description - Documents joints

Les projets commencent souvent par des spécifications, des schémas ou des notes manuscrites. Vous pouvez scanner ces documents et les conserver comme pièces jointes à votre projet. Vous pouvez également joindre toutes sortes de documents à votre projet (documents Word, Excel, images, etc.).

Documents joints au projet :



Enlever de la liste

Là, vous avez la possibilité de rajouter des documents au projet, on pourrait imaginer que vous voulez stocker des documents afférents à votre logiciel. Par exemple les modèles de facture, de bon de commande, d'ordre de fabrication qui existe chez votre client.

Nous, vu l'envergure de notre projet, nous nous contenterons juste de passer à l'écran suivant !

Description - Système d'exploitation

Quel est le système d'exploitation pour lequel est destiné votre projet ?

 **Plate-forme Windows**
Déploiement sur une plate-forme Windows.

 **Plate-forme Linux**
Déploiement d'une application sur une plate-forme Linux.
(utilisation de la librairie Qt)

Cours d'Ateliers de Génie Logiciel

Ici, l'assistant vous demande de préciser votre choix de génération, ou même la plateforme sur laquelle votre projet va s'exécuter. Comme vous le voyez, vous pouvez créer des logiciels qui vont pouvoir s'exécuter sur une plateforme Windows (32 ou 64 bits) ou sur une plateforme Linux.

Nous, nous allons choisir de créer un exécutable (.exe) pour la plateforme Windows. Laissons donc le choix par défaut et passons à l'écran suivant pour confirmer notre choix.

Description - Plate-forme

Choisissez le mode d'exécution de votre application.



Exécutable Windows 32 bits

Exécutable Windows fonctionnant sur les plates-formes 32 et 64 bits.



Exécutable Windows 64 bits

Exécutable Windows fonctionnant sur les plates-formes 64 bits uniquement.

Ici, nous confirmons que la plate-forme de destination est bien une plate-forme Windows 32 bits. Passons à la suite.

Gestionnaire De Sources (GDS)

Le Gestionnaire De Sources (GDS) est un puissant outil de versioning permettant de stocker des projets dans une base de données.

Il est fondamental dans le cas du travail en équipe et est également utile si vous travaillez seul.

Le GDS permet entre autre de partager des éléments, de conserver l'historique des modifications, de visualiser les différences entre 2 versions, ...

Voulez-vous utiliser le Gestionnaire De Sources pour votre projet ?



Oui, utiliser le GDS

Je veux utiliser le GDS pour stocker les éléments du projet.



Aide du GDS



Non, ne pas utiliser le GDS

Je ne veux pas utiliser le GDS.

Cours d'Ateliers de Génie Logiciel

Le Gds permet de centraliser les éléments constitutifs du projet sur un serveur. Cela est utile pour partager les sources avec plusieurs développeurs. Dans notre cas nous choisissons le choix Non, ne pas utiliser le GDS.

Chartes - Charte de programmation

Vous pouvez utiliser une charte de programmation pour faciliter la lecture de votre code WLangage, c'est-à-dire préfixer automatiquement les variables WLangage et les champs créés sous l'éditeur.

Voulez-vous activer le préfixage automatique des variables et des champs ?



Oui, utiliser la charte de programmation ci-dessous :

<Standard>



Editor ▾



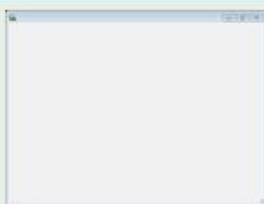
Non, ne pas utiliser une charte de programmation

Je ne veux pas activer le préfixage automatique des variables, champs, fichiers, rubriques, etc.

La charte de programmation vous préfixe les noms de variables de façon automatique. Nous allons faire sans.

Chartes - Charte graphique

Liste des gabarits



<Aucun>



ActivAndroid 4-HoloDark



ActivAndroid 4-HoloLight



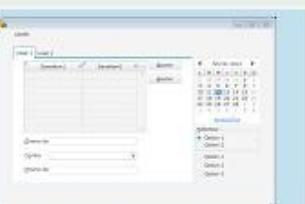
ActivPhone 5



Evolution2

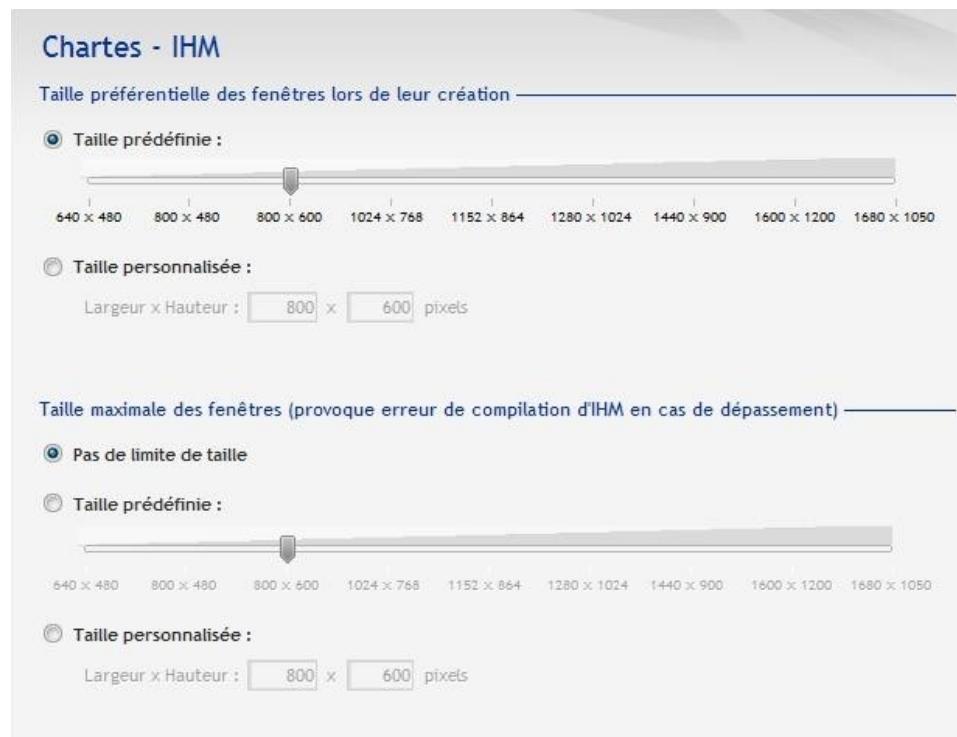


FBooking2

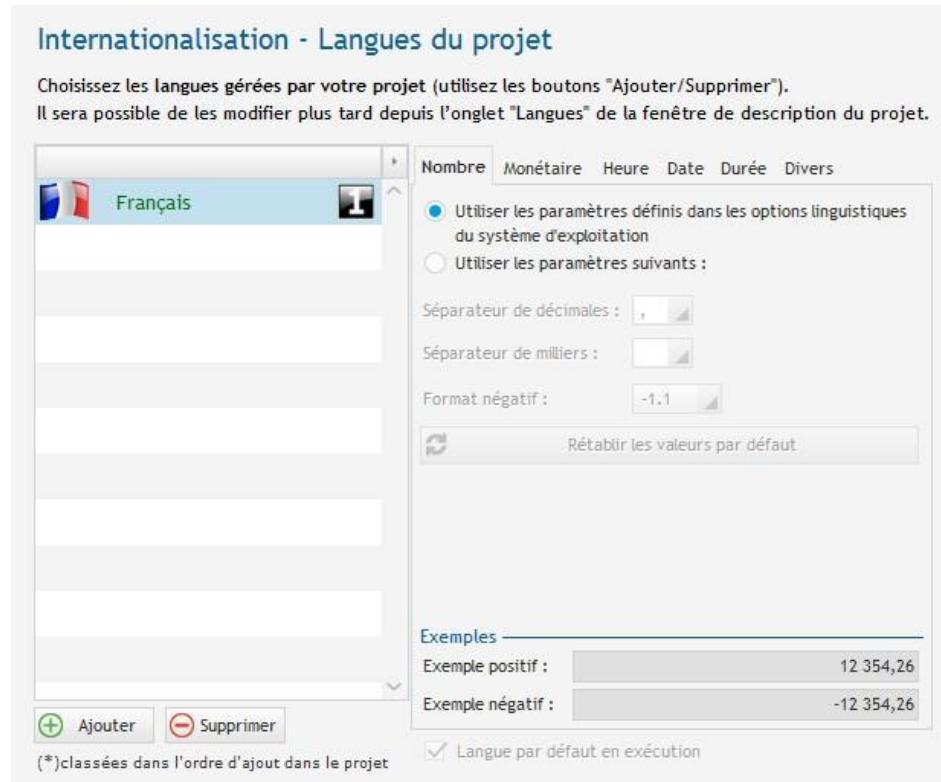


Cours d'Ateliers de Génie Logiciel

La charte graphique vous permet de donner un look sympa et sans effort à votre application.
Pour ma part j'ai choisi ActivUbuntu.



Cet assistant nous propose de définir dès à présent la taille de nos fenêtres par défaut.
Laissons les choix proposés par défaut et passons à l'écran suivant.



Cours d'Ateliers de Génie Logiciel

Un programme peut être prévu pour plusieurs pays. Dans ce cas vous pouvez dès le début du projet définir les langues qui seront utilisées. Tous vos textes pourront être saisis en plusieurs langues et votre applicatif sera customisé dans la langue de l'utilisateur de façon automatique. Nous, nous allons rester sur le Français.

Fenêtre "Le saviez-vous"

A chaque lancement de votre exécutable, une fenêtre pourra être affichée pour informer les utilisateurs de toutes les fonctionnalités automatiques (FAA) dont dispose votre application.



Afficher la fenêtre "Le saviez-vous" au lancement de l'application



Ne pas afficher la fenêtre "Le saviez-vous" au lancement de l'application



La fenêtre "Le saviez-vous" nécessite l'intégration du composant "TipOfTheDay" dans le projet. Vous pourrez supprimer ce composant du projet à tout moment.

Ici, vous pouvez choisir de faire lancer une fenêtre « Le saviez-vous » donnant des conseils ou astuces pour votre application. Dans le cadre de notre projet nous allons nous en passer.

Base de données - Utilisation d'une base

Votre projet va-t-il accéder à une base de données ?



Oui, créer une nouvelle base de données

Je veux utiliser une base de données. Celle-ci n'existe pas encore et je veux la créer.



Oui, utiliser une base de données existante

J'ai déjà une base de données HyperFileSQL Classic ou Client/Serveur, Oracle, SQL Server, MySQL, fichier Excel, etc.



Non, ne pas utiliser de base de données

Je ne veux pas utiliser de base de données pour le moment.

Cours d'Ateliers de Génie Logiciel

Pour ce projet, nous n'allons pas utiliser de base de données, prenez le choix Non, ne pas utiliser de base de données.

Fin de la description

La description du projet est terminée.

Le projet va maintenant être créé.

Aucune option choisie n'est définitive.

Vous pourrez, pendant toute la durée de vie du projet et à tout moment, faire évoluer ou changer les paramètres du projet depuis le « Tableau de bord du projet » et la fenêtre de description du projet.

Voilà, l'assistant vient de définir certains de nos besoins et est terminé. Vous pouvez cliquer sur



pour terminer l'assistance.

The screenshot shows the WinDev software interface. On the left, there is a preview window titled "Nouvelle fenêtre" showing various window templates like "Vierge", "Zone", "Liste", etc. On the right, there are three main options:

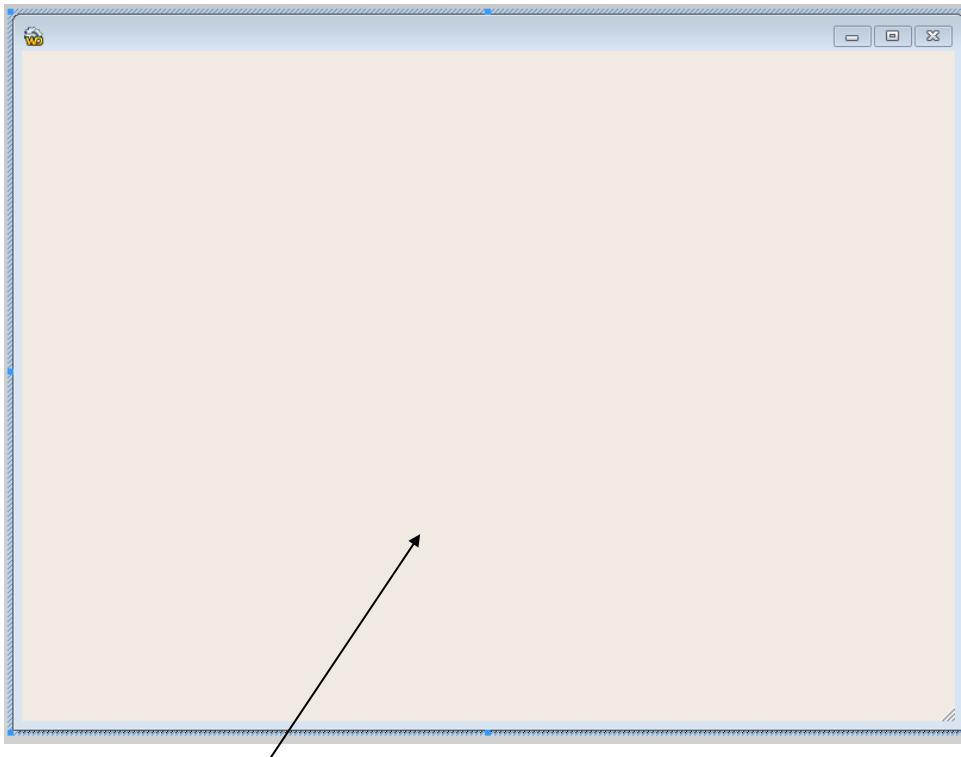
- RAD Application Complète**: Génération automatique de l'application. It features a 3D cube icon with a "RAD" logo.
- Créer une fenêtre**: Crée une fenêtre de votre choix. It features a window icon.
- Editeur de WinDev**: Allez directement dans l'éditeur de WinDev. It features the WinDev logo.

At the bottom right of the interface, there is a "Fermer" (Close) button.

L'assistant va vous poser la dernière question ? Voulez-vous créer une fenêtre ? Cliquez Créer une fenêtre puis choisissez Vierge dans l'onglet Standard. Validez par OK.

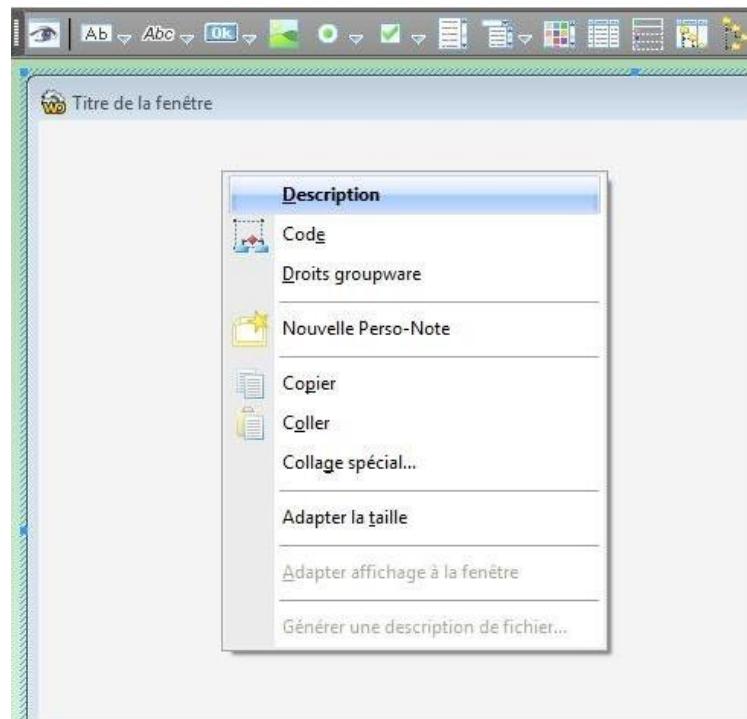
Cours d'Ateliers de Génie Logiciel

Nous voici enfin arrivé dans le vif du sujet !



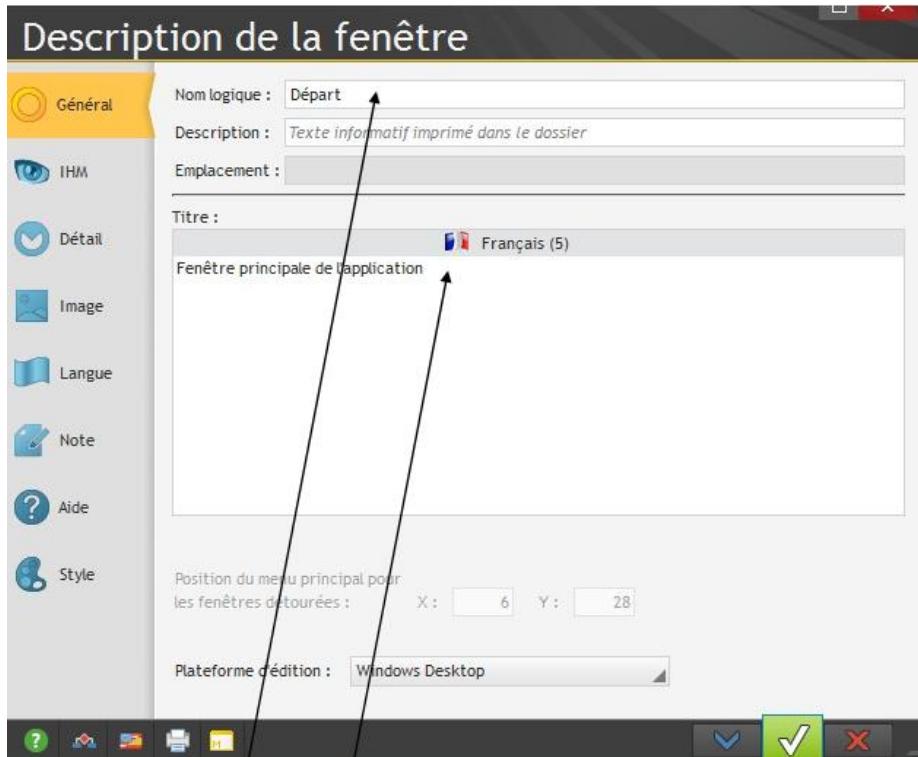
Voilà la fenêtre standard vide, nous pouvons tout modifier : le titre, la taille, les comportements par défauts etc...

Nous allons commencer les modifications de base. Pour cela placez le curseur de la souris n'importe où dans la fenêtre et faites un clic droit. Un menu contextuel doit apparaître :

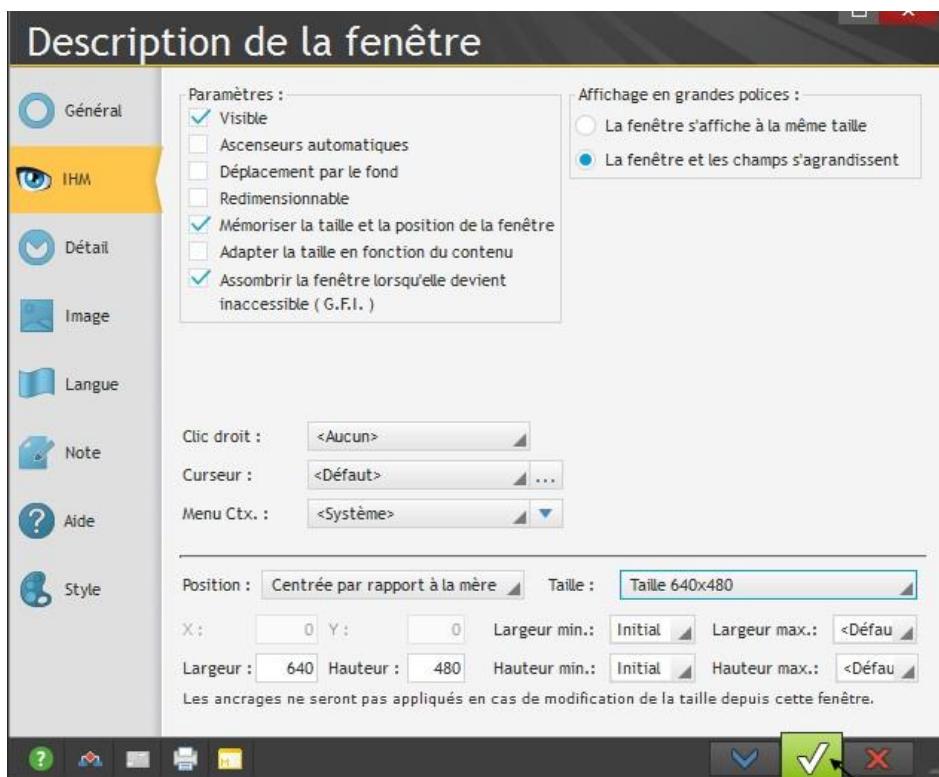


Cours d'Ateliers de Génie Logiciel

Ce menu contextuel est le point de départ de la personnalisation de la fenêtre. Cliquez sur Description :

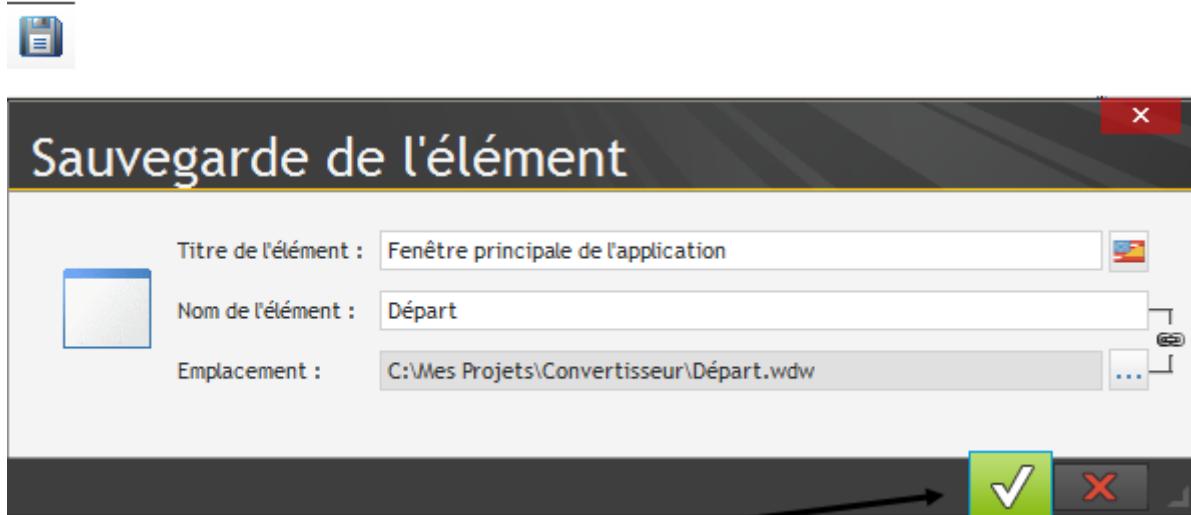


Donnez un nom à cette fenêtre et un titre. Sélectionnez l'onglet IHM (Interface Homme-Machine).



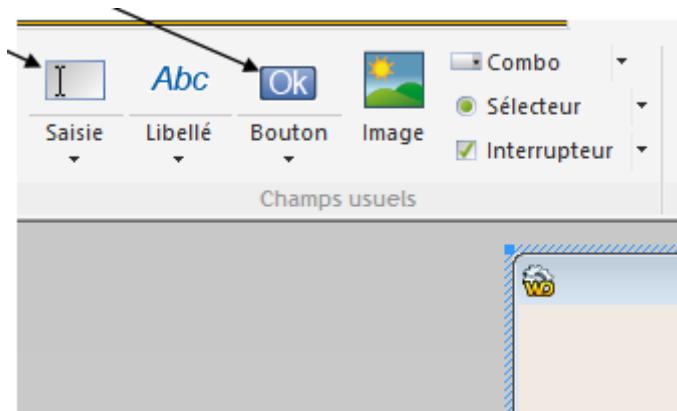
Cours d'Ateliers de Génie Logiciel

Changez la taille, le fait qu'elle ne sera pas redimensionnable puis validez en cliquant sur le jet vert. Remarquez les différences, vous avez maintenant une fenêtre avec un nom, un titre et une taille définie. Il est temps de sauvegarder, Cliquez sur le bouton Enregistrer.

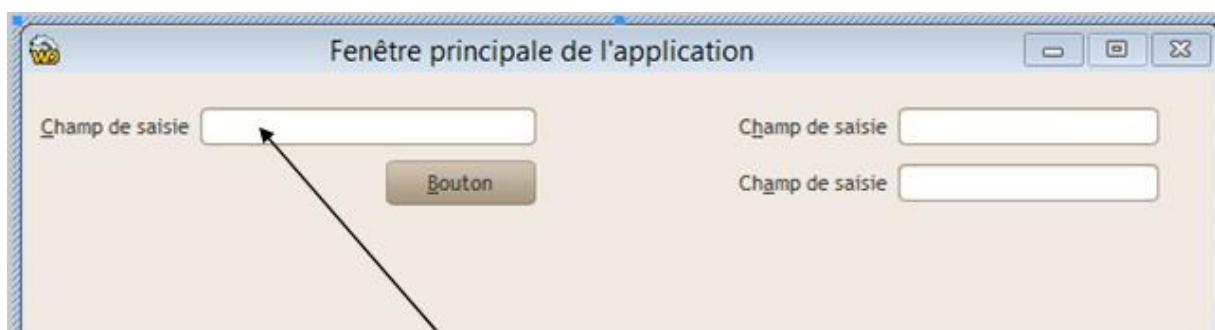


Cliquez sur le jet vert.

A l'intérieur de cette fenêtre, nous allons maintenant placer 4 objets : 3 "champs de saisie" et un bouton (faites un glisser/déposer ou drag & drop des champs dans la fenêtre)

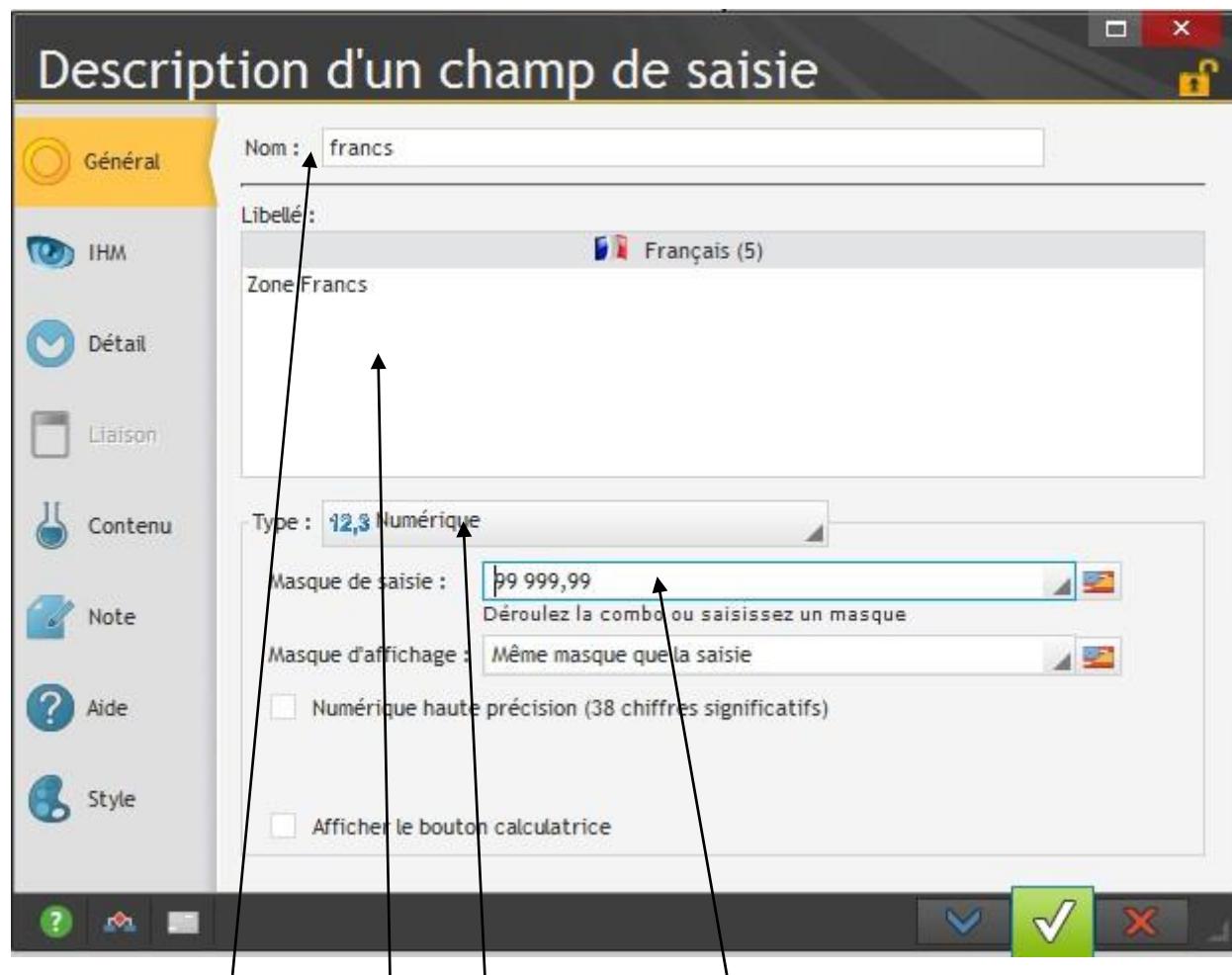


Voici à quoi pourrait ressembler votre fenêtre :



Cours d'Ateliers de Génie Logiciel

Le premier champ se nommera francs et aura comme libellé "Zone francs :". Cliquez 2 fois dessus pour en modifier les caractéristiques :



Changez son nom ici, son libellé là, son type et son masque de saisie. Son nom sera lequel cet objet sera manipulé par programmation.

Le libellé sera la zone de texte apparaissant avant la zone de saisie. Le type défini le contenu que recevra ce champ de saisie.

Le masque est un formatage des informations saisies. Ici le chiffre aura au maximum 5 chiffres avant la virgule et 2 après.

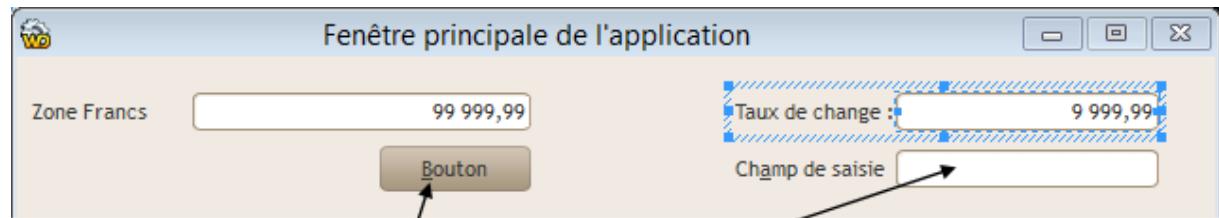
Validez cette description.



Cours d'Ateliers de Génie Logiciel

Vous voyez immédiatement le résultat ! Nous pouvons maintenant modifier le second champ.

Sur le même principe que le champ précédent le second se nommera taux et aura comme libellé "Taux de change :". Bien évidemment il sera de type numérique.

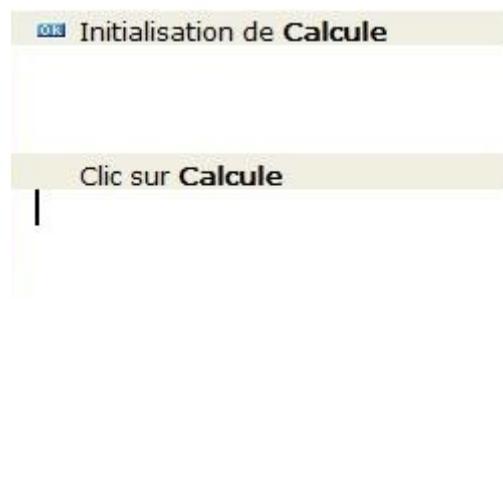


Le bouton enfin se nommera calcule et aura comme libellé "calcule".



Voici à quoi doit ressembler votre fenêtre.

Il nous reste à mettre le code correspondant dans le bouton "calcule". Pour cela, faites un clic droit dessus et choisissez "Code" dans le menu contextuel.



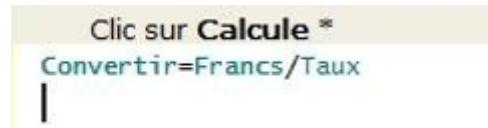
Comme vous le voyez il existe 2 zones de saisie de code : Une nommée Initialisation de Calcule et l'autre Clic sur Calcule.

Le code inscrit dans la première zone s'activera lors de la création du bouton, c'est à dire avant que la fenêtre soit active pour l'utilisateur. Ce peut être utile dans certain cas pour changer le libellé du bouton en fonction d'un contexte particulier.

Cours d'Ateliers de Génie Logiciel

Le code inscrit dans la seconde zone est celui qui nous intéresse le plus. Le code qui est dans cette partie est exécuté chaque fois que l'utilisateur clique sur le bouton.

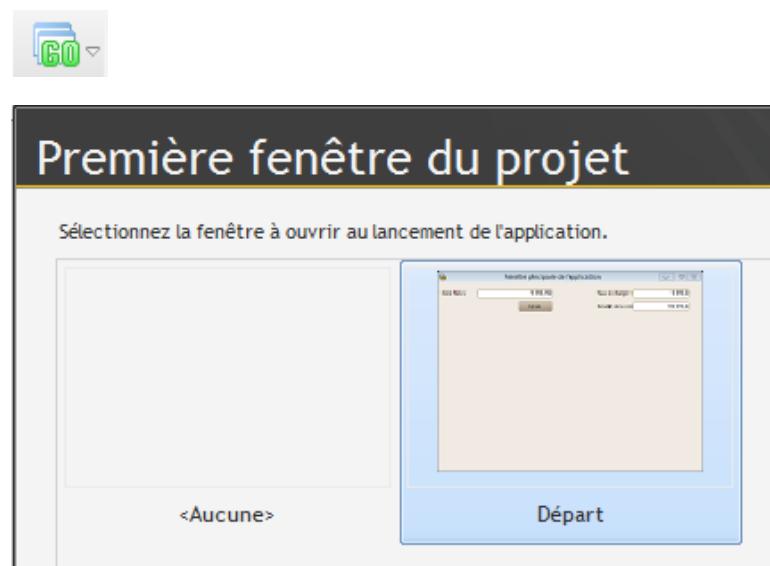
Nous allons le programmer pour que la zone convertir affiche le résultat de la conversion Francs par taux.



Saisissez comme ci-dessus. Vous allez remarquer le mécanisme de complétion automatique du code qui vous propose le nom du champ dès que vous avez saisi 3 caractères. C'est une aide appréciable !

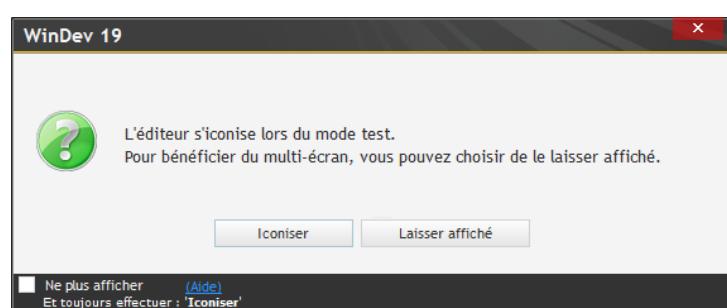
Il est temps d'enregistrer notre projet en cliquant sur l'icône d'enregistrement.

Une fois l'enregistrement achevé, nous allons tester le projet, pour cela cliquez sur Lancer le test du projet.



WinDev vous demande de définir la première fenêtre de notre méga projet, choisissez "Départ" dans la combo et validez.

Si vous voyez la fenêtre suivante, cliquez sur ne plus afficher et ensuite sur le bouton Iconiser



Cours d'Ateliers de Génie Logiciel

Vous avez maintenant devant vous notre super convertisseur. Mais avouez qu'il est franchement moche :

- ✓ trop grand ;
- ✓ Des comportements par défaut peu pratiques.

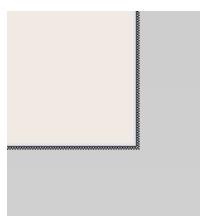
Lesquels ?

C'est simple : Essayez de saisir le taux de conversion de l'euro 6,55957 !

Comme vous le voyez, le champ ne prend que 2 chiffres après la virgule ! De plus vous allez être obligé de le saisir à chaque fois !

Nous allons remédier à tous ces petits détails.

Tout d'abord fermer l'exécutable en cliquant sur la croix en haut à droite pour revenir en mode édition. Pour la taille de la fenêtre, placez votre souris sur l'angle inférieur droit de la fenêtre "Départ" :



Une fois que le curseur change d'aspect, tenez appuyé le bouton gauche de la souris et remontez vers l'angle supérieur gauche. Relâchez la souris quand la taille souhaitée sera atteinte.

Prenez la couleur qui vous plait le plus !



Voilà mon résultat à moi !

Changeons le comportement du bouton "taux", faites un clic droit dessus, Description. Vérifiez que le type soit Numérique. Maintenant dans la zone Masque de saisie, frappez 9,99999. Appliquez les modifications puis dans le menu contextuel choisissez l'option Code.



Cours d'Ateliers de Génie Logiciel

Insérer le code : MoiMême=6.55957 dans la zone "Initialisation de Taux". Ainsi à chaque démarrage du convertisseur, la zone de saisie sera remplie. Notez que nous aurions pu écrire : taux=6.55957. MoiMême désigne l'objet dans lequel on se trouve.

Relancer le test de l'application en cliquant sur



EXERCICE APPLICATIF

Créez une application « EXO1 » contenant :

- ✓ une fenêtre
- ✓ 2 champs textes
- ✓ 1 champ numérique
- ✓ 1 bouton

Lorsque l'utilisateur clique sur le bouton, les champs textes se remplissent avec votre nom dans le premier champ, votre prénom dans le deuxième champ texte et votre année de naissance dans le champ numérique que vous aurez pré-programmés.

Vous avez toute liberté au niveau de l'ergonomie, du nommage des champs, de la fenêtre, du bouton. Je vous rappelle que tout le code doit être contenu dans le bouton.

B. Bases de WinDev

L'objectif de ce second TP est de vous faire connaitre les objets de base de WinDev, grâce à la manipulation des :

- ✓ Combo-Box,
- ✓ Interrupteurs,
- ✓ Tables Mémoire...

Pour ce second TP, nous allons créer des fenêtres comportant plusieurs objets de base dont nous allons étudier le comportement.

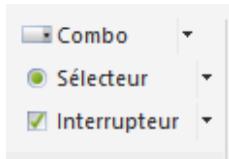
LES SELECTEURS

Créez un projet nommé TP2 et une fenêtre nommée « sélecteurs ». Le Titre de la fenêtre sera « Bonjour ».

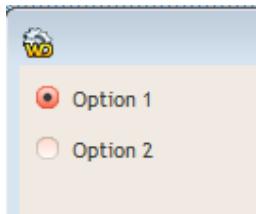
Enregistrez immédiatement cette fenêtre sous « Sélecteurs ». Il est important de le faire dès le début car ainsi WinDev vous aidera à auto compléter votre code.

Insérez un champ sélecteur à l'intérieur de votre fenêtre. Cliquez sur l'objet sélecteur et ensuite positionnez-le sur votre fenêtre de départ

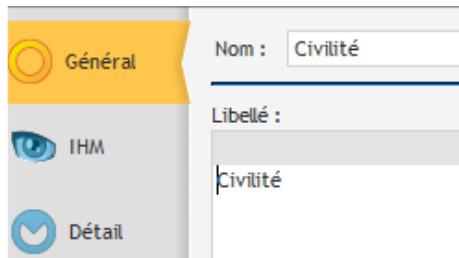
Cours d'Ateliers de Génie Logiciel



Voici ce que vous devriez découvrir à l'écran :



Allez dans la description du sélecteur (clic droit / Description / Onglet Général) dans la zone Nom du champ inscrivez : Civilité. Idem dans le Libellé du champ.



Passez maintenant sur l'onglet Contenu :

Remplissez les options comme indiqué ci-dessous. Pour rajouter la troisième option cliquez sur le symbole Plus vert.

Mode d'affichage : Standard

Options

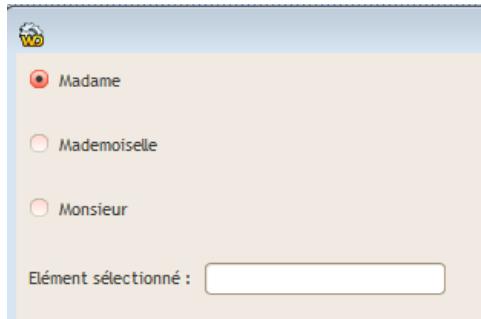
Numéro	Label	Hauteur	Valeur renvoyée
1	Madame	Auto	
2	Mademoiselle	Auto	
3	Monsieur	Auto	

CTRL+N pour ajouter une option - CTRL+T pour modifier toutes les langues

Cours d'Ateliers de Génie Logiciel

Rajoutez un champ de saisie que vous nommerez choix avec comme libellé « Elément sélectionné : ».

Votre fenêtre doit ressembler à ceci :



Vous allez faire en sorte que le champ « Elément sélectionné » se renseigne selon la Civilité. Pour cela, allez dans le Code du champ « civilité » (clic droit) dans la zone « A chaque modification de Civilité » et saisissez le code suivant :

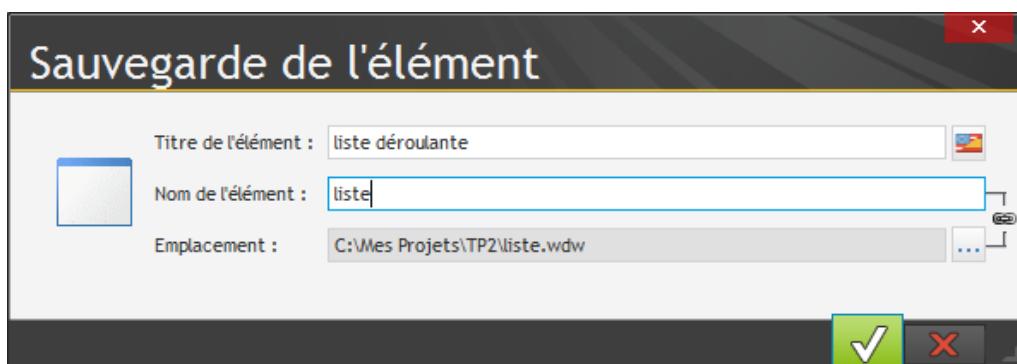
```
A chaque modification de Civilité *
SELON Civilité
    // Madame
    CAS 1
        Choix="Madame"
    // Mademoiselle
    CAS 2
        Choix="Mademoiselle"
    // Monsieur
    CAS 3
        Choix="Monsieur"
FIN
```

Comme vous le remarquez, WinDev ne renvoie pas le libellé du choix effectué mais l'index (ou position) de l'élément (1, 2 ou 3).

Testez la fenêtre en cliquant sur Go.

LES LISTES DEROULANTES

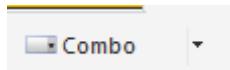
Créez une nouvelle fenêtre que vous nommerez « Liste » lorsque vous l'enregistrerez (immédiatement) et comme titre : « Liste déroulante »



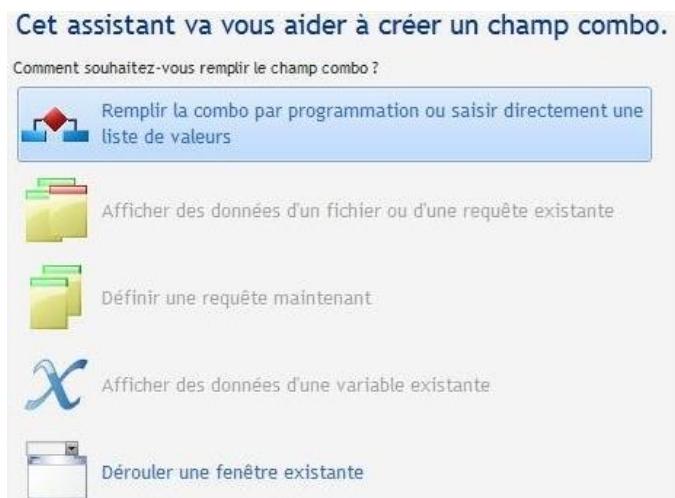
Cours d'Ateliers de Génie Logiciel

A l'intérieur, placez :

Une combo (Créer une combo)

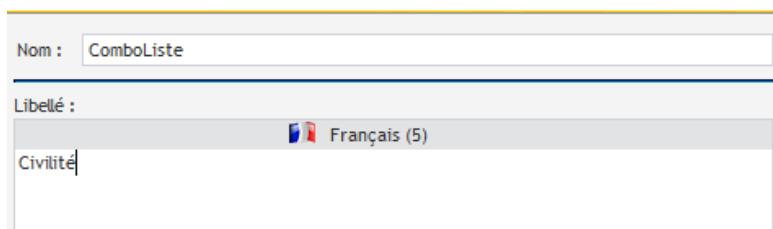


L'assistant suivant se met en œuvre :



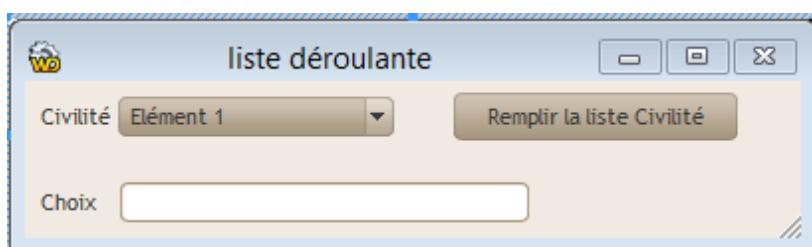
Validez le choix par défaut « Remplir la combo par programmation... », via le Jet Vert. La combo apparaît sur votre fenêtre.

Vous la nommerez « ComboListe » et elle aura comme libellé « Civilité » (ne pas saisir de contenu initial à afficher).



Maintenant, placez un champ de saisie nommé « choix » et ayant « Choix » comme libellé.

Un bouton nommé « remplir » et ayant comme libellé « Remplir la liste Civilité ». La fenêtre ressemblera à ceci :



Nous allons programmer le bouton pour qu'il remplisse la combo avec les éléments souhaités (Madame, Mademoiselle, Monsieur). Puis nous allons définir le comportement de la combo pour qu'elle affecte le champ « choix ».

Dans la zone « Clic sur remplir » du Code du bouton écrivez le code suivant :

```
Clic sur Remplir * Si Erreur : par programme Quand
ListeSupprimeTout(ComboListe) // Pour éviter d'ajouter les éléments à chaque clic sur le bouton
ListeAjoute(ComboListe,"Madame")
ListeAjoute(ComboListe,"Mademoiselle")
ListeAjoute(ComboListe,"Monsieur")
```

Nous allons vérifier que votre liste soit correctement remplie en cliquant sur le bouton Remplir puis en l'ouvrant avec la flèche vers le bas. Pour cela faites un clic de la souris pour l'avoir en exécution puis cliquez sur le bouton. Je vous laisse découvrir le résultat :



Dans la zone « sélection d'une ligne de... » du Code de la combo Civilité écrivez le code suivant :

```
Sélection d'une ligne de ComboListe
Choix=ComboListe..ValeurAffichée
```

Par cette simple ligne vous demandez à la combo de copier la valeur affichée dans le champ « choix ». Sauvegardez et testez la fenêtre, non mais !

N'oubliez pas d'utiliser l'aide pour approfondir vos connaissances en appuyant sur le bouton F1 !

LES TABLES MéMOIRES

Le champ table permet de simplifier l'affichage et la saisie d'informations stockées en mémoire ou provenant d'un fichier de données, d'une vue ou d'une requête. Une table est composée de lignes et de colonnes.

L'intersection d'une ligne et d'une colonne définit une cellule.

Une table peut être gérée ligne par ligne, colonne par colonne ou cellule par cellule. Les informations affichées dans la table peuvent :

- ✓ être déterminés complètement par programmation : on parle alors de *Table Mémoire*.

Cours d'Ateliers de Génie Logiciel

- ✓ provenir d'un fichier de données ou d'une requête : on parle alors de *Tablefichier*.

Il est bien évident que la table mémoire peut afficher les données provenant d'un ou plusieurs fichiers, ou d'une ou plusieurs requêtes.

Pour notre exemple, créez une fenêtre nommée « tablemem » et « Table mémoire » pour Titre.

Pour créer un champ de type Table :

Sous l'éditeur de fenêtres, cliquez sur l'icône Table et liste, choisissez ensuite table verticale

Cliquez dans la fenêtre à la position où le champ doit être créé. L'assistant de création d'un champ table se lance automatiquement.

Cet assistant va vous aider à créer un champ table.

Comment souhaitez-vous remplir le champ table ?



Laissez le choix par défaut et cliquez sur le Jet Vert.

Pour afficher les caractéristiques du champ, sélectionnez l'option « Description » dans le menu contextuel du champ. Deux types d'informations peuvent être visualisés :

- ✓ les informations concernant la table dans son ensemble (il suffit de sélectionner le nom de la table) ;
- ✓ les informations concernant chacune des colonnes de la table (il suffit de sélectionner le nom d'une des colonnes de la table).

Nommez la table « matable ».

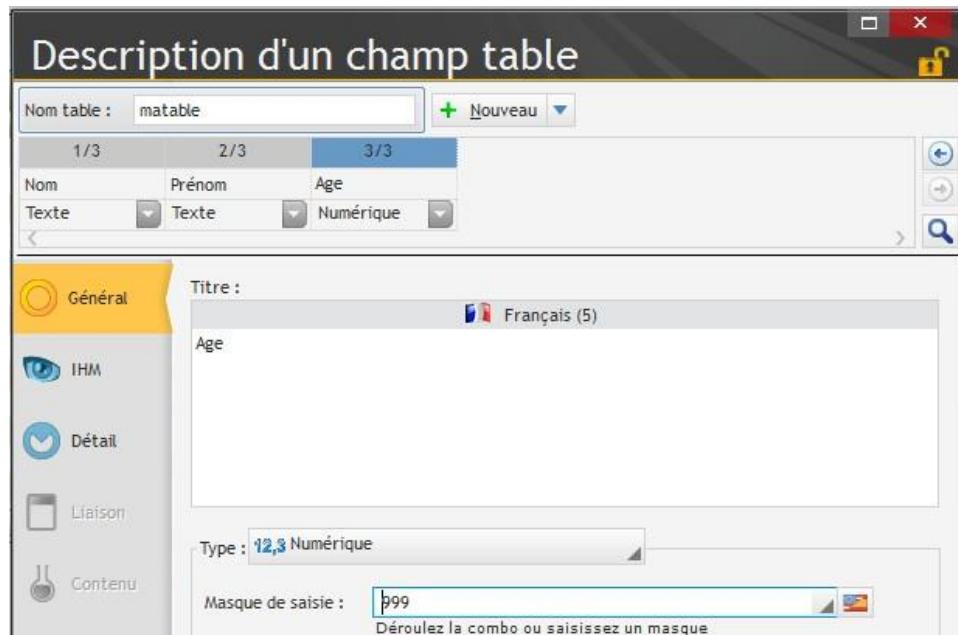
Créez 3 colonnes en appuyant 2 fois sur le bouton « Nouveau ».

Pour la première colonne : Nommez là « Nom », son type restera Texte, dans la zone « Titre » de l'onglet « Général » inscrivez « Le nom ». Dans la zone « taille de saisie » inscrivez « 50 ». Ainsi l'utilisateur ne pourra pas inscrire un nom de plus de 50 caractères.

Pour la deuxième colonne : Nommez là « Prénom », son type sera Texte, dans la zone « Titre », inscrivez « Le prénom » et 20 caractères de taille de saisie.

Cours d'Ateliers de Génie Logiciel

Pour la troisième colonne : Nommez là « Age », son type sera Numérique, dans la zone « Titre » inscrivez « Age », dans la combo « masque de saisie » trouvez le masque « 999 » (en haut de liste), cela signifie que seuls des entiers de 3 chiffres maximum seront acceptés.



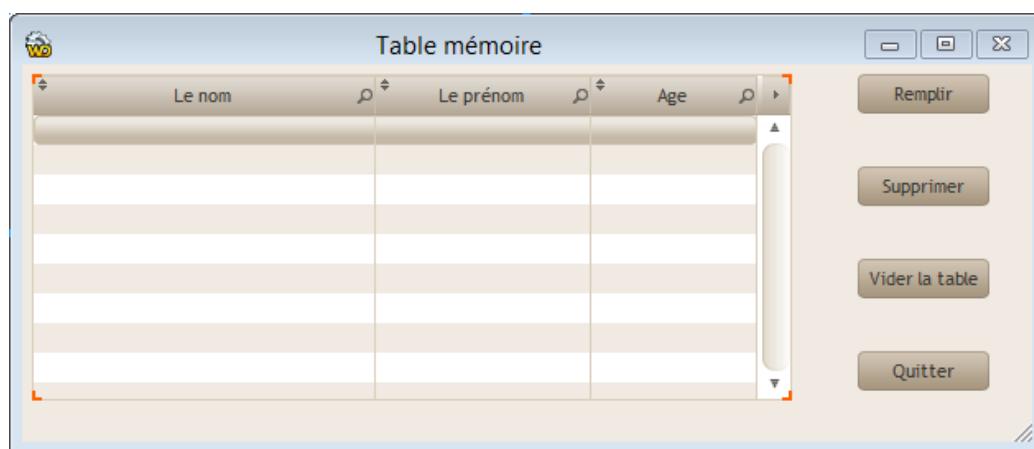
Une fois ces manipulations réalisées vous pouvez valider vos choix. Nous allons maintenant ajouter 4 Boutons :

Le premier se nommera « remplirtable » et aura comme libellé « Remplir » ;

Le second se nommera « supligne » et aura comme libellé « Supprimer » ;

Le troisième se nommera « videtable » et aura comme libellé « Vider la table » ;

Le quatrième se nommera « quitter » et comme libellé « Quitter ». Voici une représentation de votre fenêtre :



Nous allons étudier les différents codes permettant de remplir la table avec des informations, supprimer la ligne sélectionnée, vider complètement la table et enfin fermer la fenêtre.

Dans la zone « Clic sur remplirtable » du bouton « Remplir », insérez le code suivant :

Cours d'Ateliers de Génie Logiciel

Clic sur Remplirtable *

```
TableSupprimeTout(matable)
TableAjoute(matable, "Baptiste"+TAB+"Jean-Luc"+TAB+"50")
TableAjoute(matable, "Baptiste"+TAB+"Béatrice"+TAB+"46")
TableAjoute(matable, "Baptiste"+TAB+"Amandine"+TAB+"27")
TableAjoute(matable, "Baptiste"+TAB+"Cédric"+TAB+"23")
TableAjoute(matable, "Baptiste"+TAB+"Sylvain"+TAB+"18")
```

« Baptiste » correspond au Nom, « Jean-Luc » au Prénom et « 50 » à l'âge. TAB indique le changement de colonne.

Utilisez l'aide pour avoir plus de renseignements sur la fonction Tableajoute

Une autre façon de remplir la table est la suivante :

Clic sur Remplirtable *

```
TableSupprimeTout(matable)
TableAjouteLigne(matable, "Baptiste", "Jean-Luc", "50")
TableAjouteLigne(matable, "Baptiste", "Béatrice", "46")
TableAjouteLigne(matable, "Baptiste", "Amandine", "27")
TableAjouteLigne(matable, "Baptiste", "Cédric", "23")
TableAjouteLigne(matable, "Baptiste", "Sylvain", "18")
```

La différence réside dans l'absence du mot clé « Tab ». A vous de choisir la méthode qui vous convient le mieux.

Utilisez l'aide pour avoir plus de renseignements sur la fonction Tableajouteline

Dans la zone « clic sur supligne » de « Supprimer », insérez le code suivant :

Clic sur Supligne *

```
TableSupprime(Matable)
```

Ce code supprime la ligne sélectionnée dans la table.

Dans la zone « clic sur videtable » de « Vider la table », insérez le code suivant :

Clic sur videtable

```
TableSupprimeTout(Matable)
```

Dans la zone « clic sur quitter » de « Quitter », insérez le code suivant :

Clic sur quitter *

```
Ferme
```

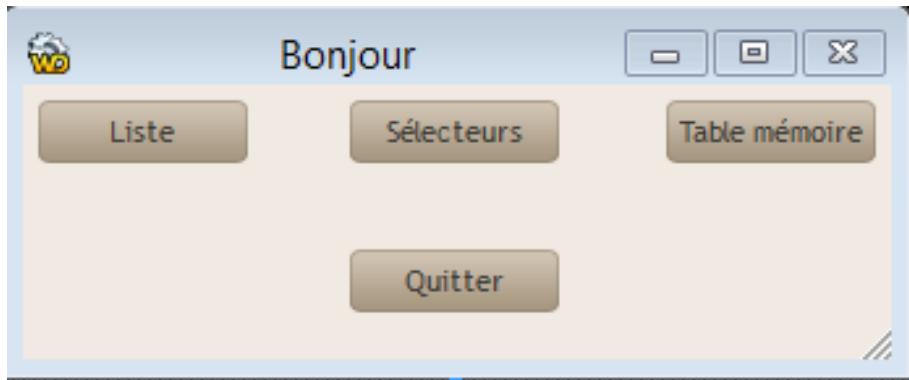
Testez les différents boutons et appuyez sur la loupe (à côté du nom de la colonne) pour tester son comportement par défaut.

Cours d'Ateliers de Génie Logiciel

Comme vous pouvez le constater, WinDev est puissant et peu de lignes de codes suffisent. Il est bien évident que le remplissage de la table peut être fait à partir de la lecture d'un fichier.

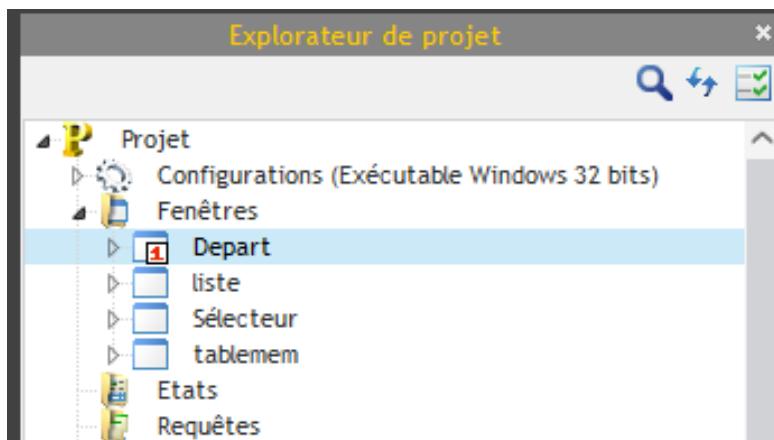
Pour terminer ce TP, il nous reste à faire une fenêtre de départ comportant 3 boutons qui ouvriront les différentes fenêtres.

Créez donc une nouvelle fenêtre vierge que vous nommerez « départ », son Titre sera « Bonjour ». Insérez-y 4 boutons : 3 serviront à lancer les fenêtres, 1 à quitter l'application :



Pour indication, le code d'ouverture d'une fenêtre est : ouvre, le code de fermeture est : ferme. Je vous laisse mettre le code correspondant.

Il ne vous restera qu'à ne pas oublier d'enregistrer cette fenêtre pour pouvoir la déclarer comme première fenêtre du projet.



Dans la zone exploratrice des éléments du projet, faites un clic droit sur la fenêtre départ et choisissez « Première fenêtre du projet ».

Exercice applicatif

Créez un projet exo1tp2 et une fenêtre nommée « départ » ressemblant à celle-ci :

Nombre	Multiplicateur	Résultat
1	7	7
2	7	14
3	7	21
4	7	28
5	7	35
6	7	42
7	7	49
8	7	56
9	7	63
10	7	70
11	7	77
12	7	84
13	7	91
14	7	98
15	7	105
16	7	112
17	7	119
18	7	126

Le but de l'exercice, comme vous le voyez, est de remplir la table en utilisant dans le code du bouton Calcul les valeurs contenues dans les champs « Multiplicateur » et « Profondeur ». Pour ce faire, vous utiliserez la structure itérative « Pour ».

Comme toujours, n'hésitez pas à utiliser l'aide.

C. Travailler avec un fichier de données

L'objectif de ce TP est de vous familiariser avec l'utilisation des fichiers, du gestionnaire d'analyse et de la conception d'états.

Vous allez commencer par créer un nouveau projet nommé TP3. Dans l'assistant, vous sélectionnerez votre thème préféré, confirmerez les choix par défauts jusqu'à ce point :



Ici, il est impératif de sélectionner le choix « Oui, créer une nouvelle base de données ». Cliquez sur Suivant.

Informations générales

Une Analyse vous permet de décrire les structures de données utilisées dans votre application ou votre site web.

Choisissez le nom de l'analyse :

Nom : Tp3

Choisissez le répertoire de l'analyse :

Répertoire : C:\Mes Projets\Tp3\Tp3.anal\



Par défaut, les informations sont correctes, vous pouvez cliquer sur suivant.

Types de bases de données

Sélectionnez les types de bases de données qui seront utilisées par le projet :

A list of database connection types:

- HyperFileSQL Classic** (selected, checked)
- HyperFileSQL Client/Serveur**
- SQL Server**
- SQL Azure**
- Oracle**
- MySQL**
- Informix**
- AS/400**
- DB2**
- Sybase**

Comme vous pouvez le constater, le choix des types de bases de données est vaste ! Nous allons travailler avec le format natif de WinDev HyperFileSQL Classic. Nous pouvons passer à l'écran suivant.

You avez décrit les informations nécessaires à la création de l'analyse.

Cliquez sur le bouton de validation pour créer l'analyse.

Voilà, la description de l'analyse est finie !

Un assistant de création de fichier va prendre le relais :

Mode de création du fichier de données



Créer une nouvelle description d'un fichier de données

Vous choisissez le nom et le type du fichier de données à créer.



Sélectionner une description parmi des fichiers de données prédefinis

Vous choisissez parmi plusieurs thèmes qui proposent des fichiers de données prédefinis que vous pourrez modifier par la suite.



Utiliser des fichiers de données d'une base existante

Vous pouvez importer des descriptions de fichiers de données depuis différents formats (HFSQL, Client/Serveur, Oracle, AS/400, etc.) avec ou sans leurs données.

Ici, nous laissons le choix par défaut.

Nous allons programmer une mini gestion de budget familial. Pour cela on va utiliser un seul fichier des mouvements dans lequel on inscrira le descriptif des opérations, la date de l'opération, le montant au débit ou le montant au crédit.

Voici la fenêtre suivante

Paramètres généraux

Nom du fichier de données _____

Nom :

Libellé :

Un enregistrement représente :

Identifiant automatique _____

- Identifiant automatique sur 8 octets (obligatoire si vous faites de la réPLICATION serveur)
- Identifiant automatique sur 4 octets
- Aucun

Dans la zone Nom, vous allez indiquer le nom suivant : Mouvement. Remarquez les champs suivants qui se remplissent automatiquement. Vérifiez que la case « Le fichier possède un identifiant automatique » soit cochée. WinDev va ainsi créer un identifiant automatique. L'identifiant automatique est comparable à un compteur, c'est lui qui vous garantit l'unicité de vos enregistrements.

Cliquez sur Suivant.



La nouvelle fenêtre vous demande de confirmer le fait que vous voulez travailler avec des fichiers de type Hyper file (Format propriétaire WinDev).

C'est terminé !

Vous avez décrit les informations nécessaires à la création du fichier de données.

Cliquez sur le bouton de validation pour créer le fichier dans l'analyse.

Et voilà, la description du type de fichier est finie. Cliquez sur le bouton vert pour terminer. L'assistant se ferme et vous ouvre la fiche de description du fichier.

C'est ici que nous allons déterminer la composition de chaque attribut de notre fichier mouvement.

Vous pouvez remarquer que notre identifiant est déjà créé « Idmouvement » et vous voyez la clé jaune à gauche qui symbolise l'identifiant. Nous allons insérer les rubriques suivantes :

Nom	Libellé	Type	Taille
Date	Date de l'opération	Date	Sera rempli automatiquement
Descriptif	Description de l'opération	Texte	50
Dépense	Montant Débit	Réel (Format 99 999,99)	Sera rempli automatiquement
Recette	Montant Crédit	Idem	Idem

Pour insérer, cliquez sur une ligne vide dans la rubrique choisie. A vous de bien remplir vos champs, cela ne devrait vous poser aucun problème. Une fois la saisie terminée, cliquez sur Valider.

L'assistant vous propose de retourner sous l'éditeur de fenêtre. Cliquez sur Fermer pour rester sous l'analyse.

Double-cliquez sur la table pour faire apparaître la description, vous devriez avoir le résultat suivant :

The screenshot shows the 'Mouvement' table definition window. At the top, there's a header bar with the title 'Mouvement'. Below it, a toolbar includes icons for saving, opening, creating, deleting, and generating. The main area is a table with columns: Clé (Key), Nom (Name), Libellé (Label), Type (Type), and Taille (Size). The table contains five rows corresponding to the fields defined in the first table. The 'IDMouvement' column is marked as the primary key (indicated by a key icon). The 'Nom' column contains labels like 'Date', 'Descriptif', 'Dépense', and 'Recette'. The 'Libellé' column provides a more descriptive text for each field. The 'Type' column indicates the data type (Date, Texte, Numérique) and the 'Taille' column specifies the size (8, 50, 4). On the right side of the table, there are buttons for adding (+), deleting (-), and generating (a flag icon).

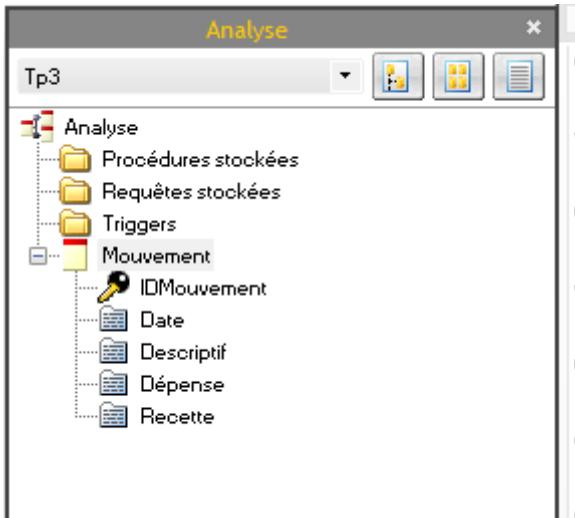
Clé	Nom	Libellé	Type	Taille
IDMouvement	Identifiant de Mouvement	Id. automatique	8	
	Date	Date	8	
	Descriptif	Description de l'opération	Texte	50
	Dépense	Montant Débit	Numérique	4
	Recette	Montant Crédit	Numérique	4

Pour que cette description de fichier soit prise en compte dans le projet nous devons générer l'analyse. Pour cela cliquez sur le drapeau

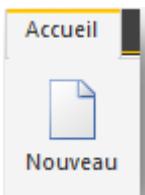


Remarques : Ce drapeau sert à lancer la génération de l'analyse. En effet, chaque fois que vous créez ou modifiez une structure de fichier, vous devez l'activer pour synchroniser l'analyse et les champs des fenêtres qui accèdent aux données.

Comme vous pouvez le constater dans le volet droit de l'Analyse nous retrouvons notre fichier:



Nous allons maintenant fabriquer les fenêtres de notre application. Cliquez sur l'icône Nouveau



Cliquez sur Fenêtre. Choisissez une fenêtre vierge.

Cette fenêtre sera la première fenêtre de notre application. Dans sa description (clic droit sur la fenêtre), vous lui donnerez les caractéristiques suivantes :

Nom logique : **départ**

Description : **Fenêtre principale de l'application**

Titre : **Bienvenue dans votre mini compte bancaire**

Enregistrez les modifications.

Nous allons insérer dans cette fenêtre une table mémoire qui sera le conteneur du fichier. Choisissez un champ table dans la barre d'outils et positionnez-le sur votre fenêtre. Dites à l'assistant que vous remplirez la table par programmation. Comme vous pouvez le constater, nous aurions pu prendre le choix par défaut (Afficher des données d'un fichier ou d'une requête existante). Ce choix par défaut paraît sensé, mais je préfère vous expliquer comment faire les choses manuellement. Vous aurez tout le temps ensuite de le faire de façon automatique.

Cours d'Ateliers de Génie Logiciel

Cet assistant va vous aider à créer un champ table.

Comment souhaitez-vous remplir le champ table ?

The screenshot shows a step in a wizard titled "Remplir le champ table par programmation". It includes three options: "Afficher des données d'un fichier ou d'une requête existante" (with a folder icon), "Définir une requête maintenant" (with a document icon), and "Afficher des données d'une variable existante" (with a blue X icon). The first option is selected.

Cliquez sur Suivant.

Paramètres supplémentaires

This step contains two sections: "Mode de saisie" and "Bandeau de sélection".
In "Mode de saisie", the radio button "Table en affichage" is selected.
In "Bandeau de sélection", dropdown menus show "Sélection simple" for Lines, "Sans sélection" for Columns and Cells.

Choisissez Type de « Table en affichage » puis Suivant.

Orientation du champ table

This step shows two radio button options: "Vertical" (selected) and "Horizontal". Below each is a preview of a 2x5 grid table. The vertical preview shows five columns labeled "Colonne 1" through "Colonne 5" and two rows labeled "Ligne 1" and "Ligne 2". The horizontal preview shows two rows labeled "Ligne 1" and "Ligne 2" and five columns labeled "Colonne 1" through "Colonne 5".

Nous allons choisir un type de présentation des données vertical. Cliquez sur Suivant.

Définition des colonnes

This step allows defining 5 columns. The first column is selected and has its title set to "Colonne1", type to "Texte", and width to "100 pixels". Other columns are labeled "Colonne2" through "Colonne5". A note says "Cliquez sur la colonne pour modifier ses caractéristiques." (Click on the column to modify its characteristics).

Cours d'Ateliers de Génie Logiciel

Cet écran nous permet de prédéfinir le nombre de colonnes de la table. Indiquez qu'il y aura 5 colonnes. Cliquez ensuite sur Suivant.

C'est terminé !

Nom du champ _____
Il vous reste à donner un nom au champ table.
Ce nom sera utilisé pour manipuler le champ dans le code WLanguage.

Nom : Tmouv

Libellé du champ _____
Français (5)
Table



Donnez « Tmouv » comme nom de la table. Et cliquez sur le bouton Terminer. La table est maintenant prédéfinie. Vous devriez voir ceci :



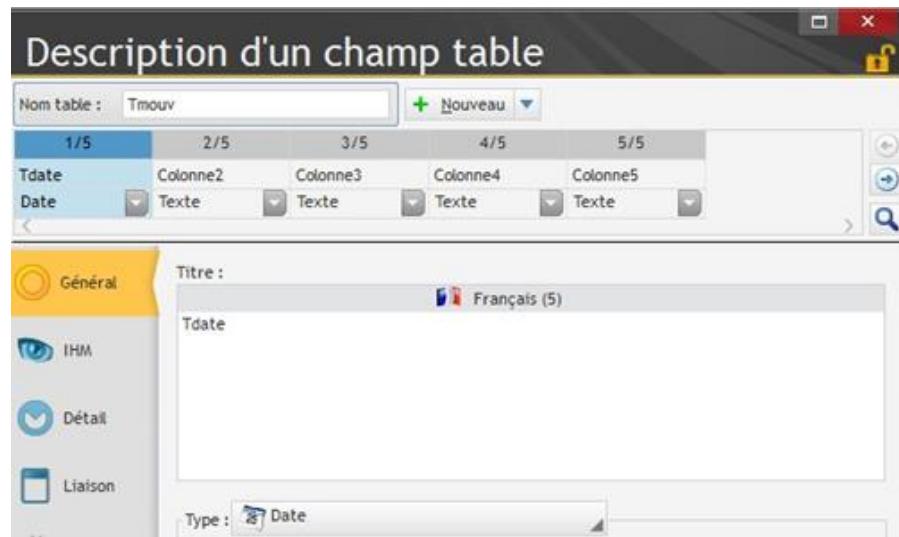
Comme vous le voyez le nom des colonnes, le type des données n'est pas défini, faites un clic droit sur la table et activez le menu Description. Vous allez suivre les instructions suivantes pour tout modifier :

Nom de la colonne 1 : Tdate

Type de la colonne 1 : Date

Titre de la colonne 1 : Date

Voici ce que vous devriez voir à ce stade :



Cours d'Ateliers de Génie Logiciel

Nom de la colonne 2 : Tdescription

Type de la colonne 2 : Texte

Titre de la colonne 2 : Description de l'opération

Taille de saisie : 50

Nom de la colonne 3 : Tdébit

Type de la colonne 3 : Numérique (Réel ayant un masque identique au fichier +99 999,99)

Titre de la colonne 3 : Dépense

Nom de la colonne 4 : Tcrédit

Type de la colonne 4 : Numérique (Réel ayant un masque identique au fichier +99 999,99)

Titre de la colonne 4 : Recette

Nom de la colonne 5 : clé

Type de la colonne 5 : Numérique

Masque de saisie : 999 999 999.

Dans l'onglet IHM de cette 5^{ème} colonne, décochez Visible, nous mettrons dans ce champ l'identifiant de la ligne. Il n'est pas nécessaire de le montrer à l'utilisateur, c'est pour ça que je vous conseille de le mettre invisible.

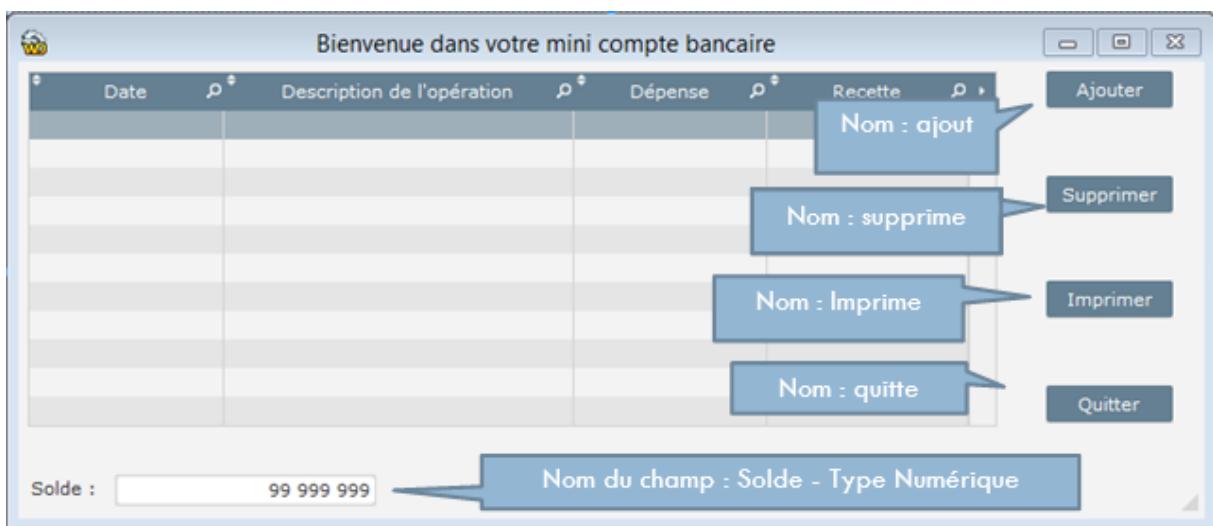


Une fois tous les champs renseignés, cliquez sur Appliquer puis OK

Vous devriez voir ceci :



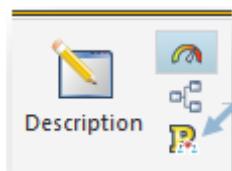
Vous pouvez redimensionner les largeurs des colonnes avec la souris si certains libellés sont tronqués. Voici maintenant, ce qu'il vous reste à concevoir pour terminer notre projet :



Il semble que ça fait longtemps que vous n'avez pas sauvegardé votre projet !

Maintenant que le décor est planté, nous pouvons commencer la programmation.

La premier chose à faire est de dire à WinDev de nous créer le fichier Mouvement s'il n'existe pas, pour cela allez dans le menu Projet / Code du projet.



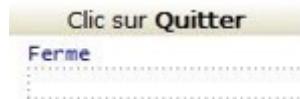
Dans la zone Initialisation de TP3 inscrivez le code suivant :

```
Initialisation de Tp3  
HCréationSiInexistant(Mouvement)
```

Cours d'Ateliers de Génie Logiciel

Remarque : Cette ligne indique à WinDev de commencer à chercher si le fichier Mouvement existe, s'il ne le trouve pas il le conçoit. Le code placé dans cette zone est exécuté avant le chargement de la première fenêtre.

Le code du bouton Quitter est très facile : Dans clic sur Quitter inscrivez :



Voyons maintenant la décomposition possible des événements. Il faut qu'au chargement de la fenêtre la table se remplisse avec les enregistrements contenus dans le fichier situé sur le disque dur. Pour cela nous allons parcourir l'ensemble des lignes du fichier « Mouvement.fic » et les placer les unes après les autres dans la table mémoire. C'est ce que nous allons faire maintenant.

Allez dans le code de la fenêtre dans la zone « Initialisation de départ » et saisissez le code suivant :

```
Déclarations globales de Départ "
PROCEDURE Depart()
POUR TOUT Mouvement
    TableAjouteLigne(Tmouv,Mouvement.Date,Mouvement.Descriptif,Mouvement.Dépense,Mouvement.Recette,Mouvement.IDMouvement)
FIN
```

Explication du code :

Pour tout Mouvement // *Cette ligne met en place une boucle qui ordonne à WinDev d'ouvrir le fichier Mouvement et de lire la première ligne en plaçant les champs correspondants en mémoire.*

La séquence tableajouteligne est connue, elle permet de positionner du texte - ici les rubriques du fichier - à l'intérieur de la table mémoire.

Fin // indique la fin de la boucle.

En résumé, nous pourrions lire ce code de la façon suivante :

Pour toutes les lignes du fichier « **Mouvement** »,
Ajoute les enregistrements physiques dans la table « **Tmouv** »

Le bouton Ajouter va ouvrir une fenêtre de saisie qui nous permettra de rentrer les informations. Donc le code sera dans Clic sur ajout :



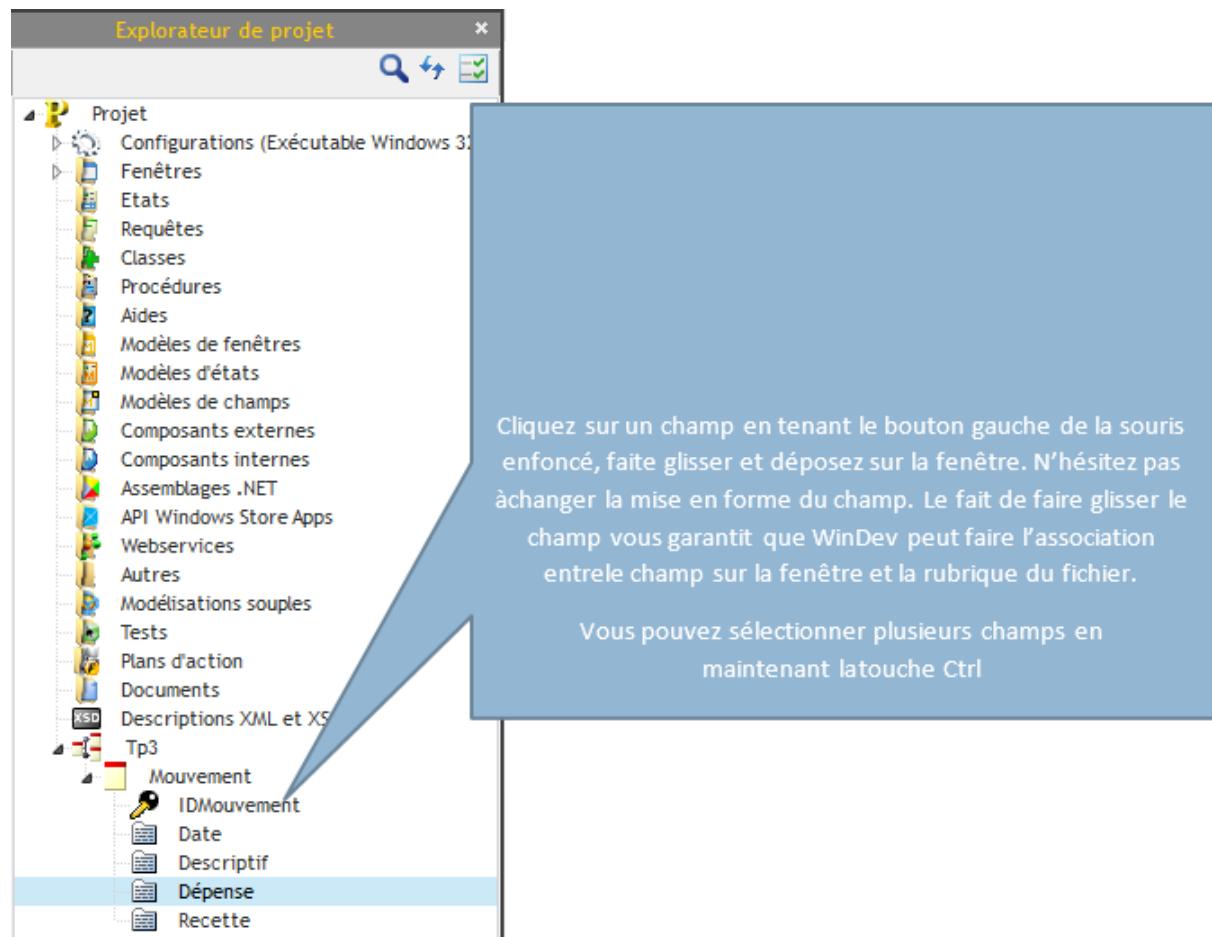
Il est normal que WinDev vous renvoie un message d'erreur si nous essayons d'exécuter le projet car la fenêtre « FSaisie » n'existe pas, créons-la de suite.

Dans la description de la fenêtre :

Nom logique : Fsaisie
Description : Fenêtre de saisie
Titre : Saisissez votre opération

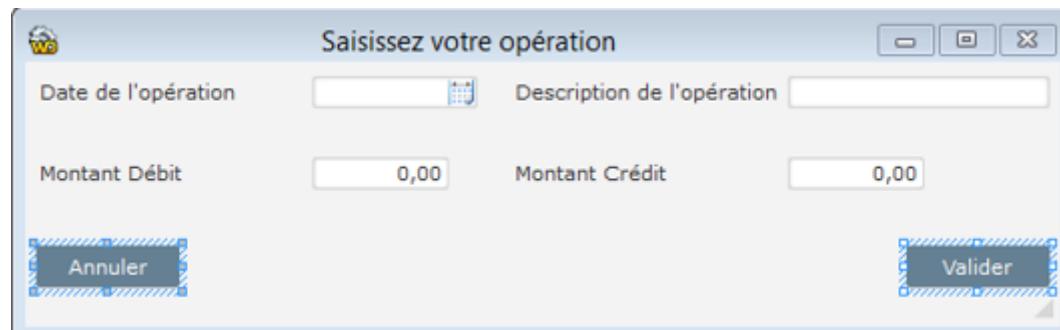
Il me semble que ça fait longtemps que vous n'avez pas sauvegardé votre projet !

Nous allons placer les champs nécessaires sur la fenêtre « Fsaisie »



Faites glisser les champs Date, Descriptif, Dépense, Recette sur la fenêtre Fsaisie.

Voici le résultat :



Cours d'Ateliers de Génie Logiciel

Pour être sûr que chaque champ est bien lié à une rubrique du fichier, cliquez sur l'un d'entre eux et vérifiez dans la barre de message en bas à gauche que le message « Lié à : Mouvement.XXXXXX » soit présent.



Cette fenêtre ne comporte que 2 boutons, le bouton Annuler nous servira juste à fermer la fenêtre. Vous connaissez la séquence de code le permettant. En cas de problème je vous rappelle que l'aide de WinDev est accessible par la touche F1.

Consacrons-nous au bouton Valider, la validation consiste à placer les rubriques de la fenêtre dans le fichier et à valider l'ajout. Pour placer les rubriques de la fenêtre dans le fichier, l'ordre est le suivant :

```
EcranVersFichier(Fsaisie) // Fsaisie étant le nom de la fenêtre
```

La validation d'ajout est commandée par l'ordre suivant :

```
HAjoute(Mouvement) // Mouvement étant le fichier dans lequel on ajoute
```

Il nous reste plus qu'à ajouter un ordre de fermeture de la fenêtre.

Voici le code intégral du bouton Valider :

```
Clic sur Valider
EcranVersFichier(Fsaisie)
HAjoute(Mouvement)
Ferme
```

Vous pouvez faire en sorte de programmer le contrôle pour qu'il vérifie que l'on n'a qu'un débit ou qu'un crédit, ou alors que l'un des champs n'est pas vide. Le code à placer juste avant celui que l'on voit ci-dessus pourrait ressembler à ceci :

```
Clic sur Valider
SI Date="" OU Descriptif="" OU (Dépense=0 ET Recette=0) ALORS
    Info("L'un des champs n'a pas été rempli !")
    RETOUR
FIN
SI Dépense <> 0 ET Recette<>0 ALORS
    Info("Vous ne pouvez saisir qu'un montant au débit ou alors au crédit !")
    RETOUR
FIN
EcranVersFichier(Fsaisie)
HAjoute(Mouvement)
Ferme
```

Testez votre fenêtre avec le bouton  ou en cliquant à droite de l'écran sur la fenêtre avec le bouton droit pour choisir Tester.

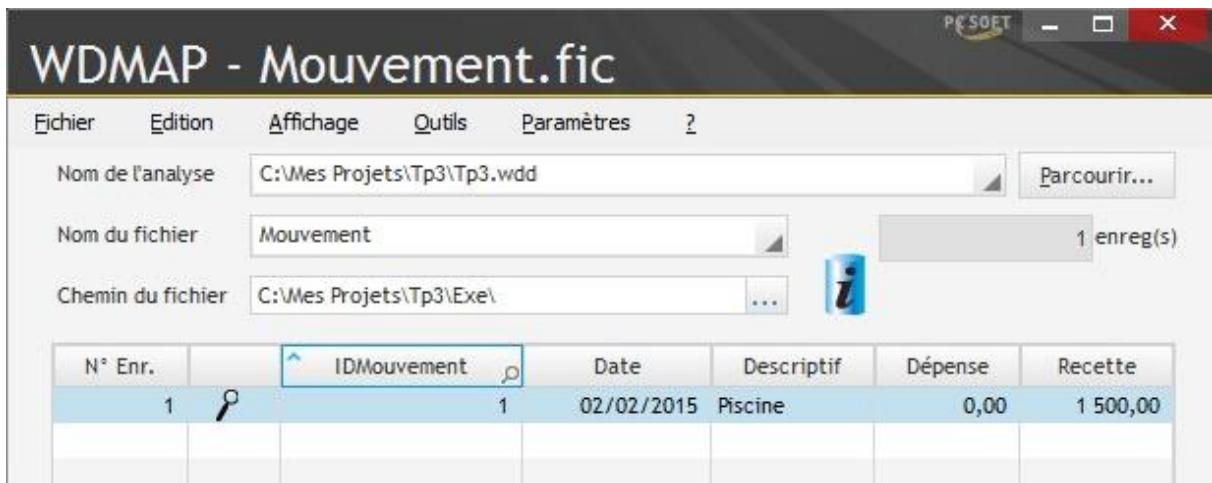
Cours d'Ateliers de Génie Logiciel

Insérez des valeurs dans les champs et validez.

Pour voir si votre nouvelle ligne est présente dans le fichier, allez dans le menu Outils / WDMAP



Choisissez Mouvement comme Nom du fichier.



Maintenant que vos lignes s'insèrent, lancez le projet en cliquant sur GO .

Cliquez sur le bouton Ajouter, saisissez et validez un nouvel enregistrement. Vous pouvez constater que la table mémoire ne réagit pas correctement : en effet l'insertion n'a pas été détectée et donc la table mémoire n'est pas synchronisée avec le fichier. Nous allons essayer de remédier à ce problème. En fait, il faudrait que lorsque la fenêtre saisie se ferme, la fenêtre départ recharge la table mémoire.

Placez-vous dans le code de la fenêtre départ, vous devez trouver une zone nommée « prise de focus de départ ». La prise de focus est le fait de remettre active une fenêtre inactive, en cliquant sur la barre de titre par exemple.

Voici le code à insérer dans cette zone :

```
Prise de focus de départ
TableSupprimeTout(Tmouv) // Efface la table mémoire pour éviter d'insérer les enregistrements à la suite des précédents
POUR TOUT Mouvement
    TableAjouteLigne(Tmouv,Mouvement.Date,Mouvement.Descriptif,Mouvement.Dépense,Mouvement.Recette,Mouvement.IDMouvement)
FIN
```

Testez cette modification, comme vous le voyez, les comportements sont maintenant cohérents.

Cours d'Ateliers de Génie Logiciel

Intéressons-nous au bouton Supprimer. Dans la table, nous avons une rubrique qui est l'identifiant de la ligne. Pour supprimer cette ligne dans le fichier nous allons donc rechercher cet identifiant dans le fichier et supprimer la ligne correspondante.

Voici la séquence de code nécessaire :

```
Clic sur Supprime
HLitRecherchePremier(Mouvement, IDMouvement, Tmouv. Clé)
SI HTrouve(Mouvement) ALORS
    HSupprime(Mouvement)
    Info("La suppression est effective")
SINON
    Info("Grave problème de l'application")
FIN
```

Explications :

La première ligne fait rechercher à l'identifiant dans le fichier Mouvement et sur l'identifiant la valeur du champ clé pointée dans notre table mémoire. Ce n'est pas parce que la rubrique clé est invisible que nous ne pouvons pas en connaître la valeur.

Si on trouve la ligne ayant le même identifiant que clé alors on la supprime et on affiche un message indiquant la bonne marche des opérations. Sinon dans un cas fort improbable où il ne trouve pas l'enregistrement on inscrit un message d'erreur.

Je vous laisse le soin de tester cette nouvelle fonctionnalité de votre programme. Comme vous venez de le remarquer, la mise à jour de la table mémoire ne s'est pas faite. La valeur a été supprimée mais la table ne le sait pas. Nous allons remédier à ce problème.

Le code de réaffichage de la table mémoire existe déjà (ex : dans la zone de prise de focus de la fenêtre départ) nous allons donc ré-exécuter un traitement existant. Sous la ligne « Info("La suppression est effective") » inscrivez la commande suivante :

```
ExécuteTraitement(depart,trtPriseFocus)
```

Cette commande fait rejouer un traitement existant.

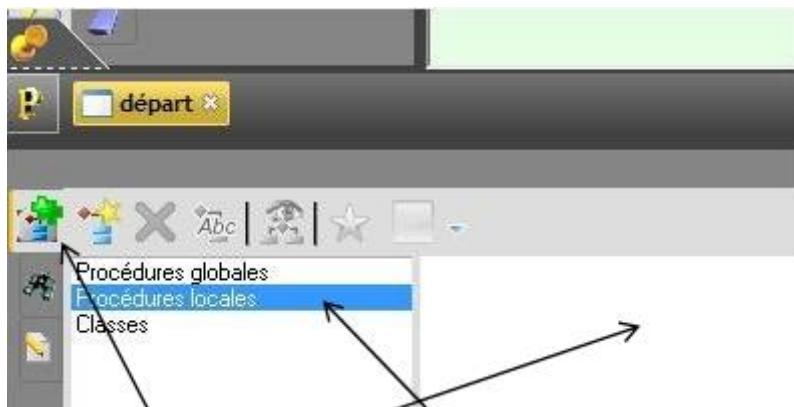
Testez et vérifiez la cohérence de votre projet.

LES PROCÉDURES

Maintenant nous avons à calculer le solde (Débit – Crédit), pour ce faire nous allons créer une procédure qui scannera le fichier et fera les calculs pour nous.

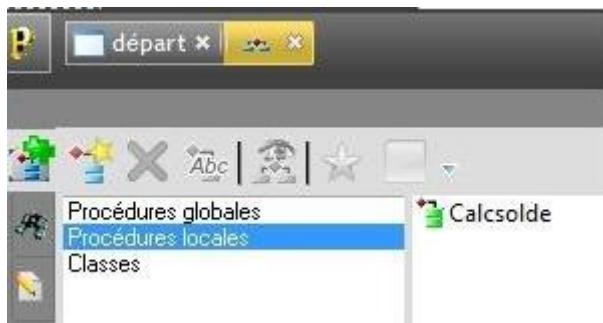
Pour créer une procédure

Cours d'Ateliers de Génie Logiciel



Recherchez en bas de l'écran cet onglet, Choisissez « Procédures locales », et faites un clic droit dans cette zone. Dans le menu contextuel choisissez « Nouvelle procédure locale ».

Nommez cette procédure Calcsolde.



Remarques : Les procédures locales ne sont vues que par les objets de la fenêtre, les procédures globales sont actives pour tous les éléments du projet

Maintenant, vérifiez bien que vous vous trouvez dans la zone code de la procédure calcsolde. (Clic droit et ensuite l'option « Code »).

Le code doit parcourir le fichier Mouvement, affecter le contenu de débit dans une variable, le contenu de crédit dans une autre et cela jusqu'à la fin du fichier et ensuite affecter la différence entre le débit et le crédit au champ solde.

Voici le code de la procédure :

```
* Procédure locale Calcsolde *
PROCEDURE Calcsolde()
    sdebit,scredit sont des réels=0 // on affecte la valeur 0 aux deux variables
POUR TOUT Mouvement
    sdebit+=Mouvement.Dépense // += signifie Sdebit=Sdebit+Mouvement.Dépense
    scredit+=Mouvement.Recette
FIN
Solde=sdebit-scredit // Solde représente le champ de la fenêtre
```

Cours d'Ateliers de Génie Logiciel

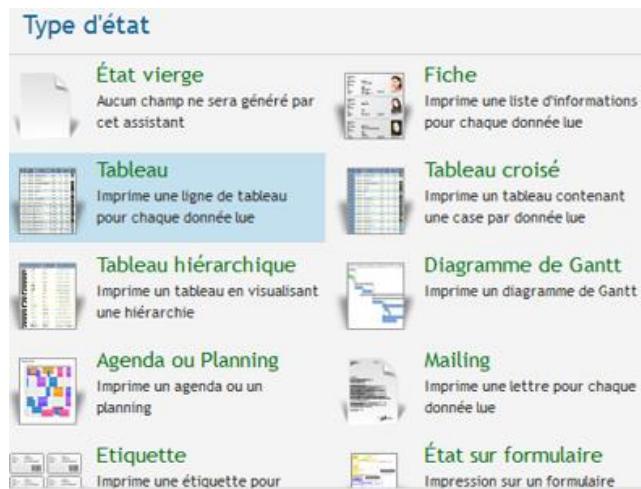
Le code est suffisamment simple pour ne pas avoir à l'expliquer. La question qui se pose est où lancer calcsolde. Les plus rusés d'entre vous auront compris qu'il nous faut activer cette procédure aux mêmes endroits où l'on a activé le rafraîchissement de la table mémoire. Je vous laisse modifier les zones de code en conséquence (dans le code de la fenêtre départ). N'oubliez pas de tester la cohérence de votre projet.

```
Initialisation de départ *
Si Erreur : par programme Quand Exception : par
Pour TOUT Mouvement
    TableAjouteLigne(Tmouv,Mouvement.Date,Mouvement.Descriptif,Mouvement.Dépense,Mouvement.Recette,Mouvement.IDMouvement)
FIN
Calcsolde()

Fermeture de départ *
Si Erreur : par programme Quand Exception : par
Prise de focus de départ *
Si Erreur : par programme Quand Exception : par
TableSupprimeTout(Tmouv) // Efface la table mémoire pour éviter d'insérer les enregistrements à la suite des précédents
POUR TOUT Mouvement
    TableAjouteLigne(Tmouv,Mouvement.Date,Mouvement.Descriptif,Mouvement.Dépense,Mouvement.Recette,Mouvement.IDMouvement)
FIN
Calcsolde()
```

LES IMPRESSIONS

Il ne nous reste plus qu'à fabriquer l'état de sortie. Choisissez Nouveau / Etat. Choisissez un état de type Tableau.



En cliquant sur Suivant WinDev vous demande la source de donnée, il vous faut préciser que ce sont des enregistrements provenant d'un fichier Hyper file.



Sélectionnez le fichier Mouvement.

Fichier de données

Sélectionnez le fichier ou la requête qui servira de source de données de l'état.

Fichier / Requête	Libellé
Mouvement	Mouvement

La clé de parcours est un identifiant qui sert pour donner l'ordre de tri.

Paramètres de parcours du fichier

Clé de parcours _____

Les rubriques de type clés organisent les fichiers de données.
La clé de parcours donne l'ordre de lecture du fichier de données.

Choisissez une clé de parcours :

Parcours borné _____

Les bornes permettent d'imprimer uniquement les données pour lesquelles la valeur de la clé de parcours est comprise entre les bornes spécifiées.

Les valeurs des bornes sont fournies à l'état à l'aide de la fonction IlprimeEtat.
Ex : IlprimeEtat(MonEtat,"2000", "2002")

Valeur minimale fournie
 Valeur maximale fournie

Remarques : Les détails du choix du style, de la mise en forme sont à votre discréption, faites comme bon vous semble. N'oubliez pas que le client n'a pas les mêmes goûts graphiques que vous. Eviter les styles « **Noir sur fond Noir** » « **Rose sur fond Vert** » et autres singularités visuelles qui feront penser à l'utilisateur qu'il devient déficient visuel.

Une fois votre état fini et enregistré, entrez dans la zone code du bouton Imprime pour appeler l'état Etatmouv :

Clic sur Imprimer *

```
iAperçu(i100) // On enverra l'état à l'écran avec un zoom de 100 %
iImprimeEtat(EtatMouv) // Génération de l'état et utilisation des paramètres définis par iAperçu
```

Si Erreur : par programm

Remarques : Les Commandes WinDev sont classées. Celles qui commencent par H sont des commandes d'accès aux fichiers, celles qui commencent par i sont des commandes de pilotage d'état.

Refaire ce TP plusieurs fois, le but étant de se passer du guide papier et d'apprécier la facilité avec laquelle on peut travailler avec WinDev.

EXERCICES APPLICATIFS

Répertoire d'amis

Enoncé :

Ecrire un programme gérant un fichier des amis. Pour chacun d'eux, on souhaite connaître le nom, le prénom et le numéro de Téléphone.

Faites en sorte que le programme puisse permettre à l'utilisateur de :

- ✓ Saisir un nouvel ami.
- ✓ Supprimer l'ami qui vous a fait une embrouille.
- ✓ Modifier le numéro de téléphone d'un ami.
- ✓ Imprimer la liste des amis.
- ✓ Afficher le nombre d'amis.
- ✓ Quitter.

La Vidéothèque

Enoncé :

Un proche vous demande de lui concevoir un programme d'archivage de ses cassettes vidéo. Il veut connaître le titre du film, l'année de sortie, l'acteur masculin principal, l'actrice féminine principale et le genre du film (comédie, dramatique, policier, western, enfants, adultes....) Il vous demande aussi des états imprimés triés soit par année de sortie, soit par genre, soit avec aucun tri.

III.3. Travaux Pratiques avec Visual Studio

Microsoft Visual Studio est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft en 1997. La dernière version s'appelle Visual Studio 2019.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Developer.

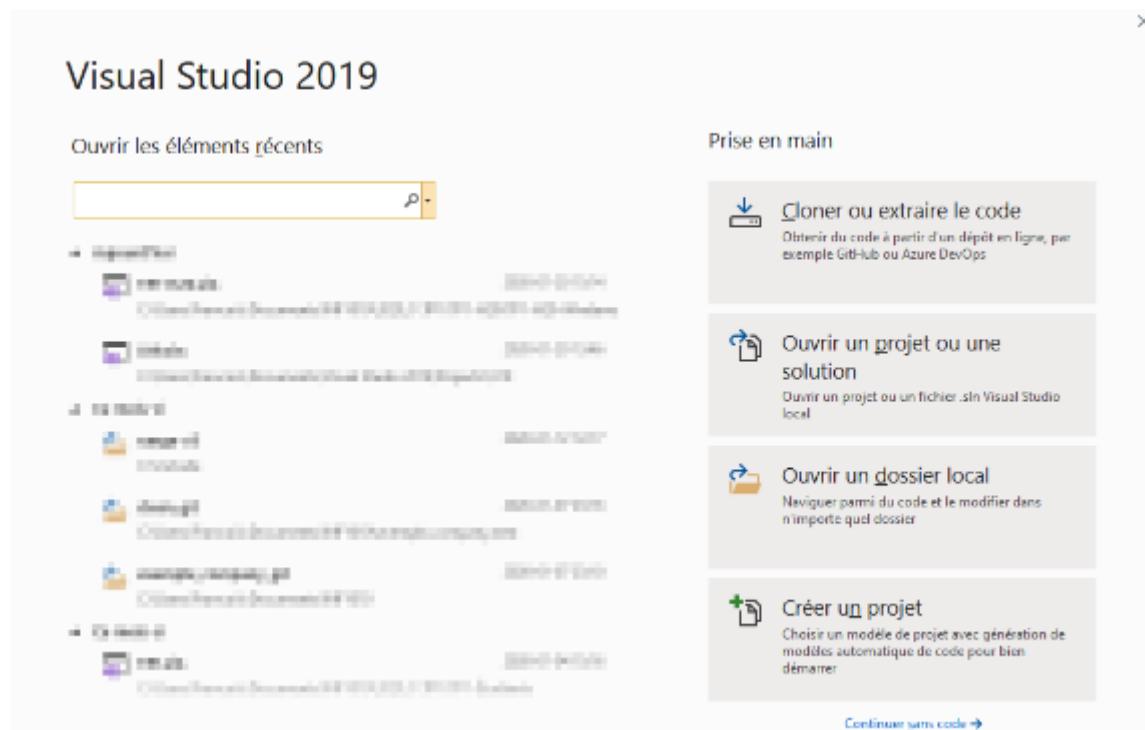
Ouverture de Visual Studio

Choisir « Visual Studio 2019 ». À sa sélection, le logiciel démarre. Lors de la première utilisation, il vous sera demandé de vous connecter, mais cliquez simplement sur « Me proposer ».

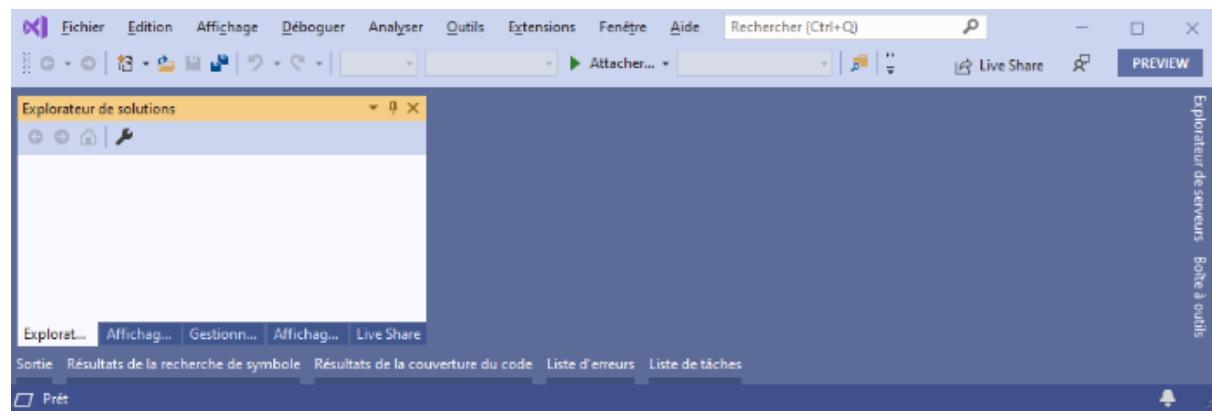
Cours d'Ateliers de Génie Logiciel

ultérieurement». Ensuite, vous est demandé de choisir un environnement de développement, et votre choix préféré de couleurs (sombres ou pâles) pour les fenêtres de l'environnement.

La fenêtre d'ouverture de projet s'affiche.

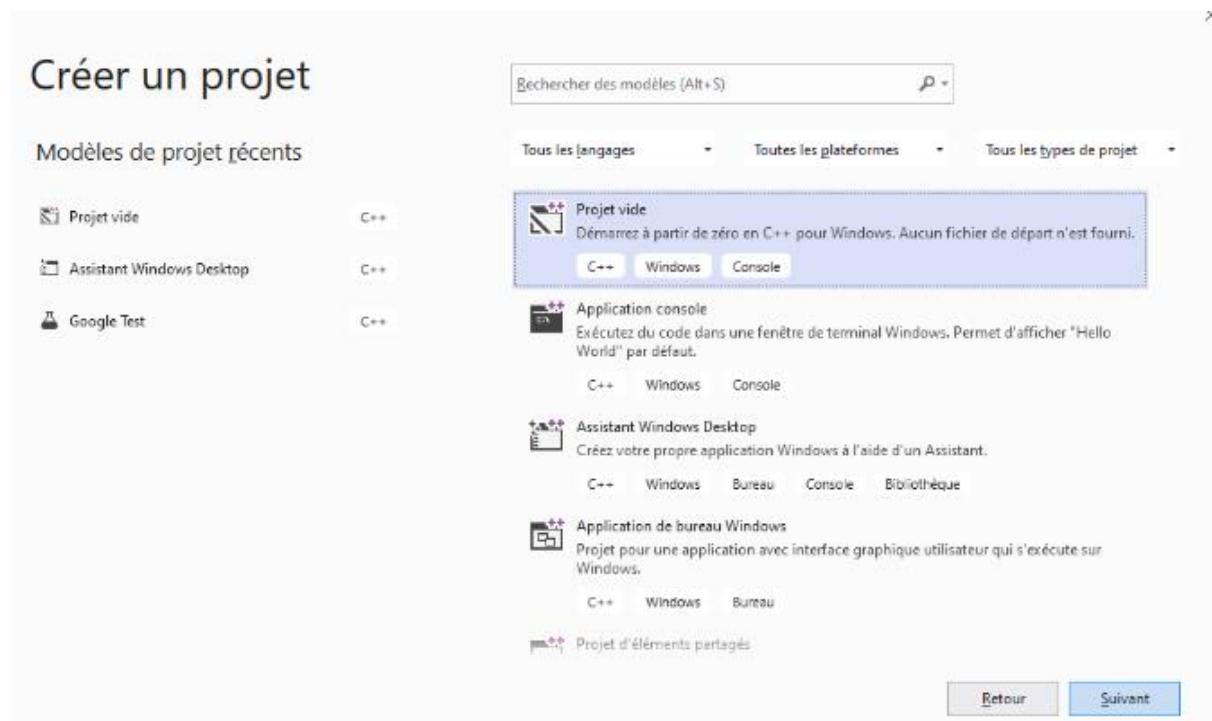


L'écran principal de Visual Studio est découpé en plusieurs zones. Pour chaque TD, vous devrez créer un projet. Un projet consiste en un ou plusieurs fichiers qui permettent de concevoir une application.

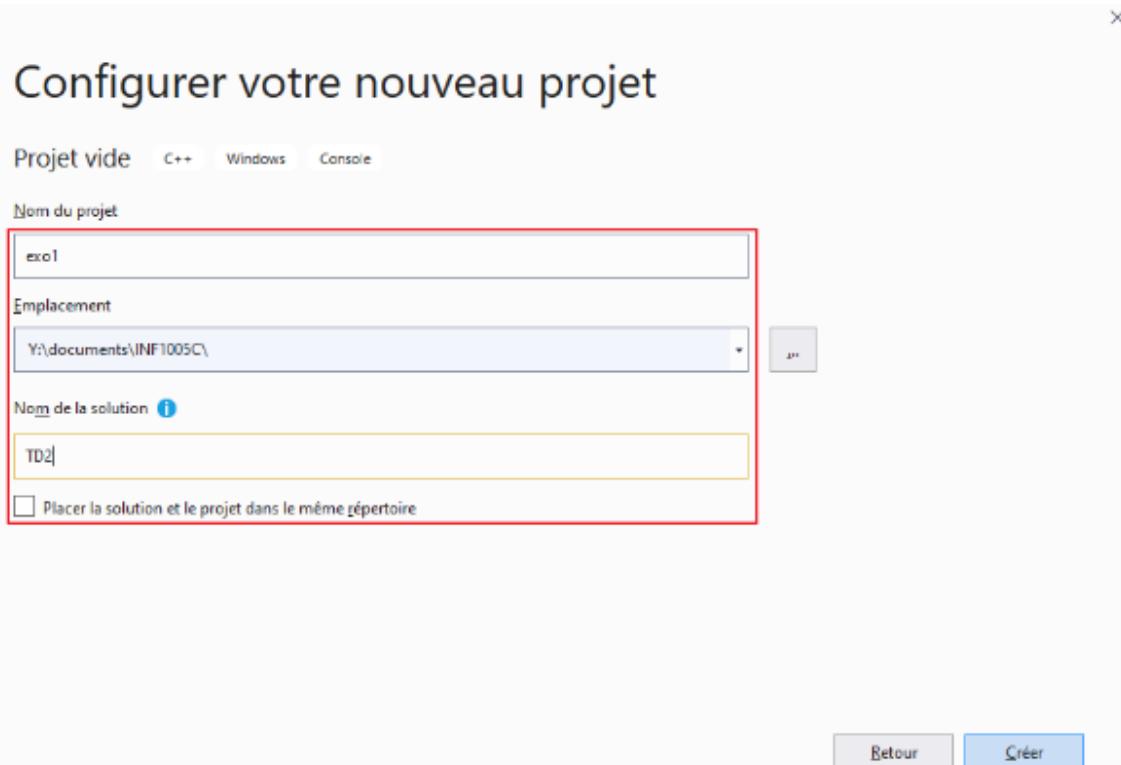


Création d'une solution contenant un projet

Pour commencer, cliquez Créeer un projet (ou le menu Fichier >Nouveau >Projet, de l'écran principal). Une fenêtre apparaît alors pour vous permettre de choisir le type de ce projet.



Tout d'abord, choisissez «Projet vide» («Empty project» en anglais). Une nouvelle fenêtre s'affiche. Précisez un nom pour le projet (par exemple «exo1»). Décochez la case «Placer la solution et le projet dans le même répertoire» et entrez un nom de solution (par exemple «TD2»). Dans Visual Studio, une «solution» peut contenir plusieurs «projets», et nous utiliseront un projet par exercice du TD. Choisissez ensuite un emplacement où sauvegarder votre travail, puis «Créer».

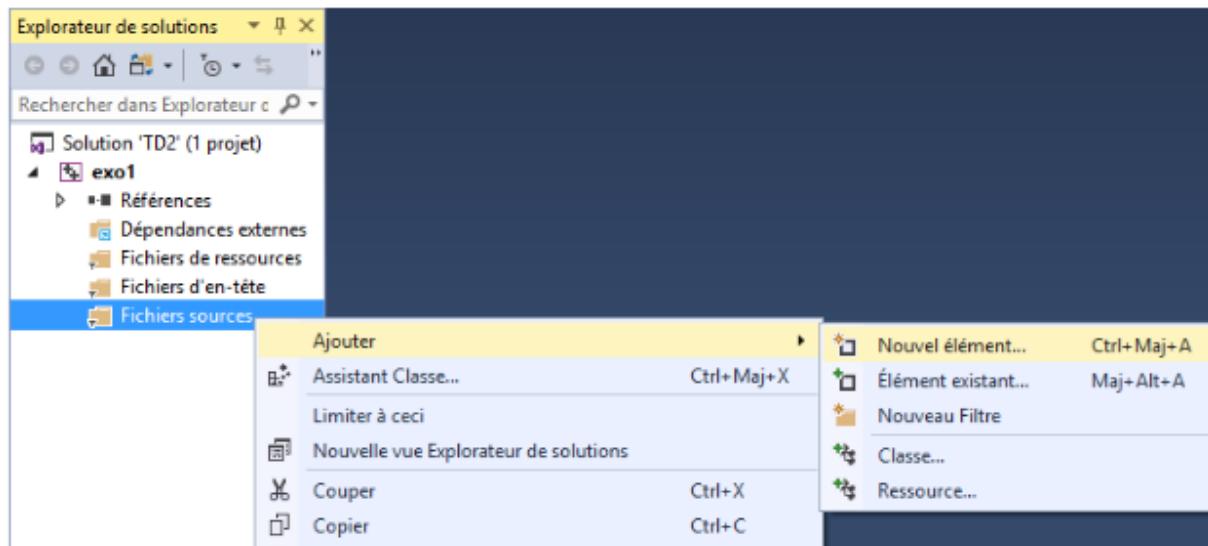


Cours d'Ateliers de Génie Logiciel

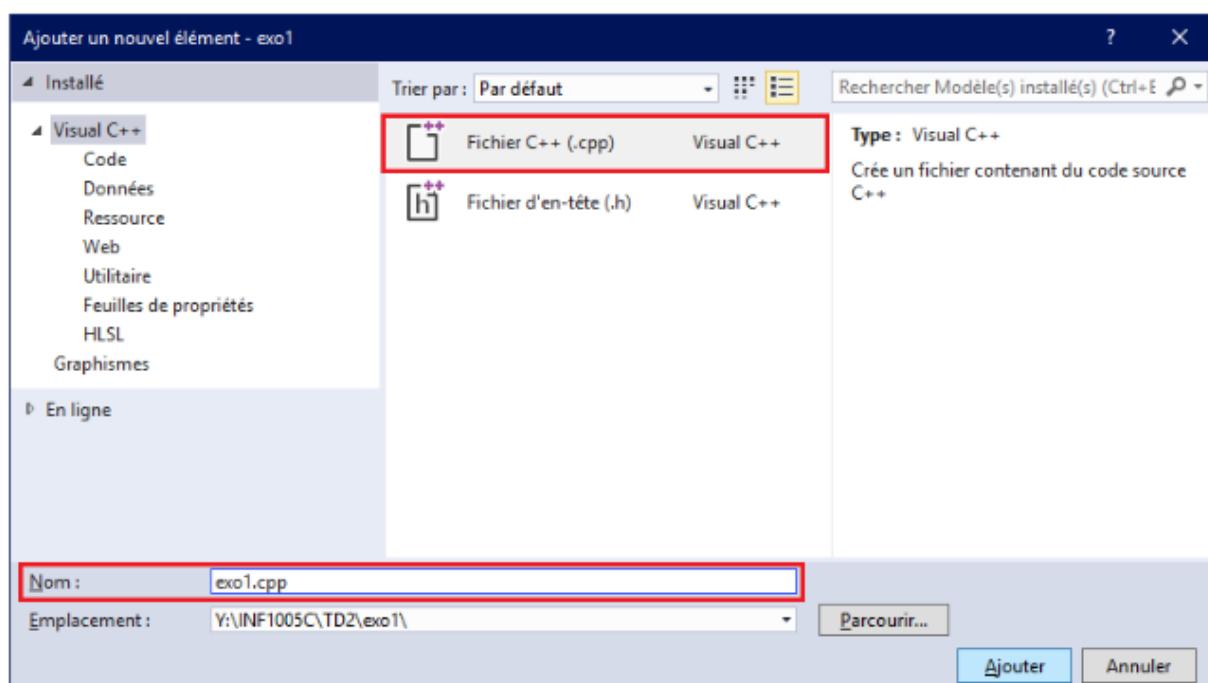
Votre projet est maintenant configuré! Il vous reste à ajouter un fichier source à celui-ci pour pouvoir y mettre le code de l'exercice.

Création d'un nouveau fichier dans le projet

Dans l'«Explorateur de solutions», faites un clic droit sur «Fichiers sources» qui se trouve quelque ligne sous votre projet (par exemple « exo1 »; il apparaît normalement en dessous de « Solution TD2 » dans la partie gauche de l'interface) et sélectionnez « Ajouter> Nouvel Élément ».



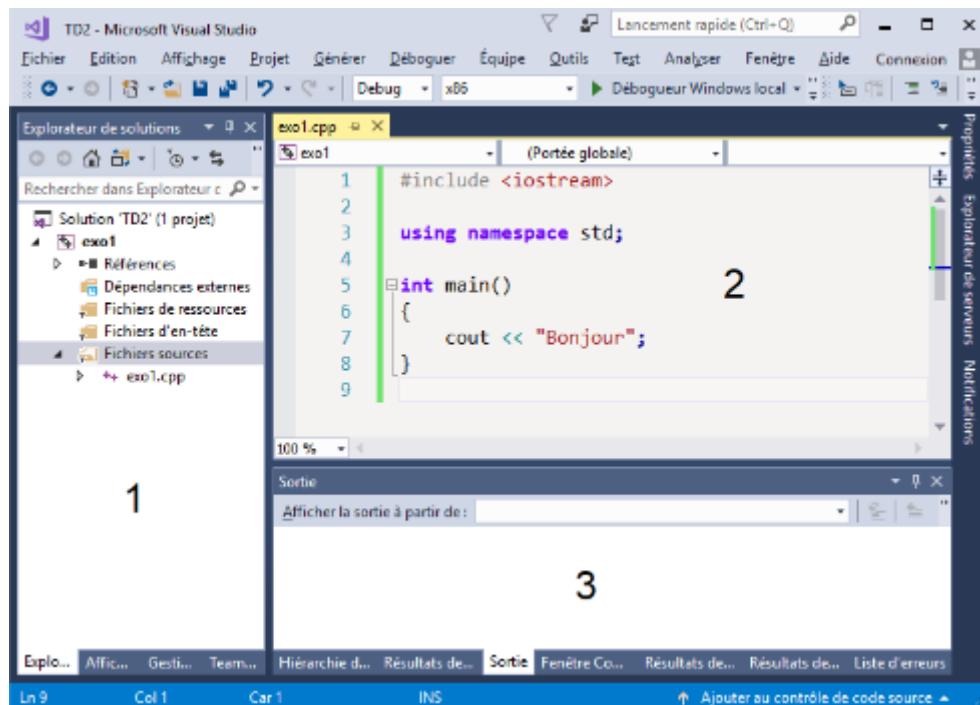
Vous aurez alors une fenêtre vous demandant le type de fichier à ajouter. Choisissez, entrez un nom (par exemple « exo1.cpp ») et ne changez pas l'emplacement (par défaut il le met dans votre projet). Cliquez ensuite sur «Ajouter».



Vous voilà maintenant prêts à programmer!

Zones de l'interface

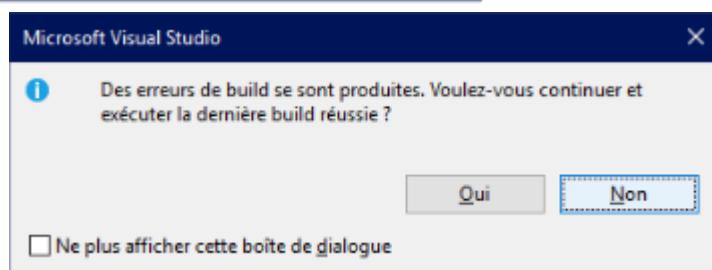
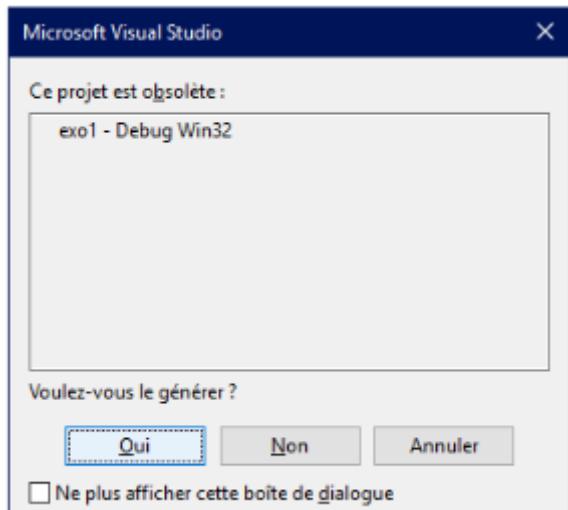
L'interface est divisée en plusieurs. Notez que l'interface est complètement configurable et que tout élément peut être déplacé comme vous le voulez; la figure montre la disposition par défaut si vous avez choisi le mode «C++» lors de l'ouverture initiale de Visual Studio. La zone 1 correspond à l'explorateur de solutions, c'est-à-dire à la liste des différents projets présents dans la solution courante (s'il n'est pas affiché, on peut l'ouvrir avec le menu Affichage > Explorateur de solutions). La zone 2 est celle qui nous intéresse le plus, c'est dans celle-ci que vous écrirez votre code! La zone 3 comprend des informations additionnelles sur la compilation ou le débogage de votre projet.



Pour tester vos applications, vous devez les compiler puis les exécuter. Vous pouvez lancer le processus en choisissant le menu «Déboguer> Exécuter sans débogage» (ou en pressant Ctrl+F5). Visual Studio vous indiquera que le projet est obsolète et vous demande si vous voulez le générer. Choisissez oui pour lancer la compilation.

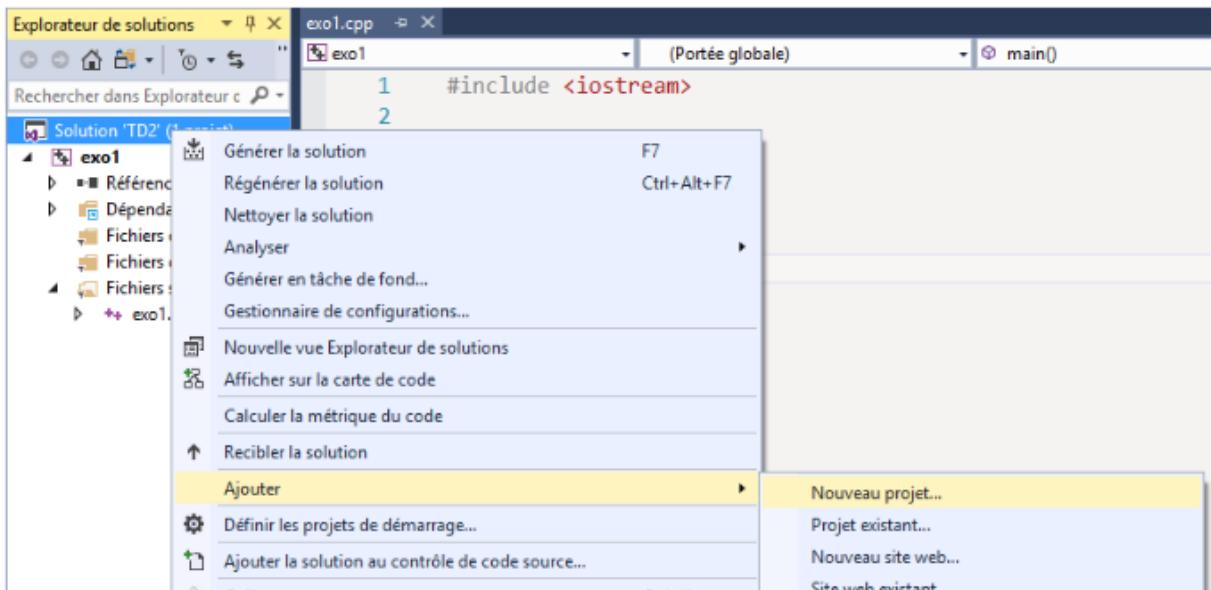
Si la compilation réussit, votre programme s'exécutera ensuite.

Si la compilation échoue, il indiquera que des erreurs de build se sont produites, et demandera si vous voulez «continuer et exécuter la dernière build réussie?», où il faut répondre «Non» et corriger votre programme avant de réessayer.



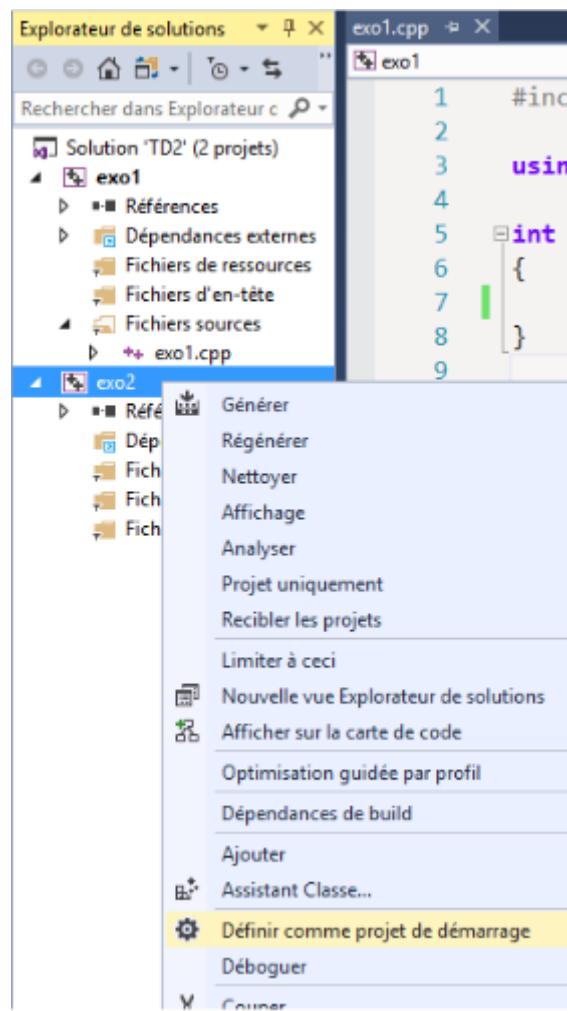
Ajout d'un projet

Faites un clic droit sur votre solution dans le panneau de gauche puis choisissez « Ajouter >Nouveau Projet ». Répétez alors les étapes de création de projets décrites plus haut. Notez que cette fois-ci, vous n'avez plus la possibilité de choisir le nom de la solution. Ceci est dû au fait que le nouveau projet est ajouté à la solution déjà créée. Laissez l'emplacement par défaut, qui devrait être celui de cette solution.



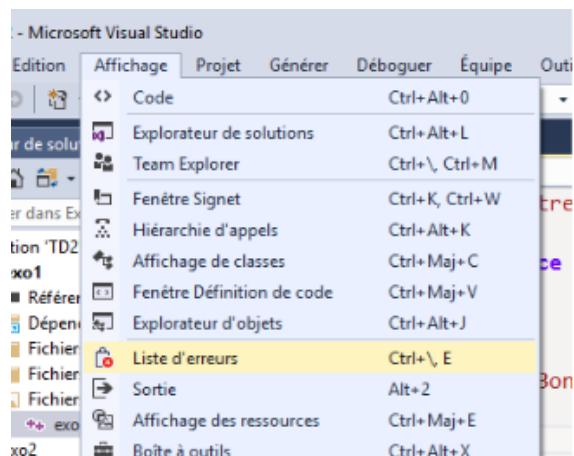
Cours d'Ateliers de Génie Logiciel

Lorsque vous voulez exécuter votre nouveau projet, il est bien important de faire un clic droit sur ce projet et de sélectionner «Définir comme projet de démarrage». Le projet qui s'exécute est toujours celui dont le nom est indiqué en gras.



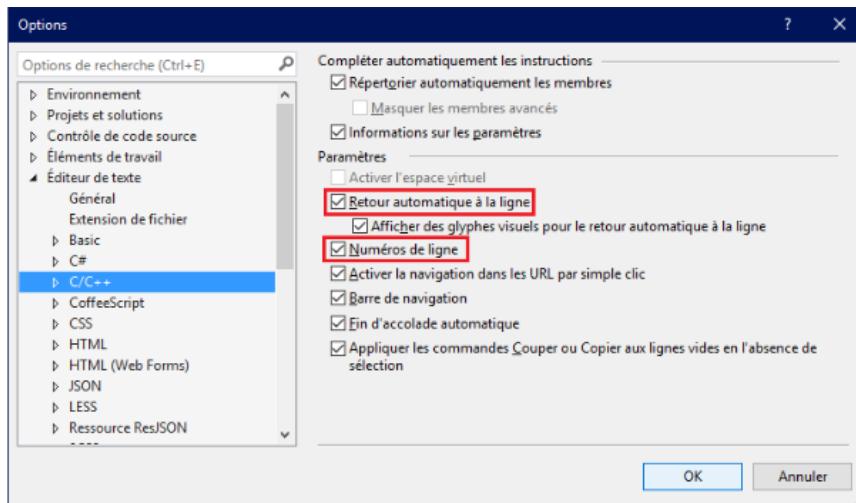
Afficher la liste d'erreurs

Choisissez le menu « Affichage >Liste d'erreurs» dans le menu en haut de l'interface. La liste d'erreurs s'affichera dans la zone 3.



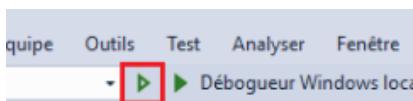
Afficher les numéros de lignes et activer le retour automatique à la ligne

Choisir Outils > Options. Dans le menu, Éditeur de texte, choisir C/C++ ou un autre langage selon le cas.



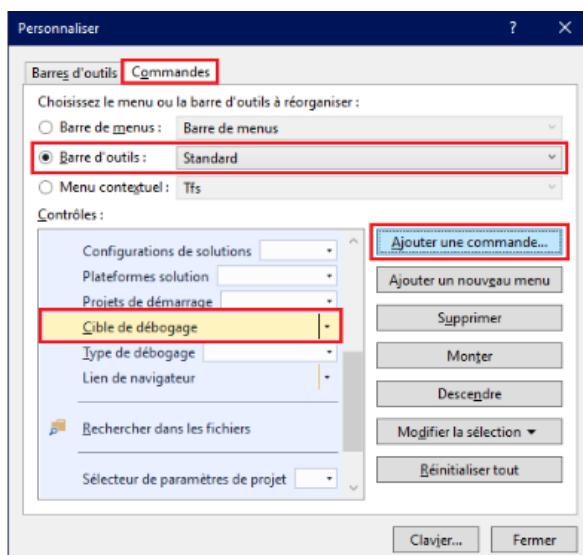
Ajouter un bouton dans la barre d'outils pour "Exécuter sans débogage"

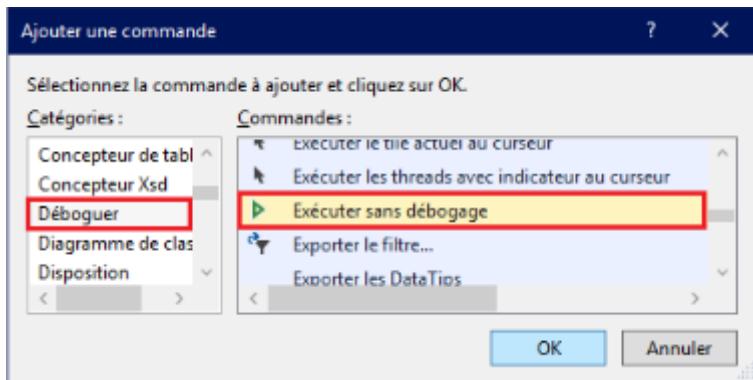
Nous cherchons à ajouter un bouton dans la barre d'outils pour faciliter l'exécution sans débogage.



Choisissez le menu Affichage > Barre d'outils > Personnaliser...

Dans l'onglet Commandes, choisissez la barre d'outils Standard. Sélectionnez sur Cible de débogage. Cliquez ensuite sur Ajouter une commande... Dans la catégorie Déboguer, choisissez Exécuter sans débogage et cliquez sur Ok. Le bouton est maintenant ajouté à la barre d'outils.





Exercices applicatifs

Convertisseur

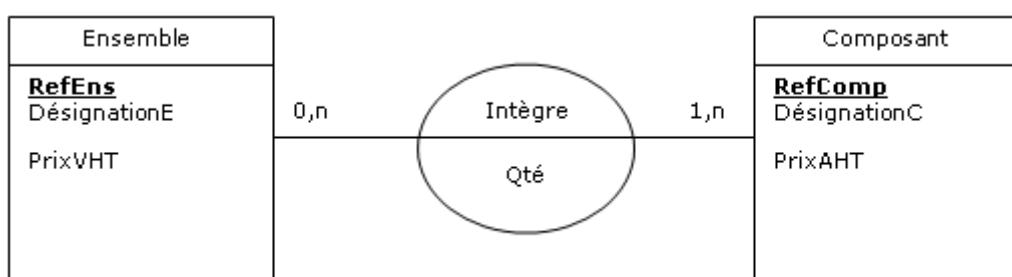
Reprendre le TP du convertisseur Franc / Euro précédemment traité avec WinDev.

Vidéothèque

Un proche vous demande de lui concevoir un programme d'archivage de ses cassettes vidéo. Il veut connaître le titre du film, l'année de sortie, l'acteur masculin principal, l'actrice féminine principale et le genre du film (comédie, dramatique, policier, western, enfants, adultes....) Il vous demande aussi des états imprimés triés soit par année de sortie, soit par genre, soit avec aucun tri.

Gestion de Production Assistée par Ordinateur

Nous allons créer une mini GPAO (Gestion de Production Assistée par Ordinateur). Vous travaillez pour un assembleur informatique, son processus de production est le suivant : Il reçoit les différentes pièces détachées (disque dur, mémoires, cartes mères...) et assemble ces différentes pièces pour en faire un modèle fini. Comme vous pouvez le percevoir, le modèle conceptuel travaillera avec 2 entités (Ensemble fini et composants). Voici une représentation du MCD :



Bibliographie

Maurizio Gabbrielli et Simone Martini, Programming Languages: Principles and Paradigms, Springer, 2010 [détail des éditions] (ISBN 9781848829138).

D. NANCI, B. ESPINASSE, B. COHEN, H. HECKENROTH, « Ingénierie des systèmes d'information avec Merise ; vers une deuxième génération », Sybex.

S. REYMANN, « DECF, MSTCF Informatique », Foucher.

G. LOUVET, « Se former à Merise », Editions d'Organisation.

H. TARDIEU, A. ROCHFELD, R. COLLETTI, « La méthode Merise, Tome I, Principes et outils », Editions d'Organisation.

Analyse de système orientée-objet et génie logiciel: Concepts, méthodes et application. G. LEVESQUE. Chenelière/Mc Graw-Hill.

Génie Logiciel, Jacques PRINTZ, Que Sais-Je, N° 2956. PUF.

Lynne Dunckley, Multimedia Databases, an object-relational approach, Pearson Education.

<http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/apex/r31/apex31nf/apex31blob.htm> : documentation Oracle pour Application Express.

SQL3 Object Model, <http://www.objs.com/x3h7/sql3.htm>.

Modélisation objet et SGBDOO, Madjid AYACHE et André FLORY, Approche Objet, Concepts et Utilisation, Economica.

G. GARDARIN, Bases de Données, Objet et Relationnel, Eyrolles.

Atelier de génie logiciel, https://fr.wikipedia.org/wiki/Atelier_de_g%C3%A9nie_logiciel.

WinDev, <https://fr.wikipedia.org/wiki/WinDev>.