

Le développement multi-niveaux , un exemple pratique : @GP3m (Progiciel RH/Finances)

SALHI Mohand Tahar,

Ingénieur-consultant,

Fondateur et Manager AKX

AKX Cité des chalets N°7 BP 67M Oued Smar ALGER

Tél / fax : 021 51 64 50 / 021 51 34 89

E-mail : akx-salhi@wissal.dz

Résumé :

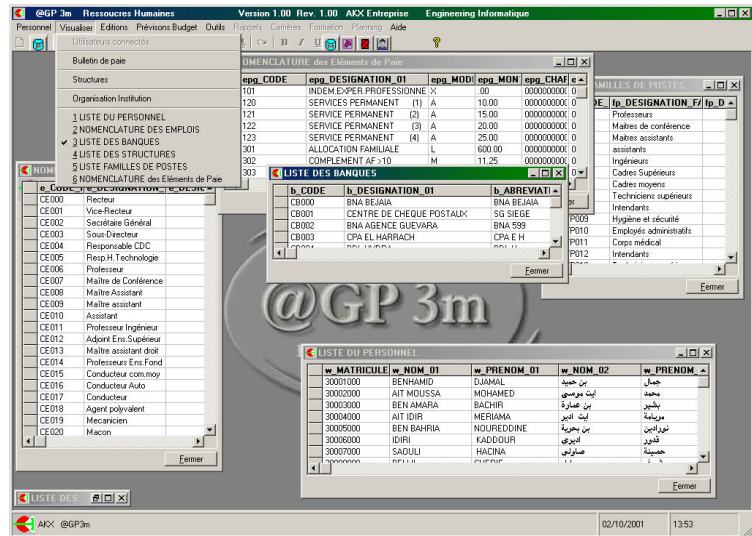
Par notre participation à ces JIE l'02, nous visons à établir des liens entre la communauté des développeurs et les utilisateurs (Institutions et Entreprise) . Les informaticiens sont souvent confrontés au moment crucial de passer à la réalisation pratique (Programmation), au délicat choix des outils de développement : le marché nous offrant de nombreux outils RAD (Rapid Application Development) . Le développement retenu pour @GP3m étant multi-niveaux et sur plate-forme Windows , la réalisation du projet s'est faite essentiellement à trois niveaux : SQL Server 7 (pour la partie serveur), Visual Basic 6 (pour la partie Client) et Front Page (Pages d'aide HTML). Le niveau client encapsule des requêtes sous formes de procédures stockées ou vues par la technologie ADO, établit des transactions avec le serveur et met en œuvre les fournisseurs d'accès OLE DB et ODBC à travers le réseau , établissant ainsi la relation Client/Serveur. L'objectif visé par cette communication est de montrer le cheminement méthodologique qui a prélué à sa réalisation. La présente communication sera complétée par un Atelier de Démonstration Pratique où @GP3m sera le pôle principal. Des informations complémentaires seront mises à la disposition du public.

Mots Clés :

@GP3m , ADO, Authentification , Autorisation, OLE DB, ODBC , Vues, Procédures Stockées, Transaction, Client/Serveur, progiciels, Ressources Humaines, Connection, Command, Recordset, Errors, Parameters, Fields, Properties, SGBDR, Curseur serveur, curseur client, Interface

Introduction

Les interfaces graphiques ont profondément bouleversé les méthodes de développement dans le monde informatique et particulièrement les applications de gestion ou progiciels destinés aux institutions et entreprises. Quoique le mode caractères a encore cours, il ne fera certainement pas long feu. Le mode graphique apporte plus de convivialité, plus de d'ergonomie, plus de facilités d'utilisation et surtout plus de fonctionnalités : textes, dessins, images diverses acquises ou virtuelles, sons et vidéos font désormais partie de l'interface qu'offrent les ordinateurs modernes – le multimédia suivant le terme consacré. Les rares réticences localisées dans le secteur financier sont en passe de succomber à ce charme irrésistible. Pour notre part, nous n'avons pas hésiter, malgré un peu de retard et ce depuis 1998 à arrêter le développement en mode caractères. Notre entreprise s'est résolument tournée vers les interfaces graphiques nettement plus modernes. Le



développement retenu pour notre progiciel @GP3m est multi-niveaux et sur plate-forme Windows en réseau de type Clients/Serveur. Les développements de ce type nécessitent l'écriture de composants à divers niveaux notamment la partie serveur (SQL) et la partie cliente (VB6), les autres niveaux n'étant qu'accessoires mais pour une meilleure qualité on ne peut s'en passer (notamment les pages d'aides HTML où Front Page a été utilisé comme générateur. Ces JIE 1'02 sont une occasion de présenter notre expérience de développement aboutie. Le sujet métier est pratiquement connu de tous : la GRH ou Gestion des Ressources Humaines. Ce que nous voulons mettre en reliefs dans cette communication, c'est surtout la méthodologie retenue pour arriver à des résultats concrets – mettant ainsi en œuvre de manière pratique et ce, à plusieurs niveaux des technologies contribuant à la construction d'un tout : le progiciel de gestion ou application de gestion spécifique. Le long du processus de développement, les impératifs économiques (marché visé) ont beaucoup contribué à faire aboutir l'expérience à une première version commerciale que nous présentons également à ces journées dans notre atelier. Dans ce qui suit, après une présentation d'usage, nous aborderons toutes les notions des technologies utilisées dans ce développement, sans les approfondir, contexte oblige, entre autres : les techniques de développement à plusieurs niveaux, les Bases de Données de Type SQL, la technologie d'accès ADO, la technique des curseurs (serveur et client), la technique des vues dans les applications clientes, la sécurité d'accès par authentification et autorisation, les fournisseurs d'accès OLE DB et ODBC et enfin la recherche de performance. Des commentaires supplémentaires accompagneront

souvent les figures et les sources présentées pour une meilleure compréhension pour le lecteur que nous invitons cordialement à visiter notre atelier.

Présentation AKX

AKX Entreprise est fondée en février 1989. Son objet principal est la conception et le développement de produits logiciels ainsi que la fourniture de services liés à l'ingénierie informatique destinés aux institutions et entreprises. En créant AKX, l'objectif était de mettre à contribution notre expérience des systèmes, du développement, de la formation et de la gestion au profit des utilisateurs professionnels. Dès le début 1990, « AKX Entreprise – Engineering Informatique » signa les premières conventions de licences avec notamment des établissements prestigieux du secteur universitaire. Depuis septembre 1999, après un important investissement en équipements et logiciels, après plusieurs mois d'études et développement, nous vous présentons notre nouveau produit logiciels dénommé **@GP3m**, acronyme pour **@KX General Planification for 3rd millenary**. Il recèle plusieurs composants personnalisables que nous mettons à la disposition de nos clients. Nous ambitionnons de construire à terme une SOLUTION INTEGRALE ou ERP (*) suivant le vocable anglo-saxon, épousant les nouvelles technologies et évoluant en parallèle avec celles-ci, en tenant compte de la spécificité de chaque client : grâce aux composants pré-validés et testés en situation réelle, nous réduisons très sensiblement les coûts et les délais de mise en production : paramètres très importants dans toute décision rationnelle.

Technique de développement à plusieurs niveaux

Ces dernières années les nouvelles technologies ont apporté une nouvelle façon de développer. Pour aboutir à des applications d'actualités utilisant diverses technologies, le seul classique langage de programmation n'est plus suffisant. De nos jours plusieurs niveaux de développement concourent à la réalisation d'un même objectif : une application aboutie et opérationnelle, fiable, robuste, performante et parfois la viabilité commerciale reste un atout pour les éditeurs de progiciels. Pour ce faire le recours à des ateliers de génie logiciel reste la meilleure solution si tel est le choix de développement. Certains ateliers offrent des outils complets permettant une réaction en symbiose à plusieurs niveaux (Bases de Données Relationnelles ou objets, langage de requêtes, Outils RAD (rapid application Development) pour les interfaces homme/machine), empaquetage et déploiement réseau, Lorsqu'il fallait choisir, cela n'a pas été une tâche facile, il y avait beaucoup de paramètres avec des poids différents heureusement, ce qui nous permet selon notre point de vue, d'établir une certaine hiérarchie pour le choix final. En définitif, pour la Description des Données et le Langage de Requêtes, SQL Server 7.0 a été retenu ; pour le développement de l'interface utilisateur VB6 (Visual Basic 6) pour son immense bibliothèque de contrôles prêts à l'emploi à fait l'unanimité, HTML n'est retenu que pour l'aide en ligne aux utilisateurs (générateur Front Page). Notre choix n'est pas opportun en réalité, la convivialité de Windows, les prix des licences des produits en question et la clientèle visée ont fait le reste. Il est indéniable que ce type de développement nécessite de fortes compétences à tous les paliers du développement : architecture des Bases de données et réseau, système de développement RAD, interfaces Windows et Web, design graphique, etc Il est rare de trouver réunies dans la même

personne toutes ces compétences – seule la formation d'une équipe où chacun apporte sa forte contribution peut accélérer la réalisation du projet.

Bases de Données de type SQL

Le serveur de données utilisé dans notre contexte est Microsoft SQL Server à sa version 7.0 . SQL (Structured Query Language) est requis pour la description de données. Ce serveur de bases relationnelles comme d'autres même type (Sybase, Oracle, DB2 entre autres), a des capacités de plusieurs dizaines, voire des centaines de tables en relationnelles. Ce système dit organise les données dans des lignes et colonnes. La modélisation de la base de données autorise l'établissement de relations entre les diverses tables la composant d'où son appellation. Une Base de Données est perçue dans SQL Server comme un tout comportant les tables organisant les données, le

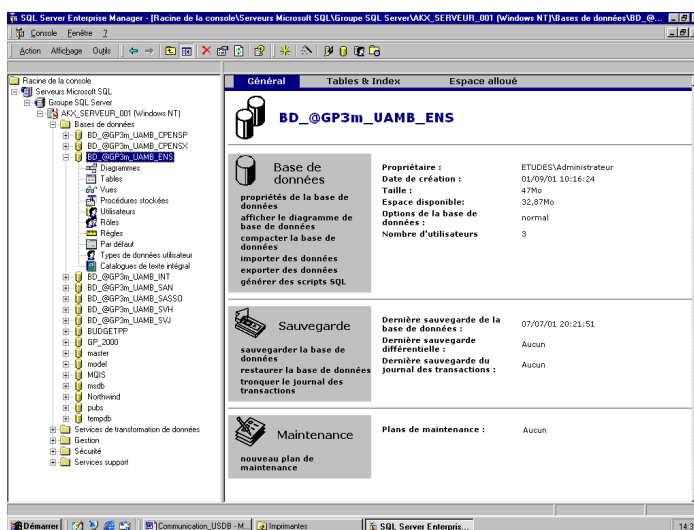
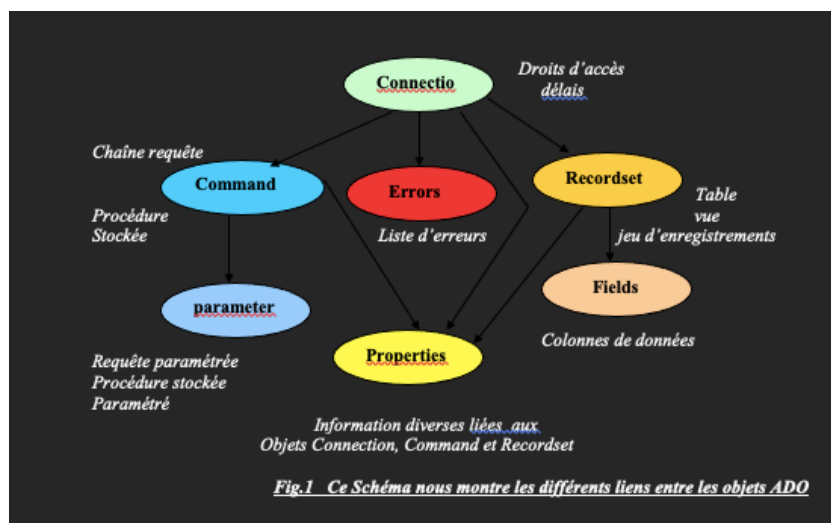


Fig.0 SQL Server 7.0 Enterprise manager

diagramme ou schéma relationnel, les règles les régissant, les vues définies sur ces tables, les procédures stockées spécialisées pour des traitements spéciaux, les déclencheurs (triggers), les utilisateurs, les rôles, les types de données définis par l'utilisateur. SQL Server permet le stockage, l'extraction et le traitement des données à partir de n'importe quel point du réseau autorisé à établir une connexion : ainsi les postes clients exécutent collectivement des opérations pour lesquelles ils sont autorisés. Toute connexion à une Base de Données SQL exige préalablement une authentification ce qui augmente le niveau de sécurité. L'interface de programmation ADO (ActiveX Data Objects) permet au programmeur, en utilisant les fournisseurs OLE DB et/ou ODBC d'établir une connexion, d'extraire des données, d'exécuter sur ces données des traitements spécifiques et de fermer à la fin la connexion ouverte. SQL Server peut gérer plusieurs Bases de données ayant une structuration schématique identique ou différente. @GP3m utilise plusieurs Bases de données possédant une structuration schématique identique.

La technologie d'accès ADO

ADO ou ActiveX Data Objects (*middleware*) est la dernière évolution des technologies d'accès aux données de Microsoft ayant des sources multiples : données relationnelles, mémoires de messagerie, système de fichiers, données web textes et graphiques. Les données qui nous intéressent ici sont celles issues du modèle relationnel (Type SQL). ADO est né de la fusion des technologies d'accès déjà éprouvées DAO et RDO respectivement pour Data Access Objects et Remote Data Objects. L'interface de programmation ADO comporte 7 objets : l'objet **Connection**, l'objet **Recordset**, l'objet **Command**, la collection **Errors**, la collection **Fields**, la collection **Parameters** et la collection **Properties**.



ADO s'appuie fortement sur l'API (Application Program Interface) OLE DB (Object Linked and Embeded for DataBase) pour réaliser les différentes opérations qu'exige l'accès à une Base de Données ou autre source. L'objet **Connection** garantit l'établissement d'une connexion vers le serveur de Bases de données en s'assurant des droits de l'utilisateur pour l'ouverture d'une session sur la Base de Données visée. Cette authentification et autorisation peuvent être réalisées en tandem par Windows NT Server et SQL Server. Une fois la connexion réalisée sans encombre, le programme utilisateur peut envoyer des requêtes par l'objet **Command** sous forme de chaîne telle que : « SELECT * FROM XPERSONNEL » ou exécuter des traitements complexes sous forme de procédures stockées par une chaîne appelante : « EXEC GENERATION_PRIME_RD ». L'objet **Recordset** contient un jeu d'enregistrements issu de la requête : cet objet définit les champs ou colonnes (telles que définies dans le schéma de la Base de données). A chacun de ces trois objets sont associées des propriétés nécessaires à leur bonne utilisation (utilisateur, délai de connexion, ID de connexion, chaîne de connexion, fournisseur d'accès, texte de commande, délai de commande, type de commande, connexion active, début des enregistrements, fin des enregistrements, taille cache, localisation du curseur, type du curseur ou verrouillage, mode, source de données, ...). Cette liste de propriétés ou collection générale **Properties** non exhaustive est complétée par des méthodes qui permettent entre autre de parcourir dans tous les sens les enregistrements. La collection **Parameters** permet de paramétrer la chaîne de requête ou la procédure stockées variant ainsi le jeu d'enregistrement suivant les valeurs affectées aux paramètres.

La collection **Fields** contient les valeurs des colonnes qui nous sont retournées. La collection **Errors** quant à elle nous renseigne sur les erreurs qui peuvent survenir durant la connexion établie, ce qui est d'une grande utilité pour élucider les cas d'erreurs qui peuvent survenir.

Exemple de Connexion ADO avec exécution d'une Procédure Stockée sous forme d'une transaction sécurisée.

```
'-----Copyright AKX-----2001 -----Partie A-----*
Public Sub Prime_rd()
Dim cnX_08 As New ADODB.Connection
Dim cmd As New ADODB.Command
Dim rs_08 As ADODB.Recordset
Screen.MousePointer = vbHourglass
'*-----définition de la connexion-----Partie B-----*
cnX_08.ConnectionTimeout = 25
cnX_08.Provider = "sqloledb"
cnX_08.Properties("Data Source").Value = ServerNameX
cnX_08.Properties("Initial Catalog").Value = DatabaseNameX
cnX_08.Properties("Integrated Security").Value = "SSPI"
'*-----ouverture connexion-----Partie C-----*
On Error GoTo ERREUR_CONNEXION
cnX_08.Open
'*----- début transaction-----Partie E-----*
cnX_08.BeginTrans
cmd.ActiveConnection = cnX_08
cmd.CommandText = "EXEC GENERATION_PRIME_RD"
cmd.CommandType = adCmdText
Set rs_08 = cmd.Execute
'*-----confirmation-----Partie F-----*
If MsgBox("Confirmation de la Transaction ? ", vbYesNo) = vbYes Then
cnX_08.CommitTrans
Else
cnX_08.RollbackTrans
End If
'*-----fermeture de la connexion-----Partie G-----*
cnX_08.Close
Screen.MousePointer = vbDefault
Exit Sub
'*-----cas d'erreur de connexion-----Partie H-----*
ERREUR_CONNEXION:
intMessage = MsgBox("Erreur de connexion...vérifier...", vbOKOnly + vbCritical)
End Sub
```

Commentaire

Cette procédure n'est bien sur utile que dans son contexte d'utilisation, notre objectif étant d'illustrer notre propos par un exemple concret. Le début (Partie A) de la procédure déclare les objets ADO **Connection** (nom connexion : **cnX_08**), **Command** (nom commande : **cmd**) et **Recordset** (nom du jeu d'enregistrement : **rs_08**). La définition de la connexion (partie B)

positionne à 25 secondes le délai de connexion, utilise le fournisseur OLE DB (« sqloledb », les variables publiques ServerNameX et DatabaseNameX renferment respectivement le nom du serveur et le nom de la base de données et sécurise l'accès par Windows NT / 2000 (valeur « SSPI »). Si aucune erreur ne pointe à l'horizon alors la connexion est établie (cnX_08.Open en Partie C) sinon détournement vers le message d'erreur (Partie H). Avant l'exécution de la requête "EXEC GENERATION_PRIME_RD" qui est dans notre appel de la procédure stockée, elle est encapsulée dans une transaction par la méthode « BeginTrans » de sorte qu'elle ne soit définitive que si elle est confirmée par « CommitTrans » ou refoulée par « RollbackTrans » (Partie F). La fermeture de la connexion est réalisée enfin par la méthode « Close » mettant fin à la procédure (partie G). Cette façon d'opérer en transaction sécurise énormément les données lorsqu'une procédure stockée ou une requête engendre beaucoup de modifications.

Curseur serveur / curseur client

L'utilité des curseurs est qu'ils offrent la possibilité d'agir sur les lignes ou les champs individuellement, notamment, les curseurs clients (frontaux) offrant ainsi une certaine granularité souvent nécessaire pour la logique de nos programmes. Un curseur dans notre contexte est emplacement en mémoire (dit cache) destiné à recevoir des données en vue de leur traitement individuellement. Les curseurs serveurs (dit d'arrière-plan) s'exécutent sur le serveur allégeant le poste client des traitements lourds implémentés sous formes de procédures stockées. On doit les utiliser à bon escient, là où ils conviennent. Lorsque le poste client opère très peu d'allers-retours en lecture/écriture le curseur client s'impose : exemple parfait pour les modules d'édition qui récupère en une fois les données et leurs font subir les formatages adéquats à l'impression. Les curseurs procurent l'avantage de rapidité car les données sont stockées dans des caches en mémoire. Les procédures stockées utilisant des curseurs offrent également plus de performance : le balayage des données est beaucoup plus facilité. Les programmeurs créent souvent les curseurs à travers des vues.

Technique des vues dans les applications clientes

Les vues sont une technique très prisée lorsqu'on utilise les curseurs : curseurs et vues se complètent parfaitement ; la vue structure les données d'un curseur à partir de colonnes issues de plusieurs tables. Seules les colonnes utiles au traitement envisagé feront partie de la vue. Cette façon de faire, permet de masquer certaines données en évitant de travailler sur les tables en entier. Il y'a forcément un gain de performance : les information inutiles ne circule plus sur le réseau.

L'exemple de définition d'une vue (code SQL) que l'on trouvera à la page suivante nous montre comment construire une vue à partir de colonnes issues de trois tables : XPERSONNEL (alias p), XFPAIE (alias fp) et XELEMENT_DE_PAIE_PERSONNE (alias ep) ; Chacune de ces tables fournit respectivement dix (10) colonnes, une (01) colonne et une (01) colonne.

Lorsque par exemple la chaîne requête «SELECT * FROM VUE_BUL_PERSONNE » est passée en commande, seules les colonnes sélectionnées et extraites des lignes présentant une concordance telle que stipulée dans la clause (WHERE fp.fp_MATRICULE = p.w_MATRICULE AND fp.fp_MATRICULE = ep.ep_MATRICULE), c'est-à-dire valeur de MATRICULE identique, seront pris en charge dans la sélection pour leur placement dans un curseur. Nous voyons ainsi l'utilité des vues : l'information

peut être servie suivant les besoins propres au traitement escompté. Les spécialistes SQL conseillent d'ailleurs d'utiliser exclusivement des vue dans les application : elles présentent l'avantage de bien cerner les droits d'accès en distillant les autorisations suivant les profils utilisateurs.

Code SQL d'un exemple de vue

```

/*-----Code--SQL-----*/
IF EXISTS (SELECT TABLE_NAME FROM INFORMATION_SCHEMA.VIEWS
           WHERE TABLE_NAME = 'VUE_BUL_PERSONNE')
  DROP VIEW VUE_BUL_PERSONNE
GO
CREATE VIEW VUE_BUL_PERSONNE WITH ENCRYPTION
AS
SELECT    p.w_MATRICULE, p.w_NOM_01, p.w_PRENOM_01, p.w_FONCTION_01,
          p.w_AFFECTATION, p.w_MODE_PAIE, p.w_NO_COMPTE,
          p.w_CATEGORIE, p.w_SECT, p.w_NO_SS, ep.ep_POINT_IND, fp.fp_TRANCHE
FROM XPERSONNEL p, XELEMENT_DE_PAIE_PERSONNE ep, XFPAIE fp
WHERE fp.fp_MATRICULE = p.w_MATRICULE AND
      fp.fp_MATRICULE = ep.ep_MATRICULE
GO

```

Authentification / autorisation

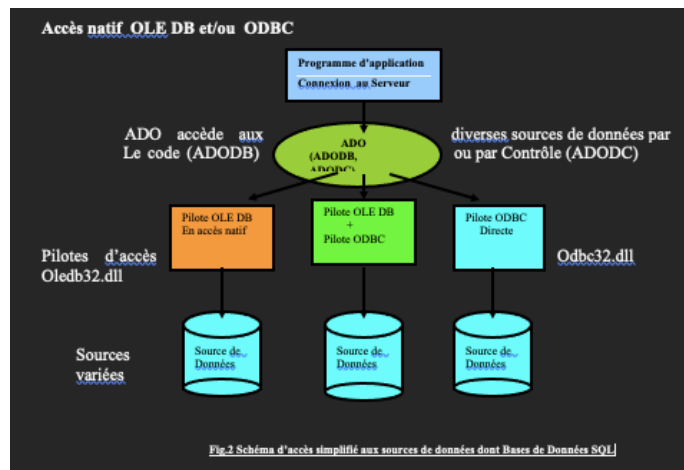
Les Systèmes de Gestion de Bases de Données Relationnelles ou SGBDR en tandem avec la plate-forme hôte (ici Windows NT 4.0 ou 2000 en version serveur) permettent une gestion convenable de la sécurité. En exigeant la reconnaissance par NT c'est déjà une première barrière pour les utilisateurs non reconnus par le réseau. Grâce au système d'exploitation une authentification est ainsi réalisée par le serveur sur un domaine. Au niveau de SQL Server même, on peut affecter les utilisateurs reconnus à des rôles déterminés (de nombreux rôles sont possibles au niveau de SQL) . Noter que l'administrateur système, lorsqu'on utilise la sécurité intégrée de NT est reconnu par SQL serveur en tant qu'utilisateur SA.

Les autorisations sont une autre façon de sécuriser les bases en donnant des autorisations sur les objets de la bases de données (tables, vues et procédures stockées) suivant les besoins : authentification et autorisation lorsqu'elles sont combinées , on obtient un niveau de sécurité suffisant pour la plupart des applications. D'autres moyens de sécurité existent et peuvent être mis en œuvre, mais notre propos n'étant pas de traiter de la sécurité, on se limitera là.

Les fournisseurs d'accès (OLE DB et ODBC)

Les fournisseurs d'accès en réseau OLE DB (Object Linking and Embedding for DataBase) et ODBC (Open DataBase Connectivity) sont fournis dans le système Microsoft sous forme de

DLL (Dynamic Link Library ou bibliothèque de liens dynamiques) dans les dossiers système respectivement Oledb32.dll et Odbc32.dll. OLE DB lorsqu'il est utilisé de manière native accède directement aux données et offre généralement plus de performance qu'ODBC (OLE DB est réservé cependant au monde Microsoft). Mais en fait, tout dépend des besoins de programmation. ODBC est le pilote universel d'accès aux Bases de Données : il existe pratiquement sur toute les plates-formes et les systèmes Microsoft le présentent dans le panneau de configuration ou dans les outils systèmes. On trouvera un exemple montrant une configuration ODBC en page suivante. Le schéma de la Fig.2 montre les différentes possibilités d'utilisation des deux pilotes d'accès indépendamment ou concurremment suivant les besoins au cas par cas. Dans @GP3m, se reporter aux source en page 5 , nous voyons un cas réel d'utilisation du pilote OLE DB en natif sans ODBC. D'autres modules utilisent ODBC suivant la configuration imagée en page suivante



La configuration d'une source de données de SQL Server par le pilote ODBC est très aisée, cela se fait à travers le panneau de configuration de Windows (Setting) . L'image ou capture d'écran ci-contre nous montre un exemple où la source de données est « MSSQL7 », où le serveur est « AKX_SERVEUR_001 » et où la Base de Données est « BD_@GP3m_UAMB_INT ». Lors de la configuration, il reste possible de faire vérifier l'accès ainsi configuré en cliquant sur le bouton de commande « Test Data source » ou test de la source de données – ce qui rassure lorsque en réseau la vérification se fait avec succès. Notez que la sécurité intégrée est positionnée : « Use Integrated security : Yes » (Windows NT Server ou Windows 2000 Server) .

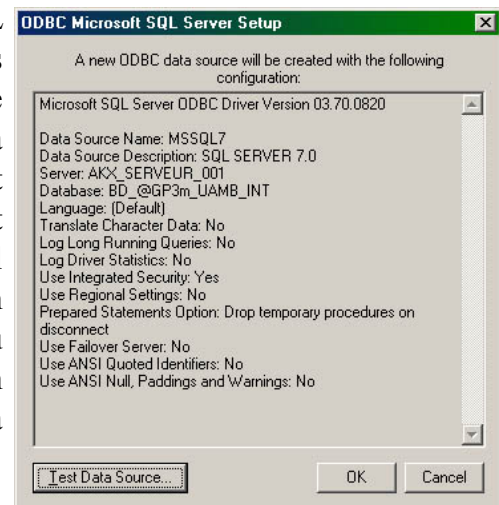


Fig.3 Exemple de configuration d'une source de données par le Pilote ODBC de Microsoft

Recherche de performance

La performance dans les applications de gestion est un facteur très important, surtout lorsque la base de données autorise de multiples accès. Ne perdons surtout pas de vue qu'un programme utilisateur suivant les besoins peut ouvrir plusieurs connexions, au besoin simultanément, les applications en réseaux sont accessibles à un grand nombre d'utilisateurs – chaque connexion mobilisant des ressources, la recherche de la performance se complique quelquefois et seuls le bon sens et les tests en situation réels ou en simulation quant à la charge prévisible permettront d'opérer les paramétrage adéquats. Le type de développement retenu ici, a nécessité avant de passer à la phase de réalisation des tests sur prototype pour mieux appréhender les difficultés liées à divers paramètres qui vont du contexte de configuration matérielle aux techniques propres à la programmation. Le réglage optimal de la taille des paquets sur le réseau (une option de SQL Server allant de 512 à 65 535 octets) est un exemple type qui peut apporter une amélioration sensible de la performance. Le choix de l'implémentation des traitements aux niveaux qu'il faut est avant tout une question de stratégie visant la performance, la facilité de mise en œuvre de la solution et surtout sa maintenabilité à long terme. Trouver le juste milieu, va certainement aider à mieux répartir les tâches entre la partie « client » ou application frontale et la partie « serveur » ou traitement d'arrière plan : c'est là l'essentiel ! Les solutions informatiques même si aujourd'hui, elles nous conviennent – on doit constamment chercher à les améliorer par l'utilisation de nouvelles technologies et d'algorithmes améliorés : c'est une façon d'assurer la pérennité d'un produit logiciel . Le côté matériel peut également apporter beaucoup à la performance d'une application : processeur, taille mémoire, mémoire cache, vitesse des disques suivant les besoins sont à même d'améliorer très sensiblement la rapidité d'accès aux données et par voie de conséquence on obtient une meilleure performance.

Conclusion

Nous aurions aimé approfondir certaines questions techniques, mais cela aurait demandé plus d'espace et dépasserait le cadre imparti. Cependant, nous espérons avoir touché l'objectif essentiel qui est avant tout de présenter un parcours technique conduisant à des progiciels dits *n-tier*. On retiendra l'utilisation de la technologie ADO de Microsoft utilisée dans la partie frontale pour accéder aux données par OLE DB ou ODBC à travers des vues de la base de données et la déportation des traitement lourds en partie serveur sous forme de procédures stockées , pour qu'ils puissent s'exécuter en arrière plan sur machine plus puissante(serveur).

BIBLIOGRAPHIE

- 1 – Documentation SQL Server 7.0 , Microsoft, Microsoft
- 2 – SQL Server 7.0 , Stephen Wynkoop, Macmillan Compus Press
- 3 – HTML et la programmation des serveurs web, Phillipe Chalaze et Daniel Charnay, Eyrolles
- 4 – Visual Basic 6 , Greg Perry, Compus Press
- 5 – Programmation Internet en C et C++, Kris Jamsa et Ken Kope , Thomson Publishing
- 6 – Windows NT 4 Server, Drew heywood, Macmillan
- 7 – MSDN (Documentation intégrée), Microsoft