



Prise en main

Rappels

Revue du package

Migration Application

FAQ

Version 2.0.4
Janvier 2006

SQLManagerX Team
Firetox@SQLmanagerX.com

1. Sommaire

1. SOMMAIRE	2
2. RAPPELS	4
2.1. RAPPEL GENERAL	4
2.2. SUIVI DES MODIFICATIONS	4
2.3. TODO LIST	4
3. COMMENT LIRE CE DOCUMENT.....	5
3.1. RAPPELS TYPOGRAPHIQUES	5
3.2. JE N'Y CONNAIS RIEN DU TOUT	5
3.3. JE NE CONNAIS PAS WINDEV DEPUIS LONGTEMPS	5
3.4. JE N'Y CONNAIS RIEN AUX SGBD.....	5
3.5. JE VIENS DE DECOUVRIR SQLMANAGERX	5
3.6. J'UTILISE DEJA UN ACCES NATIF ALTERNATIF MAIS SEUL	6
3.7. J'UTILISE DEJA SQLMANAGERX.....	6
4. PRESENTATION D'UN ACCES [ALTER]NATIF	7
4.1. RAPPEL : LES METHODES EXISTANTES STANDARDISEES	7
4.2. METHODES POUR WINDEV (INSPIREES DE LA DOCUMENTATION)	8
<i>Accès à une base de données HyperFile (diffusion libre et gratuite avec vos applications WinDev).....</i>	8
<i>Accès par un driver ODBC direct.....</i>	8
<i>Accès ODBC via le provider OLE DB.....</i>	8
<i>Accès par un provider OLE DB.....</i>	8
<i>Accès par un accès natif : Accès natif Oracle, SQL Server, AS/400,</i>	8
4.3. L'ACCES [ALTER]NATIF	9
4.4. PRESENTATION DU PACKAGE LIVRE POUR UN ACCES	10
4.5. COMMENT L'INTEGRER DANS UN PROJET EXISTANT	10
<i>Partie technique.....</i>	10
<i>Partie développement</i>	10
5. PRESENTATION DE SQLMANAGERX.....	12
5.1. L'ENCAPSULATION DU CODE SQL.....	12
5.2. MULTI-BASES DE DONNEES.....	13
5.3. SQLMANAGERX EST OPEN-SOURCE	13
6. PREMIERE UTILISATION DE SQLMANAGERX	14
6.1. PREMIER PROJET	14
6.2. ARBORESCENCE PROJET	15
<i>Création des répertoires.....</i>	15
<i>Recopie des fichiers nécessaires.....</i>	15
<i>Intégration des éléments dans le projet</i>	17
6.3. CREATION DU CODE D'INITIALISATION DU PROJET.....	17
6.4. CREATION DE LA TABLE VILLES DANS LA BASE	17
<i>Création de la table</i>	17
<i>Alimentation de la table.....</i>	18
6.5. CREATION DE LA CLASSE D'ACCES VILLES.....	18
<i>Création de la classe</i>	18
<i>Déclaration globale.....</i>	18
6.6. CREATION DE NOTRE PREMIERE FENETRE EN MODE FICHE	19
<i>Création de la fenêtre</i>	19
<i>Insertion de la fonction d'affichage.....</i>	19
<i>Insertion des boutons mode parcours.....</i>	20
<i>Insertion des boutons nouveau, modifier, supprimer.....</i>	20
6.7. CREATION DE NOTRE PREMIERE FENETRE EN MODE TABLE	22
<i>Création de la table + alimentation en mode fetch global</i>	22
<i>Création de la table + alimentation en mode fetch partiel.....</i>	23
6.8. LES ELEMENTS UTILES	24
<i>La gestion des filtres. Quelle méthode utiliser ?.....</i>	24
<i>La gestion des vues ?</i>	24
<i>Utilisation d'une combo.....</i>	25
7. LES IMPRESSIONS AVEC SQLMANAGERX	26

7.1.	IMPRESSION MODE FICHE.....	26
7.2.	IMPRESSION RAPIDE MODE TABLE	26
7.3.	IMPRESSION COMPLETE	26
8.	ALLER PLUS LOIN AVEC SQLMANAGERX	27
8.1.	CONSTRUCTION AUTOMATIQUE D'UNE FENETRE TABLE.....	27
	<i>Création de la table + alimentation en automatique.....</i>	<i>27</i>
8.2.	GESTION AUTOMATISEE D'UNE FENETRE FICHE.....	28
8.3.	LA GESTION DES LOBS (LARGE OBJECT) AVEC SQLMANAGERX.....	28
	<i>Insérer un BLOB.....</i>	<i>28</i>
	<i>Lecture d'un BLOB.....</i>	<i>28</i>
8.4.	ACCES MULTI-BASES AVEC SQLMANAGERX.....	28
	<i>Création d'un fichier de paramétrage</i>	<i>29</i>
	<i>Code à ajouter à l'initialisation du projet</i>	<i>29</i>
8.5.	FONCTIONNEMENT DE PHP4WD.....	30
	<i>Rappels du besoin</i>	<i>30</i>
	<i>Architecture proposée.....</i>	<i>31</i>
	<i>Un petit exemple</i>	<i>31</i>
8.6.	EXEMPLE AVEC WDSRIPT	32
9.	FAQ (FOIRE AUX QUESTIONS).....	33
9.1.	JE NE TROUVE PAS DE DOCUMENTATION EN ANGLAIS, EST-CE NORMAL ? (I DO NOT FIND ENGLISH DOCUMENTATION, IS THIS NORMAL ?).....	33
9.2.	ON PARLE DE SQLMANAGERX, ON PARLE DES ACCES [ALTER]NATIFS MAIS PEUT-ON UTILISER L'UN SANS L'AUTRE ?	33
9.3.	PUIS-JE UTILISER LES ACCES (SOUS LICENCE WD-LIBRE) DANS UN SOFT COMMERCIAL ? DOIS-JE RETRIBUER L'AUTEUR ?	33
9.4.	QUAND UTILISER SQLMANAGERX, QUAND UTILISER LES ACCES [ALTER]NATIFS DIRECTEMENT ?.....	33
9.5.	CONCERNANT LES DLL, OU VAUT-IL MIEUX LES PLACER ?	34
9.6.	LA GRANDE QUESTION : QUELLE BASE POUR MON PROJET ?	34
9.7.	POURQUOI UTILISEZ-VOUS DES TABLES MEMOIRES ?	34
9.8.	J'UTILISE REGULIEREMENT WDMAP, MAINTENANT JE FAIS COMMENT ?	34
9.9.	POURQUOI TOUTES LES METHODES DISPONIBLES COMMENCENT-ELLES PAR MYSQL ?	35
9.10.	EXISTE-T-IL UNE VERSION WINDEV5.5 DES ACCES NATIFS ? DE SQLMANAGERX ?	35
9.11.	ACCES NATIFS : QUELS SONT LES NUMEROS DE REQUETES PERMIS?.....	35
9.12.	ACCES NATIFS : POURQUOI DOIS-JE FAIRE MYSQLFERME(X)?.....	35
9.13.	Y-A-T-IL COMPATIBILITE ASCENDANTE ENTRE LES VERSIONS SUCCESSIVES DES ACCES NATIFS ?	36
9.14.	SQLMANAGERX EST-IL COMPATIBLE WEBDEV ?.....	36
9.15.	MYSQL4WD : PUIS-JE ME CONNECTER SUR 2 SERVEURS DIFFERENTS ?.....	36
9.16.	MYSQL4WD : J'AI UN MESSAGE A L'EXECUTION DE MON PROJET "RUN OUT OF SYNC" ?	36
9.17.	ORACLE4WD : AU LANCEMENT DE MON APPLICATION J'AI LE MESSAGE «IMPOSSIBLE DE CHARGER LA DLL : PAS PRESENTE OU DEFECTUEUSE » ?.....	36
9.18.	ACCES NATIFS : POURQUOI CERTAINS DES ACCES NATIFS GENERENT DES FICHIERS .FIC?.....	37
9.19.	SQLMANAGERX : EN REGARDANT VOTRE CODE, JE ME RENDS COMPTE QUE VOUS NE GEREZ QUE 4 TYPES DE DONNEES. COMMENT FAITES VOUS ?.....	37
10.	DIVERS.....	38
10.1.	GENEALOGIE DES SGBDR.....	38
10.2.	QUEL ACCES POUR MA BASE ?	39
10.3.	LES EQUIVALENCES DE TYPES.....	40
10.4.	LES ACCES NATIFS DE L'EDITEUR	41
10.5.	LIENS ET REFERENCES	42
11.	INDEX	43

2. Rappels

2.1. *Rappel général*

WinDev©, WebDev©, WinDev Mobile© et HyperFile© sont des marques déposées par PCSoft. La documentation ici présente n'a aucun lien direct ou indirect avec cette société.

2.2. *Suivi des modifications*

- Version 1.1.6 : réécriture de l'exemple.
- Version 1.1.7 : ajout des états génériques SQLManagerX dans l'arborescence projet.
- Version 1.1.8 : revue de la FAQ (ajout de questions).
- Version 2.0.0 : ajout des impressions. Test de tout l'enchaînement didactique proposé. Ajout de la nouvelle fonctionnalité V4 SQLTableConstruit().
- Version 2.0.1 : compléments de la FAQ, ajout de la généalogie des SGBD.
- Version 2.0.2 : un exemple avec php4wd (architecture de fonctionnement). Revue de la mise en page, revue des rappels typo, revue de l'arbre de connaissance, ajout d'un rappel, revue de mise en forme plus aérée.
- Version 2.0.3 : relecture. Ajout de la gestion des vues
- Version 2.0.4 : ajout d'un index, ajout des liens et références, ajout de la liste des accès natifs de l'éditeur.

2.3. *TODO List*

- Expliquer les impacts sur le résultat en fonction des paramètres passés aux méthodes.
- Préciser les versions des accès utilisés, les versions des SGBD testés

3. Comment lire ce document

Tout dépend de ce que vous devez faire. Nous avons recensé ci dessous les principaux cas de figure. Néanmoins, nous partons du principe que vous avez un minimum de bagages informatiques.

3.1. *Rappels Typographiques*



Cette icône annonce une information ou un rappel. *La teneur de ce point est présentée en italique.*



Cette icône annonce un point important. *La teneur de ce point important est présentée en italique.*



Cette icône annonce un exemple de code source. `CE CODE EST PRESENTE EN PETITES MAJUSCULES.`

3.2. *Je n'y connais rien du tout*

SQLManagerX se basant sur l'AGL¹ WinDev, nous vous invitons avant toute chose à prendre contact avec un commercial pour l'acheter.

Si vous avez WinDev installé sur votre machine, vous pouvez passer au point suivant.

3.3. *Je ne connais pas WinDev depuis longtemps*

Dans votre cas il sera parfois difficile de suivre ce document. Certains aspects (utilisation des classes, héritage, utilisation de DLL) vont vous paraître bien compliqués. Nous vous conseillons de bien lire la documentation sur ces points avant de suivre ce guide.

3.4. *Je n'y connais rien aux SGBD*

Ce n'est pas très grave vu que le but de SQLManagerX est de faire abstraction de la base de données. Vous pouvez passer au point suivant. Le minimum est que vous ayez une base de données accessible depuis votre poste de développeur avec les droits nécessaires.

3.5. *Je viens de découvrir SQLManagerX*

Ordre de lecture préconisé :

- Présentation d'un accès [alter]natif à la page 7
- Présentation de SQLManagerX à la page 12
- Première utilisation de SQLManagerX à la page 14
- Les impressions avec SQLManagerX à la page 26
- Aller plus loin avec SQLManagerX à la page 27

¹ AGL : Atelier de Génie Logiciel

3.6. J'utilise déjà un accès natif alternatif mais seul

Ordre de lecture préconisé :

- Présentation de SQLManagerX à la page 12
- Première utilisation de SQLManagerX à la page 14
- Les impressions avec SQLManagerX à la page 26
- Aller plus loin avec SQLManagerX à la page 27

3.7. J'utilise déjà SQLManagerX

Ordre de lecture préconisé :

- Aller plus loin avec SQLManagerX à la page 27

4. Présentation d'un accès [alter]natif

4.1. Rappel : les méthodes existantes standardisées

Il existe différentes méthodes pour accéder à une base de données tierce (entendons par tierce tout autre SGBD² que HyperFile) au travers d'une application WinDev :

- **ODBC** (*Open Database Connectivity*) fournit une interface API (*Application Program Interface* - Interface de programmation) que différents éditeurs de bases de données implémentent par l'intermédiaire de pilotes ODBC spécifiques à un système SGBD particulier. Votre application utilise cette API pour appeler le gestionnaire de pilotes ODBC, qui transmet les appels au pilote approprié. Le pilote, à son tour, interagit avec le SGBD par l'intermédiaire de SQL.
- **DAO** (*Data Access Objects*) utilise le moteur de bases de données Microsoft Jet pour fournir un ensemble d'objets d'accès aux données. Il est optimisé autour du moteur de bases de données Microsoft Jet (Access !).
- **OLE DB** (*Object Linking and Embedding DataBases*) est une api COM qui permet un accès unifié à toutes sortes de sources de données. Des fournisseurs OLE DB existent pour la plupart des serveurs de bases de données, Active Directory, les feuilles de calcul Excel, les fichiers XML ou les fichiers texte (.txt ou .csv).
- **ADO** (*ActiveX Data Object*) est un composant ActiveX permettant d'accéder aux bases de données de façon beaucoup plus facile que les méthodes précédentes sans se soucier de tout ce qui est allocation des environnements de travail (cf. programmation avec la couche basse d'ODBC). ADO fournit des objets (les principaux sont *Connection*, *Command* et *Recordset*) qui permettent de se connecter à une base et de réaliser des requêtes SQL (*Structured Query Language* – langage structuré de requête) sur cette base.

Mais quel est le lien entre toutes ces méthodes d'accès :

- ODBC a été avant tout conçue par Microsoft pour répondre aux besoins des programmeurs C/C++.
- Microsoft introduit RDO pour faciliter le travail des programmeurs en Visual Basic (RDO se base sur l'ODBC).
- Microsoft introduit DAO pour accéder à Access en pleine évolution.
- Microsoft unifie tout cela et développe OLE-DB mais son exploitation est particulièrement difficile aussi bien en C++ qu'en VB.
- Microsoft met alors au point ADO et tout récemment ADO.net

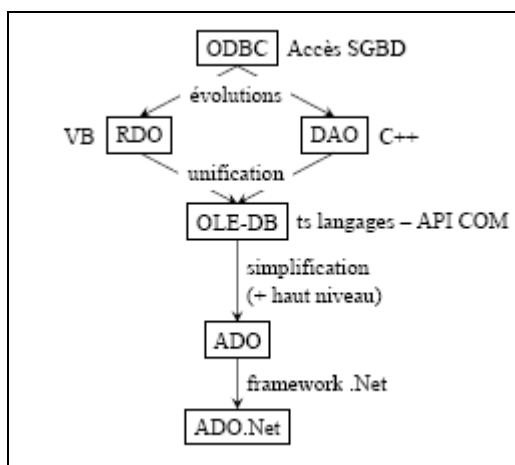


Figure 1: Historique

² SGBD : *Système de Gestion de Bases de Données*

4.2. Méthodes pour WinDev (inspirées de la documentation)

WinDev propose différents modes d'exécution des requêtes SQL en fonction du type d'accès effectué à la base de données.

Accès à une base de données HyperFile (diffusion libre et gratuite avec vos applications WinDev)

Aucune contrainte d'installation.

Les fonctions SQL et HyperFile (HLitxxx, ...) peuvent être utilisées avec ce type d'accès.

Accès par un driver ODBC direct

Les caractéristiques de la connexion à la base de données doivent être définies dans l'administrateur ODBC de Windows. Seules les fonctions de type SQL sont utilisables pour ce type d'accès. Les fonctions HyperFile (HLitxxx, ...) ne sont pas utilisables.

Accès ODBC via le provider OLE DB

Ce type d'accès utilise un provider OLE DB spécifique. Ce type d'accès est déconseillé car plus lent qu'un accès par un driver ODBC. En effet, les performances sont moins bonnes que par un driver ODBC direct car l'accès se fait à la fois par le driver ODBC et par le provider OLE DB.

Les fonctions HyperFile (HLitxxx, ...) et SQL peuvent être utilisées avec ce type d'accès.

Il est nécessaire de définir les caractéristiques de la connexion à la base de données dans l'administrateur ODBC de Windows. Le provider ainsi que le MDAC 2.6 (ou supérieur) doivent être installés sur le poste.

Accès par un provider OLE DB

Ce type d'accès utilise un provider OLE DB. Le provider ainsi que le MDAC 2.6 (ou supérieur) doivent être installés sur le poste.

Les fonctions SQL et HyperFile (HLitxxx, ...) peuvent être utilisées avec ce type d'accès.

Accès par un accès natif : Accès natif Oracle, SQL Server, AS/400, ...

Pour chaque type d'accès natif, il est nécessaire de posséder un module complémentaire (et payant sauf pour MySQL NDLR) à WinDev. Il permet d'accéder à la base sans drivers externes depuis un programme en W-Langage.

L'accès est direct sur base sans passer par une couche intermédiaire : MDAC inutile, OLE DB inutile, ODBC inutile. Seule l'installation de la couche client sur le poste de l'utilisateur est nécessaire.

La structure de la base peut être récupérée dans l'analyse WinDev.

Le RAD permet de générer du code avec les fonctions Hxxx (HLitSuivant, ...) ou SQLxxx.

L'outil visionneur de données WDMAP est utilisable sur la base de données.

Les fonctions SQL et HyperFile (HLitxxx, ...) peuvent être utilisées avec ce type d'accès.

Ce type d'accès est plus rapide que les accès par ODBC ou par un provider OLE DB.

4.3. L'accès [alter]natif

Un accès [alter]natif s'appuie sur les accès bas niveau fournis par l'éditeur du SGBD accédé (pas de couches intermédiaires utilisées).

Les accès bas niveau plus communément appelés API sont fournis bien souvent par un jeu de fichiers écrits en C (par exemple *OCI*, Oracle Call Interface pour Oracle, *DBLIB* pour SQL Server, *Jet* pour Access). Nous devons avoir un « lien » entre ces API et notre application WinDev. Pour se faire, une DLL (*Dynamic Link Library* - Bibliothèque de Liens Dynamique) est l'élément le plus simple à mettre en œuvre. On se base donc sur les API pour créer une DLL à l'aide d'un logiciel spécifique (entre autre : Dev-Cpp, Visual Studio, Borland C++) tout en répondant à nos besoins fonctionnels (se connecter à la base, effectuer des ordres SQL,...). L'utilisation d'une DLL présente un double intérêt :

- Une DLL est plus facilement exploitable par WinDev (ou tout autre application Windows)
- Une DLL est plus facilement intégrable dans un projet WinDev

Pour terminer une classe WinDev encapsule les différents appels à la DLL pour faciliter le développement.

Le tout permet :

- Une faible consommation de ressource mémoire et CPU avec une seule DLL chargée.
- Une intégration dans votre projet facilitée.
- Un mode de programmation facile à maîtriser.
- Une génération de type RAD (à venir) grâce aux fichiers MDL et MDE de WinDev.

Le tout ne permet pas :

- De créer vos requêtes dans l'éditeur WinDev.
- De récupérer la structure des tables dans une analyse WinDev.
- D'utiliser WDMMap car l'analyse est inexistante (voir la FAQ « J'utilise régulièrement WDMMap, maintenant je fais comment ? » page 34).

4.4. Présentation du package livré pour un accès

Le package livré a souvent une arborescence type :

- La racine contient généralement 2 fichiers :



- `changelog.txt` : trace de toutes le versionning de l'accès



- `LisezMoi.txt` : reprend la licence WD-Libre



- **DLL** contient les sources de la DLL. Leur mise en forme peut varier en fonction de l'outil utilisé pour générer la DLL
- **Windev7** contient un projet type.



Par défaut, le projet exemple fourni utilise la base de données créée par défaut avec votre SGBD.

4.5. Comment l'intégrer dans un projet existant

Partie technique

Pour utiliser l'accès que vous venez de récupérer, vous avez besoin de :

- Copier la DLL présente dans le répertoire Windev7/Exe du projet type dans le répertoire Exe de votre projet,
- Copier la classe présente dans le répertoire Windev7 dans le répertoire hébergeant les classes de votre projet. Certaines classes utilisent la classe générique `c_log4WD.wdc` (Celle-ci permet de tracer les ordres envoyés à la base), il est nécessaire de la copier aussi.

Vous avez donc maintenant les fichiers nécessaires au bon fonctionnement de l'accès. Il faut maintenant importer la (ou les) classe(s) dans votre projet (clic droit dans le kouglof et importer une classe).



Par soucis de vérification immédiate, une compilation totale du projet apparaît intéressante.



La DLL de l'accès devra être livrée dans le package de l'installation à destination de l'utilisateur final.

Partie développement

Pour pouvoir exploiter l'accès, il faut ouvrir une connexion à la base de données. Ensuite, vous pouvez vous baser sur les nombreux exemples fournis dans le projet type pour commencer vos développements.

Les exemples fournis ci-dessous sont liés à l'accès *Oracle4WD*. Ils seront repris et expliqués en détail dans les pages suivantes.

Déclaration :

```
// Instance de la classe accès natif  
MonOracle est c_Oracle4WD
```

Connexion :

```
//Gère la connexion à la base Oracle avec comme paramètre la chaîne  
//de connexion au format user/pwd@base  
  
SI PAS MonOracle:mysqlConnecte("user/pwd@base") ALORS  
    Info (MonOracle:mysqlErreur+ "/" + MonOracle:mysqlGetErrorMessage ())  
FIN
```

Affichage dans une table mémoire :

```
// Remplir une table mémoire avec une requête  
TableSupprimeTout(TABLE1)  
  
// Exécution avec ouverture implicite du curseur 0  
retCode = MonOracle:mysqlExec("SELECT work_list, lnotes FROM work_list ",0)  
  
SI (retCode =1) ALORS  
    MonOracle:mysqlTable(0, "TABLE1")  
SINON  
    Info (MonOracle:mysqlErreur+ "/" + MonOracle:mysqlGetErrorMessage ())  
FIN  
  
// Fermeture du curseur 0  
MonOracle:mysqlFerme(0)
```

5. Présentation de SQLManagerX

La Classe SQLManagerX permet de gérer les tables SQL d'une façon simple et rapide en ne manipulant que des objets représentant une ligne³ d'une table. Cette technique permet de ne pas s'occuper du code SQL qui découle pour faire des sélections, des insertions, des modifications, des suppressions. La gestion des tables SQL au travers de la classe s'apparente à celle des fichiers HyperFile. C'est le premier avantage de SQLManagerX : **encapsuler le code SQL lié à la gestion d'une table SQL.**

Deuxième avantage de SQLManagerX : cette classe est couplée avec les classes d'accès [alter]natifs présentés précédemment. Cela vous permet de gérer les tables en provenance de différentes bases (MySQL, Oracle, PostgreSQL, SQLite,...) avec un code unique.

SQLManagerX est également compatible avec le projet [Wdscript](#)⁴, il suffit dans le projet de mettre le nom du fichier WDL correspondant aux classes de SQLManagerX. Ensuite le script connaît les objets et peut afficher le résultat de votre requête dans une page Web.

5.1. L'encapsulation du code SQL

SQLManagerX permet de gérer simplement les tables d'une base SQL : le but initial étant de se rapprocher de la gestion des fichiers HyperFile. Pour cela un système de classe est utilisé. Chacune des classes correspond à une table au niveau de la base de données. Chaque colonne correspondant à un membre de la classe. Les ordres SQL de type parcours, mise à jour, filtre, recherche,... sont alors repris depuis la classe SQLManagerX et utilisés par héritage dans l'objet déclaré.



Le développeur saisit ce code :

```
SQLTableVerseEcran()  
SQLUpdate()
```

et SQLManagerX envoie comme ordre à la base :

```
UPDATE VILLES set Zone = 1  
WHERE CodeP='59000' AND Commune='LILLE'
```

Ce mode de programmation permet de faire oublier au développeur le code SQL généré et envoyé à la base de données. Il peut alors s'orienter sur les parties les plus spécifiques de son développement. Nous sommes à l'air des L4G voire des L5G, il est maintenant important que les programmeurs ne perdent plus de temps sur les actions simples et les codes redondants. Leur temps doit être au maximum utilisé pour les développements qui demandent plus de réflexion, pour respecter les délais de plus en plus courts. Le code standard doit être rapidement écrit et doit être dans la mesure du possible le plus sur possible.

³ Un *tuple* pour les puristes.

⁴ **WDScript** est un module de type CGI pour application Web. Celui-ci permet aux utilisateurs de WinDev de créer des sites dynamiques accédant à des bases de données HyperFile (ou autre) en utilisant le *W-Langage* intégré à WinDev.

5.2. Multi-bases de données

Autre point lié au précédent, le code SQL bien que standardisé contient quelques subtilités pour telle ou telle base (on peut citer par exemple les fonctions de gestion de dates ou de chaînes de caractères). La gestion au travers SQLManagerX permet d'oublier ces contraintes pour les requêtes SQL.

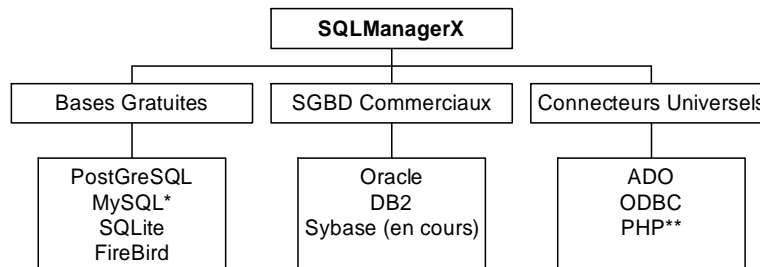


Figure 2: SGBD accédés au travers de SQLManagerX

Dans la vision des choses des auteurs, le code du programme doit être au maximum indépendant du système de base de données utilisé. Il faut arriver à avoir un code de programme indépendant de la source de données. Cela dans un but d'ouverture et de compatibilité avec les différentes bases de données du marché. SQLManagerX répond en tout point à ces préoccupations : *SQLManagerX permet de développer des applications se basant sur des bases de données sans pour autant connaître le langage SQL.*

5.3. SQLManagerX est OPEN-SOURCE

SQLManagerX et sa suite sont **OPEN-SOURCE** soumis à la licence WD-Libre. Sauf quelques exceptions (DLL Oracle4WD, DLL SQLite4WD, SQLManagerX-Convert, SQLManagerX-Outils) vous disposez du code source dans son intégralité depuis le code W-Langage jusqu'au code C permettant de générer la DLL. **Vous pouvez modifier et adapter SQLManagerX à vos besoins.**

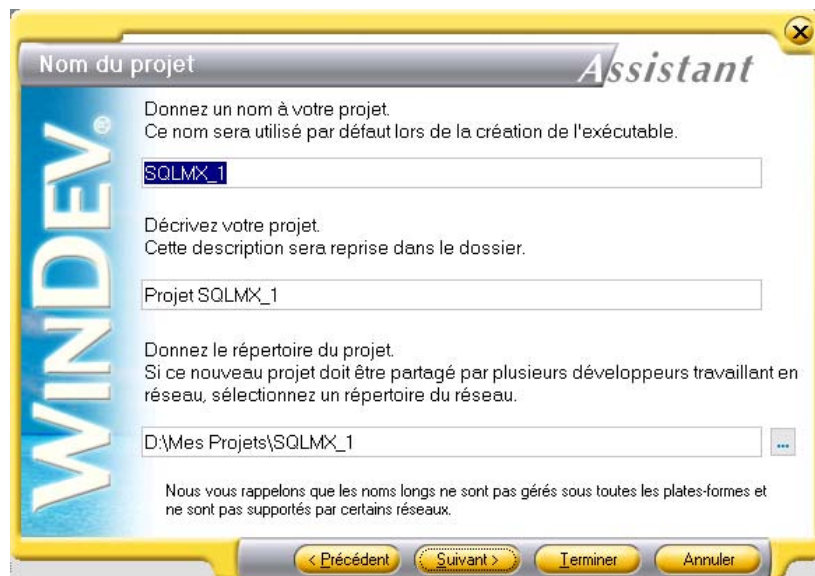
Vous pouvez faire bénéficier à la communauté des développeurs WinDev de vos remarques et/ou de vos ajouts.

6. Première utilisation de SQLManagerX

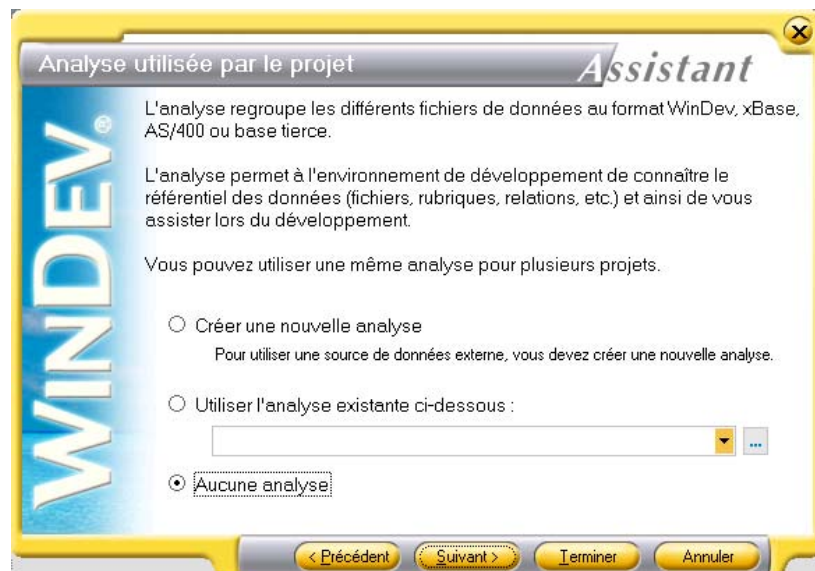
6.1. Premier Projet

Comme premier projet, nous allons simplement créer un ensemble de fenêtres tel que le ferai le RAD⁵ sur pour un fichier HyperFile.

Comme pour tout projet WinDev 7.5, nous allons le créer à partir de l'AGL. Pour cela faites Fichier/Nouveau et choisissez nouveau projet.

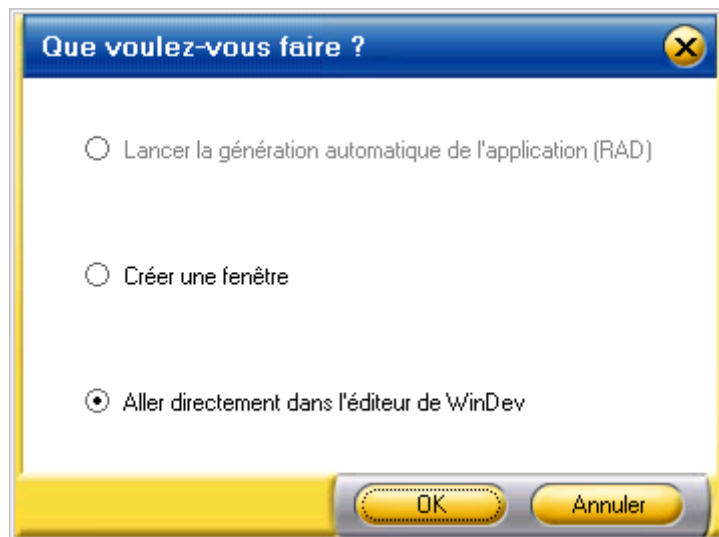


Saisissez SQLMX_1 comme nom de projet



Prenez soin de bien sélectionner « Aucune analyse » puis cliquez sur Terminer.

⁵ RAD : *Rapid Application Development*



Vous avez donc créé votre projet. Nous allons maintenant l'enrichir des composants logiciels livrés avec SQLManagerX.

6.2. Arborescence projet

Création des répertoires

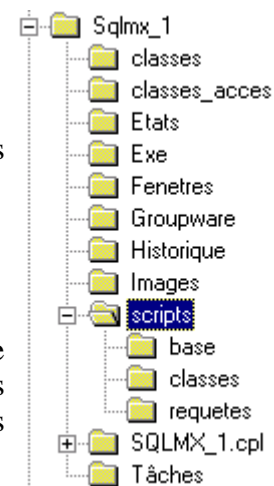
Pour plus de facilité nous allons créer une arborescence type. Nous vous conseillons vivement de la reprendre dans vos futurs projets.

Actuellement l'arborescence des votre projet est la suivante



Faites le nécessaire pour avoir l'arborescence ci contre :

- **classes** contiendra toutes les classes du projet
- **classes_acces** contiendra toutes les classes liées aux objets d'accès aux tables par SQLManagerX
- **Etats** contiendra tous les états
- **Fenêtres**, les fenêtres
- **Scripts** contiendra tous les scripts SQL utilisés dans le cadre de votre développement (script de création de la base, code des classes si vous passez par SQLManagerX-Outils, script des requêtes SQL).



Recopie des fichiers nécessaires

Pour fonctionner avec SQLManagerX, votre projet à besoin :

- De toutes les classes SQLManagerX et natives,
- Des deux états génériques fournis avec SQLManagerX,

- Des différents éléments liés à chaque accès natif.

Dans le répertoire Exe du projet il faudrait les fichiers DLL suivants :

- ado4wd.DLL Accès ado
- fbclient.DLL DLL d'interface entre la base FireBird et la c_FB4WD.wdc
- *libmysql.DLL* DLL Mysql à récupérer sur <http://www.mysql.com/> (API propriétaire),
utiliser libmysql.DLL version 5.0.15.
- *libpq.DLL* DLL Postgresql présente dans le package ou à récupérer sur
<http://www.postgresql.com/> (API propriétaire)
- mysql4wd.DLL DLL d'interfaçage entre libmySQL.DLL et la c_mysql4wd.wdc
- oracle4wd.DLL DLL d'interfaçage entre le client Oracle (API propriétaire) et la
c_oracle4wd.wdc
- postgresql4wd.DLL DLL d'interfaçage entre libpq.DLL et la c_postgresql4wd.wdc
- sqlite4wd.DLL DLL d'interface entre la base SQLite et la c_SQLite4WD.wdc
- rtl60.bpl Ressource Borland c++ nécessaire à la DLL SQLite4wd.DLL
disponible dans l'accès SQLite4WD



Par exemple pour exploiter SQLManagerX avec une base MySQL prévoir l'utilisation des DLL libmySQL.DLL et mysql4WD.DLL.

Dans le répertoire classes nous allons copier les fichiers suivants :

- c_ado4wd.wdc classe pour accès ADO
- c_fb4wd.wdc classe pour accès FireBird ou InterBase
- c_hf4wd.wdc classe pour accès HF
- c_log4wd.wdc classe de log, nécessaire pour la majorité des accès
- c_mysql4wd.wdc classe pour accès Mysql
- c_natif4wd.wdc classe exemple pour accès Natif de l'éditeur
- c_oracle4wd.wdc classe pour accès Oracle
- c_php4wd.wdc classe pour accès par PHP
- c_postgresql4wd.wdc Classe pour accès Postgresql
- c_sqlite4wd.wdc Classe pour accès SQLite
- SQLManagerX.wdc

Dans le répertoire Etats nous allons copier les fichiers suivants :

- R_SQLEtat Etat générique SQLManagerX
- R_SQLEtatP Etat générique SQLManagerX mais en mode Portrait

Intégration des éléments dans le projet

Intégrer les classes nécessaires dans votre projet par la méthode que vous préférez. Par exemple pour MySQL : prévoir `c_log4wd.wdc`, `c_mysql4wd.wdc`, `SQLmanagerx.wdc`.

Refaites une compilation pour valider le tout.

6.3. Création du code d'initialisation du projet

Copiez ces quelques lignes dans le code d'initialisation de votre projet.

```
GLOBAL
convSql est c_mySQL4WD

BaseHost est une chaîne //serveur
BaseUser est une chaîne //utilisateur MySQL
BasePasswd est une chaîne //Mot de passe MySQL
BaseDB est une chaîne //Base de données

BaseHost="localhost"
BaseUser="root"
BasePasswd=""
BaseDB="test"

// Déclaration de la connexion MySQL
// A faire avant toute chose (la déclaration des classes entre autre)
SI PAS convSql:mysqlConnecte(BaseHost,BaseUser,BasePasswd,BaseDB) ALORS
Info("Pas de connexion SQL: "+convSql:mysqlGetErrorMessage())
FinProgramme()
FIN

convSql:mysqlDebugMode = Vrai // activation du mode debug

// Déclaration des classes d'accès à la base
```

Explications : nous avons créé 4 variables pour pouvoir se connecter à la base MySQL.



Pour une connexion à une autre base, il sera opportun de créer les variables utiles.

L'activation du mode Debug (`convSql:mysqlDebugMode = Vrai`) permet de tracer les ordres SQL envoyés à la base de données dans un fichier texte à l'aide de la classe `c_log4wd`.



Très utile en phase de développement, il ne faut pas oublier de le désactiver avant déploiement chez le client.

6.4. Création de la table villes dans la base

Création de la table

```
CREATE TABLE T_FORUM_FRM (
  FRM_ID          INTEGER          NOT NULL PRIMARY KEY,
  FRM_NOM         CHAR(64)         NOT NULL,
  FRM_SUJET       VARCHAR(255)     NULL,
  FRM_DATE_CREATION DATE          NOT NULL
)
```

Alimentation de la table

```
INSERT INTO T_FORUM_FRM (FRM_ID, FRM_NOM, FRM_SUJET, FRM_DATE_CREATION)
VALUES (1, 'sgbd', 'Bases de données', '2005-01-01')
INSERT INTO T_FORUM_FRM (FRM_ID, FRM_NOM, FRM_SUJET, FRM_DATE_CREATION)
VALUES (2, 'mysql', 'RDBMS IBM DB2', '2004-02-19')
INSERT INTO T_FORUM_FRM (FRM_ID, FRM_NOM, FRM_SUJET, FRM_DATE_CREATION)
VALUES (3, 'windev', 'AGLWINDEV', '2001-03-21')
INSERT INTO T_FORUM_FRM (FRM_ID, FRM_NOM, FRM_SUJET, FRM_DATE_CREATION)
VALUES (4, 'oracle', 'RDBMS ORACLE', '1997-03-01')
```

6.5. Création de la classe d'accès villes

Création de la classe

Il existe 2 possibilités pour créer votre classe : soit en passant par SQLManagerX-Outils soit en la créant de toute pièce. C'est cette seconde méthode que nous allons exposer ici.

Il faut tout d'abord créer une classe `c_t_forum_frm`.

Déclaration de `c_t_forum_frm`

```
C_T_FORUM_FRM EST UNE CLASSE
PUBLIC
  UN OBJET SQLMANAGERX
  M_FRM_ID      EST UN ENTIER
  M_FRM_NOM     EST UNE CHAINE
  M_FRM_SUJET   EST UNE CHAINE
  M_FRM_DATE_CREATION EST UNE DATE
FIN
```



Remarquez le `c_nomdelatable` pour nommer la classe. Cette notion est primordiale car elle sera reprise dans l'appel du constructeur ci dessous (variable `p_prefixeClasse`)

Remarquez le `m_` pour nommer chacun des membres de la classes. Cette notion est primordiale car elle sera reprise dans l'appel du constructeur ci dessous (variable `p_prefixeMembre`)

Constructeur

```
PROCEDURE CONSTRUCTEUR(P_SQLCLASS = NULL, P_TABLESQL = "", P_NOMOBJET = "",
P_PREFIXECLASSE = "C_", P_PREFIXEMEMBRE = "M_")
CONSTRUCTEUR SQLMANAGERX(P_SQLCLASS, P_TABLESQL, P_NOMOBJET, P_PREFIXECLASSE,
P_PREFIXEMEMBRE)
```



Ne pas oublier de sauvegarder votre classe dans le répertoire `classes_acces` en non dans le répertoire racine de votre projet.

Déclaration globale

Ajoutez la ligne suivante dans le code d'initialisation de votre projet juste en dessous de la ligne « //

Déclaration des classes d'accès à la base »

```
I_T_FORUM_FRM est un c_t_forum_frm (convSql, "t_forum_frm", "I_T_FORUM_FRM")
```

Explication :

`I_T_FORUM_FRM` est notre instance de la classe `C_T_FORUM_FRM` (objets disponible dans le TreeView des objets du projet).

Lors de son instanciation, `I_T_FORUM_FRM` va utiliser la connexion principale (définie explicitement dans le code du projet) d'accès à MySQL `convSql`. Le nom de la table accédée en

base est «t_forum_frm» et on rappelle à l'objet SQLManagerX que le nom de notre instance est I_T_FORUM_FRM.



Concernant le nom de la table accédée suivant l'accès et l'OS il faut respecter la casse de la table (sous MySQL avec les tables innodb toujours utiliser les minuscules afin de pouvoir passer d'un système Unix à Windows et inversement)

6.6. Création de notre première fenêtre en mode Fiche

Création de la fenêtre

Créez une fenêtre vierge avec 4 champs

- un champ de saisie **Frm_Id**, de type numérique sur 4 octets
- un champ de saisie **Frm_nom**, de type texte sur 64 caractères
- un champ de saisie **Frm_sujet**, de type texte sur 255 caractères
- un champ de saisie **Frm_date_creation**, de type date

Enregistrez là dans le sous répertoire Fenêtre en tant que **Fiche FRM.wdw**

Ajoutez un bouton Fermer pour plus de facilité.

Vous devriez avoir quelque chose comme ça :



Insertion de la fonction d'affichage

Dans le bloc de Déclaration globales, saisissez les 2 lignes suivantes



```
Déclarations globales de Fiche FRM
I_T_FORUM_FRM:SQLPREMIER()
I_T_FORUM_FRM:SQLTABLEVERSÉCRAN("FICHE FRM")
```

On met le nom de la fenêtre courante en paramètre de la fonction (méthode) SQLTableVersEcran(). Ceci vient du fait que nous nous trouvons dans le code de déclarations globales. Si nous étions dans le code Initialisation, nous n'aurions pas ce problème mais nous vous conseillons d'utiliser toujours ce mode de programmation.

Pourquoi ? Une petite explication s'impose. Soit une fenêtre appelante et une fenêtre appelée. Dans le code Initialisation de la fenêtre appelée, la fonction FenEnCours() renvoie le nom de la fenêtre appelante ! Par défaut, c'est cette fonction qui est utilisée dans SQLTableVersEcran() si le nom de

la fenêtre n'est pas passé en paramètre. C'est pour cela que je conseille de toujours passer en paramètre le nom de la fenêtre.

Insertion des boutons mode parcours

Créez 4 boutons de parcours avec les codes associés

Premier : se positionner sur le premier enregistrement.



```
I_T_FORUM_FRM:SQLPREMIER()  
I_T_FORUM_FRM:SQLTABLEVERSECRAN("FICHE FRM")
```

Précédent : se positionner sur l'enregistrement précédent.



```
I_T_FORUM_FRM:SQLPRECEDENT()  
I_T_FORUM_FRM:SQLTABLEVERSECRAN("FICHE FRM")
```

Suivant : se positionner sur l'enregistrement suivant.



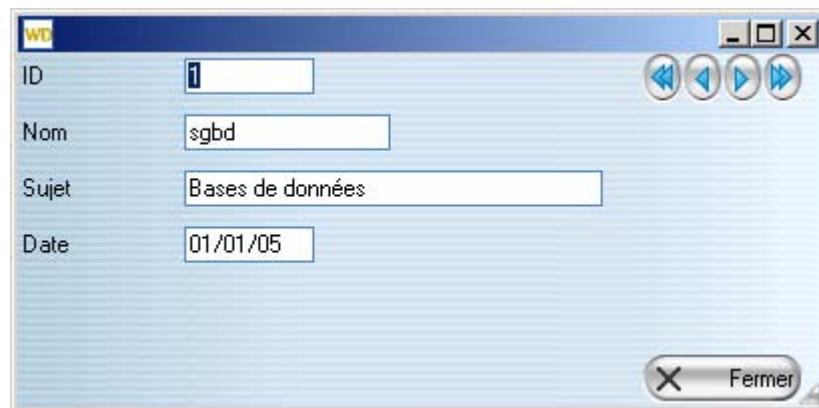
```
I_T_FORUM_FRM:SQLSUIVANT()  
I_T_FORUM_FRM:SQLTABLEVERSECRAN("FICHE FRM")
```

Dernier : se positionner sur le dernier enregistrement.



```
I_T_FORUM_FRM:SQLDERNIER()  
I_T_FORUM_FRM:SQLTABLEVERSECRAN("FICHE FRM")
```

Vous devriez maintenant avoir :



Lancer un GO du projet et tester le parcours de la table.

Insertion des boutons nouveau, modifier, supprimer

Créez 3 boutons avec les codes associés

Nouveau :



```

RETCODE EST UNE ENTIER
I_T_FORUM_FRM:SQLECRANVERSTABLE("FICHE FRM")
RETCODE = I_T_FORUM_FRM:SQLINSERT()
SI PAS RETCODE ALORS
    ERREUR(I_T_FORUM_FRM:ERREURSQL)
FIN

```



La méthode `SQLEcranVersTable` permet de transférer les données dans les champs de la fenêtre dans les membres de votre objet.

La méthode `SQLInsert` permet de transférer les valeurs des membres de votre objet dans la base de données.

Modifier :



```

RETCODE EST UNE ENTIER
I_T_FORUM_FRM:SQLECRANVERSTABLE("FICHE FRM")
RETCODE = I_T_FORUM_FRM:SQLUPDATE()
SI PAS RETCODE ALORS
    ERREUR(I_T_FORUM_FRM:ERREURSQL)
FIN

```



La méthode `SQLUpdate()` permet de transférer les valeurs des membres modifiés de votre objet dans la base de données. En effet, seules les valeurs modifiées sont transférées dans l'ordre update à la base de données.

Supprimer :

```

RETCODE EST UNE ENTIER

// DEMANDE DE CONFIRMATION
SI OUI NON(NON, "VOULEZ-VOUS VRAIMENT SUPPRIMER L'ENREGISTREMENT ?") ALORS
    // SUPPRESSION
    RETCODE=I_T_FORUM_FRM:SQLDELETE()
    SI PAS RETCODE ALORS
        ERREUR(I_T_FORUM_FRM:ERREURSQL)
    RETOUR
FIN
// LECTURE DE L'ENREGISTREMENT SUIVANT
I_T_FORUM_FRM:SQLSUIVANT()
// SI L'ENREGISTREMENT SUPPRIME ETAIT LE DERNIER ENREGISTREMENT
SI I_T_FORUM_FRM:ENDEHORS ALORS
    // LECTURE DU DERNIER ENREGISTREMENT
    I_T_FORUM_FRM:SQLDERNIER()
    // IL N'Y A PLUS D'ENREGISTREMENT DANS LE FICHIER
    SI I_T_FORUM_FRM:ENDEHORS ALORS
        // VIDE LES CHAMPS
        RAZ()
        I_T_FORUM_FRM:SQLRAZ()
        // INFORME L'UTILISATEUR QUE LE FICHIER EST VIDE
        INFO("LE FICHIER EST VIDE")
        // TERMINE
        RETOUR
    FIN
FIN
// TRANSFERT DU BUFFER DU FICHIER DANS LES CHAMPS
I_T_FORUM_FRM:SQLTABLEVERSECRAN("FICHE FRM")
FIN

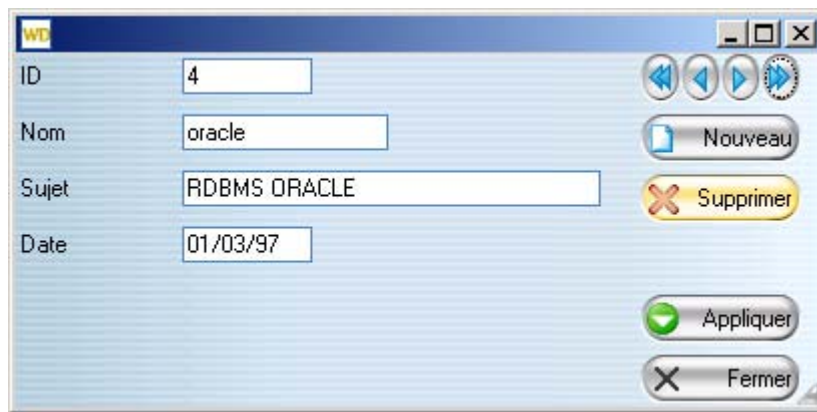
```



La méthode `SQLDelete()` supprime l'enregistrement en cours.

La méthode `SQLRaz()` va vider les tampons intermédiaires de l'objet.

Vous devriez finalement avoir :



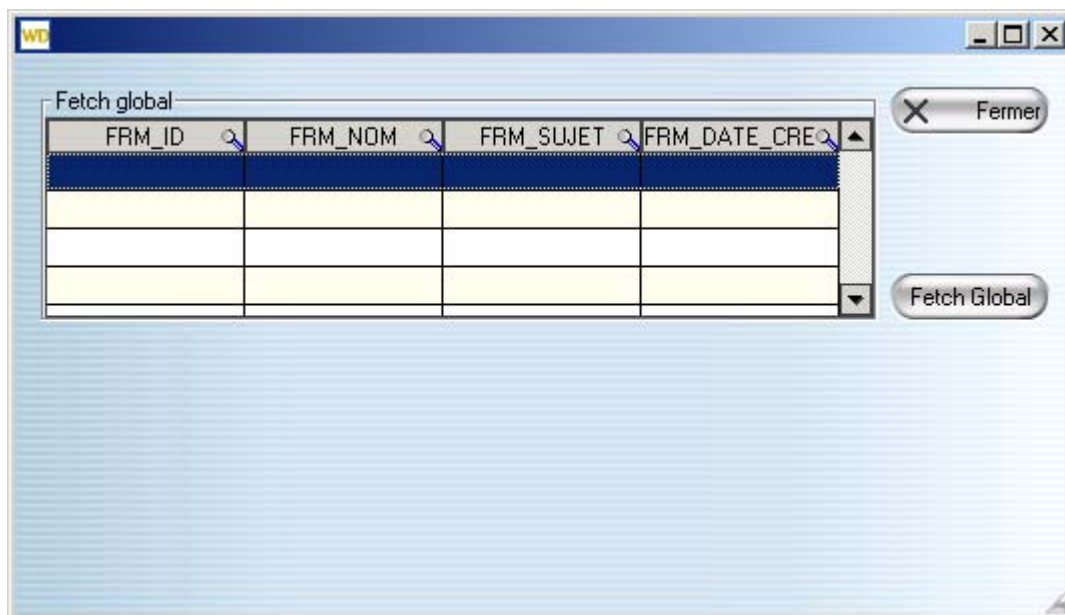
Lancer un GO du projet et tester les nouvelles actions proposées.

6.7. Création de notre première fenêtre en mode Table

Création de la table + alimentation en mode fetch⁶ global

1. Créez une fenêtre Vierge. Insérez un bouton Fermer avec le code associé.
2. Créez une table mémoire Table1 avec 4 colonnes (une colonne numérique, deux colonnes texte, une colonne date). Nous n'allons pas définir de code associé à cette table immédiatement.
3. Enregistrez là dans le sous répertoire Fenêtre en tant que **Table FRM.wdw**

Vous devriez avoir quelque chose comme ça :



1. Créez un bouton avec le code associé



```
I_T_FORUM_FRM:SQLFILTRE("T_FORUM_FRM.FRM_ID < 3","ORDER BY FRM_NOM")
I_T_FORUM_FRM:SQLCTABLE("TABLE1","FRM_ID, FRM_NOM, FRM_SUJET,
FRM_DATE_CREATION")
```



On remarquera dans notre exemple que nous avons appliqué un filtre pour ne prendre en compte que les forums dont l'identifiant est inférieur à 3. Le résultat sera ordonné sur le nom du forum par ordre croissant.

⁶ Lecture



Quand on emploie un nom de table dans un filtre, il faut s'assurer de la gestion de la casse par le SGBD (ex : pour MySQL t_forum_frm et différent de T_FORUM_FRM)

La méthode utilisée est très simple : SQLCTable. On lui passe en paramètre la table mémoire cible ainsi que les colonnes à afficher séparées pas des virgules.



*On doit nommer toutes les colonnes retournées par la requête. Le symbole * (généralement utilisé pour récupérer toutes les colonnes dans une requête SQL) n'est pas autorisé.*



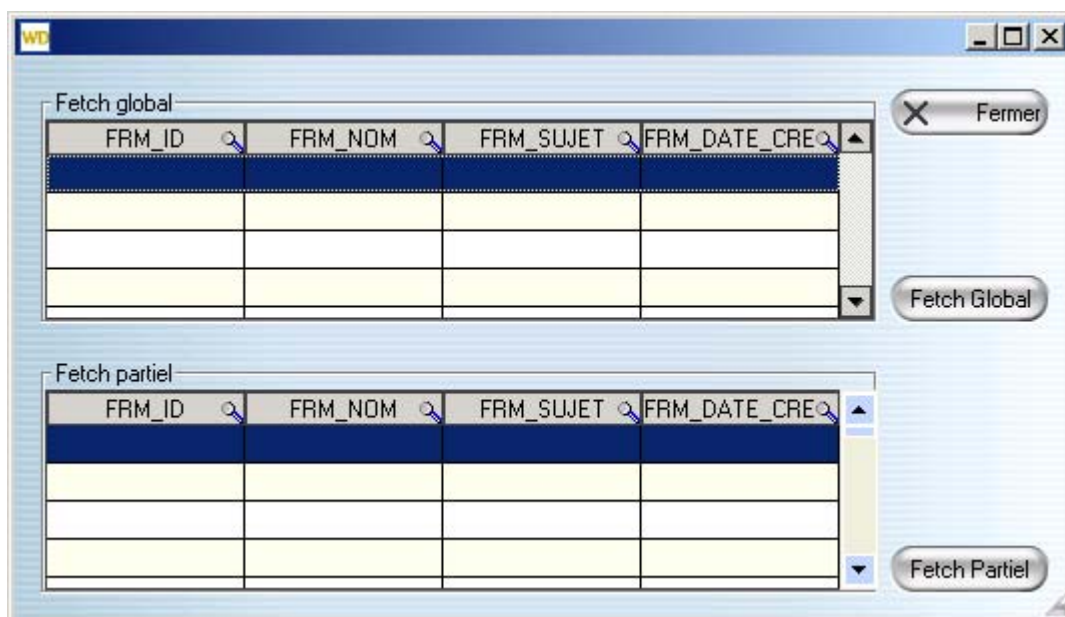
*L'intérêt de cette méthode au contraire de la suivante est l'utilisation des **loupes** et **tris** dans la table.*

Création de la table + alimentation en mode fetch partiel

L'inconvénient de la méthode précédente concerne son mode de fonctionnement intrinsèque : Toutes les lignes de la requête seront chargées en mémoire : le traitement peut être lent si la requête renvoie beaucoup de lignes. Pour palier à cela, SQLManagerX propose une méthode dite à fetch partiel. Le nombre de lignes ramené par la méthode est égal au nombre de lignes dans votre table.

1. Créez une table mémoire Table2 avec 4 colonnes (une colonne numérique, deux colonnes texte, une colonne date). Nous n'allons pas définir de code associé à cette table immédiatement.
2. Créez un ascenseur Ascenseur1. Mettez le un peu en forme pour qu'il ait la même hauteur que votre table et qu'il soit collé à elle à la droite de celle-ci. **La valeur minimale de l'ascenseur doit être 0.**

Vous devriez avoir maintenant :



3. Créez un bouton avec le code associé



```
I_T_FORUM_FRM:SQLFILTRE("T_FORUM_FRM.FRM_ID < 3","ORDER BY FRM_NOM")
I_T_FORUM_FRM:SQLXTABLE("TABLE2","ASCENSEUR1","FRM_ID, FRM_NOM, FRM_SUJET,
FRM_DATE_CREATION")
I_T_FORUM_FRM:SQLTABLEAFFICHE()
```

4. Ajoutez le code suivant à l'ascenseur



```
A chaque modification de Ascenseur1
I_T_FORUM_FRM:SQLTABLEAFFICHE()
```

5. Ajoutez le code suivant à notre table



```
Touche enfoncée (WM_KEYDOWN) de Table2
I_T_FORUM_FRM:XTableGereTouche(_EVE.WPARAM)
```

Explications : On remarque que le parcours initial ne diffère pas tellement de la méthode précédente hormis l'utilisation de la méthode TableAffiche.

Ensuite, on remarque l'utilisation de cette méthode à chaque modification de valeur de notre ascenseur (Ascenseur1) : normal car c'est le fonctionnement normal d'un ascenseur dans une table, celui-ci permet le parcours dans la table.

Pour finir la méthode XtableGereTouche qui nous permet de mettre à jour la valeur courante de notre ascenseur en fonction des touches appuyées (haut, bas, 'page up', 'page down') puis réaffiche la table.

A compléter...

6.8. Les éléments utiles

La gestion des filtres. Quelle méthode utiliser ?

Nous avons vu que nous pouvions déclarer un filtre tout simplement à l'aide de la méthode SQLFiltre. Le premier paramètre correspond à la future clause de restriction sans le mot clé WHERE, le second correspond à l'ordre dans lequel les éléments doivent être lus (le ORDER BY).

Une annulation de filtre est effectuée au moyen de la méthode SQLFiltre("", "").



Pour une annulation, on pourrait tout simplement utiliser la méthode sans aucun paramètre : SQLFiltre().

La méthode DesactiveFiltre() désactive temporairement le filtre créé précédemment. La méthode ActiveFiltre() quant à elle le réactive.

La gestion des vues ?

Nous pouvons gérer les vues (au sens HyperFile). L'intérêt de la vue est de diminuer la taille du résultat renvoyé, ce qui va augmenter la vitesse de réponse de l'application. La diminution se traduit par un périmètre restreint des colonnes récupérées par l'ordre de sélection en base.



```
I_T_FORUM_FRM:SQLCREEVUE("FRM_ID,FRM_NOM",("T_FORUM_FRM.FRM_ID < 3",  
"ORDER BY FRM_NOM")
```

On remarquera dans notre exemple que nous avons créé une vue avec un filtre pour ne prendre en compte que les forums dont l'identifiant est inférieur à 3. Le résultat sera ordonné sur le nom du forum par ordre croissant, et contiendra uniquement les colonnes ID et Nom.



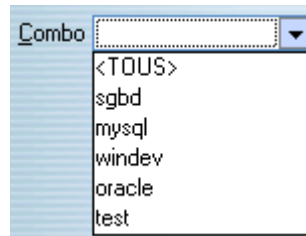
POUR UNE ANNULATION, ON POURRAIT TOUT SIMPLEMENT UTILISER LA METHODE : SQLDETRUITVUE().

Utilisation d'une combo



```
I_T_FORUM_FRM:SQLCHARGECOMBO("COMBO1","SELECT FRM_ID, FRM_NOM FROM  
T_FORUM_FRM", "<TOUS>")
```

Exemple de résultat :



7. Les impressions avec SQLManagerX

En abordant ce chapitre, nous supposons que le lecteur a les notions nécessaires à sa compréhension.

7.1. *Impression Mode Fiche*

TODO

7.2. *Impression rapide Mode Table*

Comme vous avez pu le constater, le package SQLManagerX fourni deux états génériques R_SQLetat, R_SQLetatP. Leur utilisation est très simple : on affiche les données dans la table mémoire. Si on lance une impression, SQLManagerX récupère l'ordre de sélection et le réutilise pour imprimer l'état.

Créez un bouton imprimer avec le code suivant :



```
IAPERÇU(100)
// PORTRAIT
I_T_FORUM_FRM:SQLTABLEEDIT("LISTE DES FORUMS")
// PAYSAGE
I_T_FORUM_FRM:SQLTABLEEDIT("LISTE DES FORUMS","PAYSAGE")
```

Lancer un GO de la fenêtre et tester les deux impressions possibles

7.3. *Impression complète*

Dans cette partie, nous souhaitons imprimer une liste mais en gérant totalement le parcours en base et les ordres d'impression.

TODO

8. Aller plus loin avec SQLManagerX

8.1. Construction automatique d'une fenêtre Table

La version 4 de SQLManagerX s'est vue complétée par une méthode de construction dynamique d'une table mémoire. **Une seule ligne de code est nécessaire, SQLManagerX fait le reste !**

Création de la table + alimentation en automatique

1. Créez une fenêtre Vierge. Insérez un bouton Fermer avec le code associé.
2. Créez une table mémoire Table1 avec une seule colonne. Nous n'allons pas définir de code associé à cette table immédiatement.
3. Enregistrez là dans le sous répertoire Fenêtre en tant que **Table FRM auto.wdw**
4. Ajoutez le code suivant à notre table



```
Touche enfoncée (WM_KEYDOWN) de Table1
I_T_FORUM_FRM:XTableGereTouche(_EVE.wparam)
```

Souvenez-vous, c'est ce traitement qui nous permet de gérer les touches lorsque la table est alimentée par la méthode SQLXtable().

5. Créez un ascenseur Ascenseur1. Mettez le un peu en forme pour qu'il ait la même hauteur que votre table et qu'il soit collé à elle à la droite de celle-ci. **La valeur minimale de l'ascenseur doit être 0.**
6. Ajoutez le code suivant à notre ascenseur



```
A chaque modification de Ascenseur1
I_T_FORUM_FRM:SQLTABLEAFFICHE()
```

7. Créez un bouton afficher avec le code suivant :



```
XL_RET EST UN BOOLEEN
I_T_FORUM_FRM:SQLFILTRE("")
XL_RET=I_T_FORUM_FRM:SQLTABLECONSTRUIT("TABLE1","FRM_ID,FRM_NOM,FRM_SUJET,FR
M_DATE_CREATION","ASCENSEUR1",FAUX,VRAI,VRAI,FAUX)
SI XL_RET ALORS
    TABLE1..LIBELLE="DETAIL TABLE "
    ASCENSEUR1=0
    I_T_FORUM_FRM:SQLTABLEAFFICHE()
SINON
    INFO("CONTROLLER LES COLONNES", "AU MOINS UNE COLONNE N'EXISTE PAS")
FIN
```

Lancer un GO de la fenêtre et tester cet affichage entièrement automatisé. Nous vous laissons le soin de modifier les quatre derniers paramètres pour tester les impacts sur le fonctionnement.

8.2. Gestion automatisée d'une fenêtre Fiche

La version 5 de SQLManagerX sera complétée par une méthode de gestion automatisée d'une fenêtre Fiche. **Une seule ligne de code dans le code d'initialisation de la fenêtre est nécessaire, SQLManagerX fait le reste !**



Il faut simplement respecter le nom des boutons employés afin qu'ils soient en conformité avec ceux attendus dans la méthode.



Cette version présentée aux dernières Windreveries a remportée un franc succès auprès de l'auditoire. Elle est actuellement en version bêta auprès des personnes nous ayant demandé de l'utiliser malgré nos recommandations d'usage (nos tests internes étant en cours). Dès que la version sera publiée, nous compléterons ce point.

TODO

8.3. La gestion des LOBs (Large Object) avec SQLManagerX

Vous connaissez tous les Mémos sous HyperFile, ce type de données s'appelle plus communément LOB (Large Object). Il existe 2 types d'objets larges : **CLOB** – Character Large Object et **BLOB** Binary Large Object.

Nous allons voir comment SQLManagerX gère ce type de données bien particulier.

Insérer un BLOB

Avec cet exemple nous allons insérer une image, un fichier, ou tout autre élément binaire dans la table « prodimages ». Cette table est composée de deux colonnes : **id** l'identifiant de ligne, **image** la colonne contenant le BLOB.



```
I_T_PRODIMAGES:M_ID = 1
I_T_PRODIMAGES:SQLATTACHEMEMO("IMAGE","C:\TEST.JPG")
I_T_PRODIMAGES:SQLINSERT()
```



SQLAttacheMemo attend deux paramètres : le premier reprend le nom de la colonne contenant le BLOB, le second le chemin physique du fichier à prendre en compte.

Lecture d'un BLOB

Après avoir insérer un enregistrement, nous allons maintenant le lire.



```
I_T_PRODIMAGES:SQLFILTRE("ID=1")
I_T_PRODIMAGES:SQLPREMIER()
I_T_PRODIMAGES:SQLECRANVERSTABLE()
I_T_PRODIMAGES:SQLCHARGE MEMO("CHAMP_IMAGE","IMAGE")
```



SQLChargeMemo attend deux paramètres : le premier reprend le nom du champ dans la fenêtre le second la colonne contenant le BLOB.

8.4. Accès multi-bases avec SQLManagerX

Puis je faire un source unique, un fichier exécutable unique, fonctionnant sur différentes bases ?
Oui !

L'un des intérêts majeurs de SQLManagerX est d'être multi-bases au niveau du mode de programmation. **C'est l'avantage coté développeur : un seul code, plusieurs bases accessibles.**

Avec la méthode ci-dessous, nous allons étendre cet avantage de multi-bases **coté utilisateur** : Plus de versions multiples de votre application en fonction du client et de sa base de données ! **Une seule application, plusieurs bases accessibles.**

Grâce aux possibilités de WinDev concernant les instanciations dynamiques d'objet, nous vous proposons ici une méthode qui permet de choisir la base de données accédée lors de l'exécution de votre application à l'aide d'un paramétrage (ici c'est un fichier .ini mais ce peut être aussi une variable stockée dans la base de registre).

Création d'un fichier de paramétrage

4. Créer un fichier **programme.ini** et qui est localisé dans le répertoire d'exécution de l'application
5. Insérer les lignes suivantes :



```
[BASE]
RDBMS=MYSQL
```

RDBMS correspond au SGBDR

Code à ajouter à l'initialisation du projet

Modifier la ligne suivante dans le code d'initialisation de votre projet :

```
GLOBAL
convSql est c_mySQL4WD
```

Par le code suivant :

```
Code à ajouter à l'initialisation du projet
// Choix de la base
convSql est un objet dynamique
gXP_ACCES est une chaîne

gXP_ACCES = INILit("BASE", "RDBMS", "", fRepEnCours()+"\programme.ini")

SELON Majuscule(gXP_ACCES)
CAS "MYSQL" :
    convSql = allouer c_mySQL4WD()
CAS "PGSQL" :
    convSql = allouer c_postgreSQL4WD()
CAS "PHP" :
    convSql = allouer c_PHP4WD()
CAS "ORACLE" :
    convSql = allouer c_oracle4WD()
AUTRE CAS :
    FinProgramme("Accès pour "+gXP_ACCES,"n'est pas permis.")
FIN
```



Pendant la phase de développement utiliser l'ordre « convSql est un votre_accès ». En effet l'allocation dynamique ne permet pas d'utiliser la complétion pour le code (proposition des méthodes).

8.5. Fonctionnement de PHP4WD

Le but de cet accès est **d'accéder à un serveur MySQL chez un hébergeur public sur Internet au travers du langage PHP**. L'idée a été lancée en 2002 sur le site wdforge (rbesset à l'époque). Notre équipe a développé ce concept simple de prime abord mais demandant des connaissances PHP en sus.

Rappels du besoin

Les serveurs MySQL des hébergeurs publics ne sont évidemment pas disponibles directement, par soucis de sécurité ceux-ci bloquent le port standard 3306. Ces hébergeurs proposent souvent le couple PHP/Apache. La page PHP est elle capable d'accéder au serveur de donnée (HTTP → PHP → MySQL).

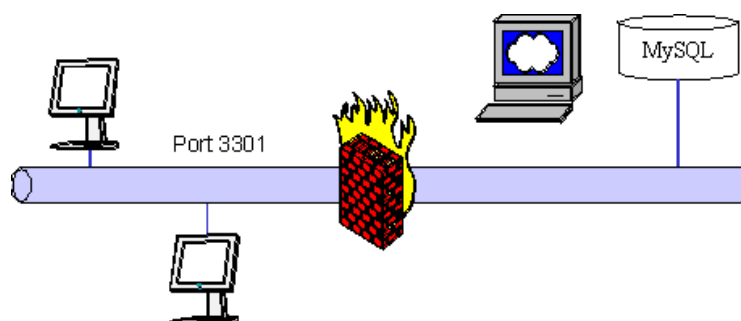


Figure 3: Constatation avec un serveur MySQL sur le port 3301

Pourquoi ne pas alors utiliser un script PHP qui reçoit en paramètre la requête SQL à exécuter sur le serveur de donnée et retourne le résultat ? Le PHP s'occupe de la connexion avec le serveur de donnée, exécute la requête et peut même proposer un retour en HTML.

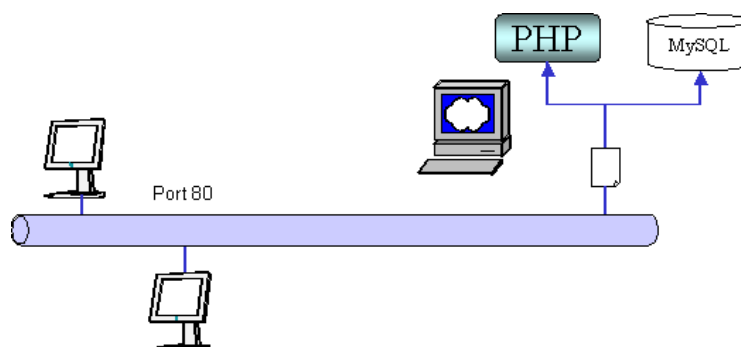


Figure 4: Principe de contournement

Architecture proposée

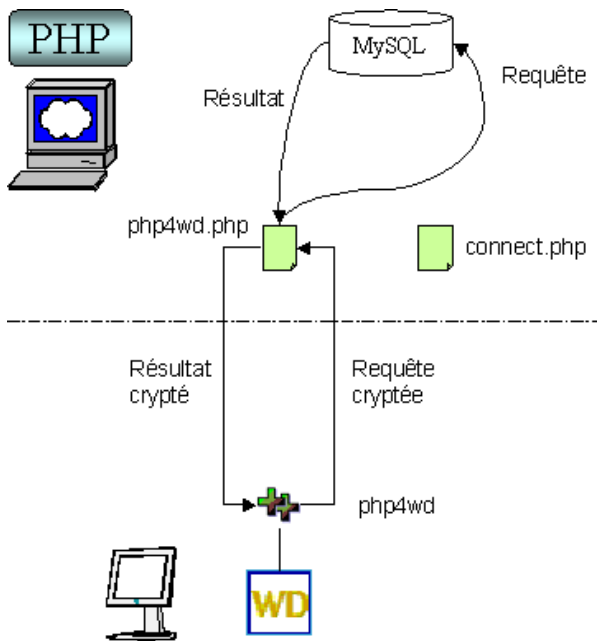


Figure 5: Principe mis en œuvre

La classe php4wd envoie la requête cryptée par une clé publique et privée au script PHP se trouvant sur le site Web (chez votre hébergeur).

Le script php4wd.php exécute la requête, récupère le résultat, crypte celui-ci et le transmet à la classe par HTTP.

La classe lit le résultat et conserve celui-ci en mémoire (ce que font les DLL des accès natifs). Ensuite la classe php4Wd exploite ce résultat au même titre qu'un autre accès.

La clé publique est celle qui sert pour l'élaboration de la clé privée, qui permet le cryptage et le décryptage.



Seule la dépose des deux fichiers PHP est nécessaire dans la mise en œuvre. Il suffit de modifier les paramètres de connexion.

On peut donc venir interfacier une application WinDev sur un site WEB existant !



Attention, utilisez vos propres clés. Modifiez les clés dans les fichiers pour vous assurer un chiffrement personnel.

Un petit exemple

```

monAccess is c_Php4WD()

v_retCode = MonAccess:mysqlConnecte("http://monsite.com/php4wd.php")
IF (v_retCode = false) THEN
  Erreur("Impossible de se connecter à la base de donnée 'test'.", ...
  "Erreur n° " + monAccess:mysqlErreur, monAccess:mysqlGetErrorMessage())
END
retCode = monAccess:mysqlExec("SELECT * FROM client", 0)
IF (retCode=1) THEN
  monAccess:mysqlPremier(0)
  WHILE (NOT monAccess:mysqlEnDehors)
    Trace( monAccess:mysqlCol(0,1) + "/" + monAccess:mysqlCol(0,2) )
    monAccess:mysqlSuisvant(0)
  END
ELSE
  Erreur("Erreur n° " + monAccess:mysqlErreur, ...
  monAccess:mysqlGetErrorMessage())
END
monAccess:mysqlFerme(0)

```



Etant donné que la classe communique avec la page PHP, et que c'est la page PHP qui communique avec la base de données, on remarque que toute base pouvant être accédée par le PHP le sera aussi par la classe.

8.6. Exemple avec WDScript

TODO

9. FAQ (Foire aux questions)

9.1. ***Je ne trouve pas de documentation en anglais, est-ce normal ? (I do not find English documentation, is this normal ?)***

Le fait de maintenir une documentation en deux langues est très compliqué. Nous sommes à la recherche de personnes anglophones pouvant nous aider dans cette traduction de documentation papier. Concernant la traduction du site, l'outil *babelfish* permet une traduction à la volée.

9.2. ***On parle de SQLManagerX, on parle des accès [alter]natifs mais peut-on utiliser l'un sans l'autre ?***

SQLManagerX nécessite l'utilisation des accès [alter]natifs. En effet, la classe SQLManagerX se base sur les méthodes des accès. Par contre si vous n'avez pas d'utilité pour SQLManagerX, vous pouvez utiliser directement les accès [alter]natifs.

9.3. ***Puis-je utiliser les accès (sous licence WD-Libre) dans un soft commercial ? Dois-je rétribuer l'auteur ?***

Oui, vous pouvez intégrer les différentes ressources autour de SQLManagerX dans un produit même à finalité commerciale. En aucun cas une redevance ne vous sera exigée. La seule obligation « morale » concerne un remerciement au travers de la fenêtre « A propos de » dans votre application.

9.4. ***Quand utiliser SQLManagerX, quand utiliser les accès [alter]natifs directement ?***

Ce qui est généralement mis en œuvre par les développeurs utilisateurs de notre solution :

Utilisation des accès alter[natif] pour :

- Tout ce qui est liste, fenêtre en mode Table non basée sur une seule table en base
- Préparation d'impression d'un état (source de données).
- Algorithmique compliquée

Utilisation de la classe SQLManagerX pour :

Toutes les créations / modifications en mode Fiche ou Table

9.5. **Concernant les DLL, où vaut-il mieux les placer ?**

Lorsque l'on installe une application, on se demande où placer les DLL. Deux possibilités s'offrent à nous : dans le répertoire exe de l'application, dans le répertoire system32 du poste client :

- L'avantage dans system32 est une non multiplication des DLL sur le poste, l'inconvénient est que toutes les applications nécessitant cette DLL vont utiliser cette DLL unique. Tout va bien si toutes les applications impactées utilisent bien la même version de la DLL sinon de gros problèmes apparaissent (la majorité des cas).
- L'avantage dans le répertoire exe est que l'application fonctionnera toujours vu son indépendance vis-à-vis des autres. L'inconvénient comme vous vous en doutez à la multiplication des DLL.

9.6. **La grande question : quelle base pour mon projet ?**

Plutôt que de reprendre une analyse déjà effectuée, je vous renvoie sur une page de developpez.net : [comparatif réalisé par developpez.com](http://fadace.developpez.com/sbdbcmp) (<http://fadace.developpez.com/sbdbcmp>) sur les différents SGBD du marché.

9.7. **Pourquoi utilisez-vous des tables mémoires ?**

Tout simplement parce que les tables fichier sont directement liées sur un fichier ou une requête HyperFile. Nous ne pouvons donc pas utiliser cette fonctionnalité.

9.8. **J'utilise régulièrement WDMMap, maintenant je fais comment ?**

Comme HyperFile, tout éditeur de SGBD propose une application permettant de consulter et piloter les données contenues en base.

En plus de cela, il existe des **produits commerciaux** (TOAD pour Oracle, IB-Expert pour FireBird,...), des **programmes en open-source** (TORA pour Oracle, Sqlyog pour MySQL,...) ou bien encore des outils au travers de page **web** (le plus connu phpMyAdmin pour MySQL, IBWebAdmin pour FireBird, SQLiteManager pour SQLite, phpPgAdmin pour PostgreSQL,...).

9.9. Pourquoi toutes les méthodes disponibles commencent-elles par MySQL ?

C'est tout simplement lié à un historique. MySQL4WD a été le premier accès [alter]natif créé juste avant SQLManagerX. Par un souci de compatibilité de nommage des méthodes entre les différents accès (surtout au niveau SQLManagerX), les méthodes ont toutes le même nom de méthode au niveau de la classe. Sinon la légende veut aussi que Rodolphe soit un peu possessif et qu'il ait débuté ses méthodes par MySQL pour « MonSQL ».

9.10. Existe-t-il une version WinDev5.5 des accès natifs ? de SQLManagerX ?

Non. Les accès sont développés en version WinDev 7.5. Pour les versions ascendantes (WinDev 8, 9 et 10) une simple recompilation suffit. La méthode pour WinDev55 consiste à demander sur le forum une version texte du code source de la classe WinDev 7.5. Comme aucune spécificité WinDev 7.5 n'est utilisée (excepté le typage des objets WinDev), le code est généralement compatible à 100%.

9.11. Quels sont les numéros de requêtes permis ?

SQLManagerX utilise généralement le numéro de requête 0 (et également 1 dans la méthode SQLEdit). Un nombre limité d'identifiant (généralement 5) de requêtes simultanées vous est proposé.

9.12. Accès natifs : Pourquoi dois-je faire MysqlFerme(X) ?

A chaque exécution d'une requête, vous lui affectez un identifiant (Cet identifiant est transparent avec SQLManagerX). Afin de pouvoir réutiliser cet identifiant, il est nécessaire de fermer la requête précédemment ouverte afin de libérer les ressources allouées en mémoire pour celle-ci. Dans les accès [alter]natifs, cela est fait de manière explicite par le développeur avec la commande MysqlFerme(X).

9.13. Accès natifs : Y-a-t-il compatibilité ascendante entre les versions successives des accès natifs ?

Dans la mesure du possible oui. Seule exception est faite si l'API de l'éditeur change et nous oblige à ajouter dans paramètre dans les méthodes existantes. Dans tous les cas, une mention spéciale vous est communiquée lors de la mise à disposition de l'accès.

9.14. SQLManagerX : SQLManagerX est-il compatible WebDev ?

Oui. Au moins deux utilisateurs utilisent des accès [alter]natifs au travers de WebDev en environnement de production avec des temps de traitement plus que satisfaisants.

9.15. MySQL4WD : Puis-je me connecter sur 2 serveurs différents ?

Depuis la version 2.0.0 l'accès gère la connexion sur plusieurs bases simultanément.

9.16. MySQL4WD : J'ai un message à l'exécution de mon projet "run out of sync" ?

Réponse de Rodolphe Jouannet sur la ML le 06 Octobre 2005 : « La commande 'Out of sync' est générée si on fait un mySQLLitCol(), un mySQLSuivant(),... sans avoir fait de mySQLPremier(). La même erreur est constatée lorsqu'on fait un mySQLPremier() sans de mySQLExec(). Vérifier l'ordre d'exécution de votre code dans l'état. »

9.17. Oracle4WD : Au lancement de mon application j'ai le message «Impossible de charger la DLL : pas présente ou défectueuse » ?

Cette anomalie peut provenir du fait de la non présence de la DLL Oracle4WD dans le répertoire d'exécution ou dans le PATH Windows. Après vérification, si vous constatez sa présence, le problème peut venir de la présence de deux clients oracle sur le poste.

9.18. Accès natifs : Pourquoi certains des accès natifs génèrent des fichiers .fic?

La majorité des SGBD ne gèrent pas les ordres « précédent » ou « dernier ». Nous devons donc simuler ces ordres assurer la compatibilité de SQLManagerX avec le fonctionnement à la HyperFile. Le meilleur moyen est de gérer un fichier HyperFile en local pour chaque requête. On peut ainsi décrocher le parcours sur ce fichier en cas d'ordre « précédent ». Si vous utilisez les accès sans SQLManagerX, vous pouvez désactiver ce mode de fonctionnement à l'aide de la commande `mySQLSetMode(2)`. Ces fichiers sont gérés par une déclaration dynamique. Aucune analyse n'est nécessaire.

9.19. SQLManagerX : En regardant votre code, je me rends compte que vous ne gérez que 4 types de données. Comment faites vous ?

SQLManagerX ne gère que les types *numérique*, *text*, *datetime* et *BLOB*. Les données étant stockées dans des zones tampon de type chaîne de caractères, la différence entre un entier et un monétaire n'apparaît qu'à l'affichage ou dans votre programme.

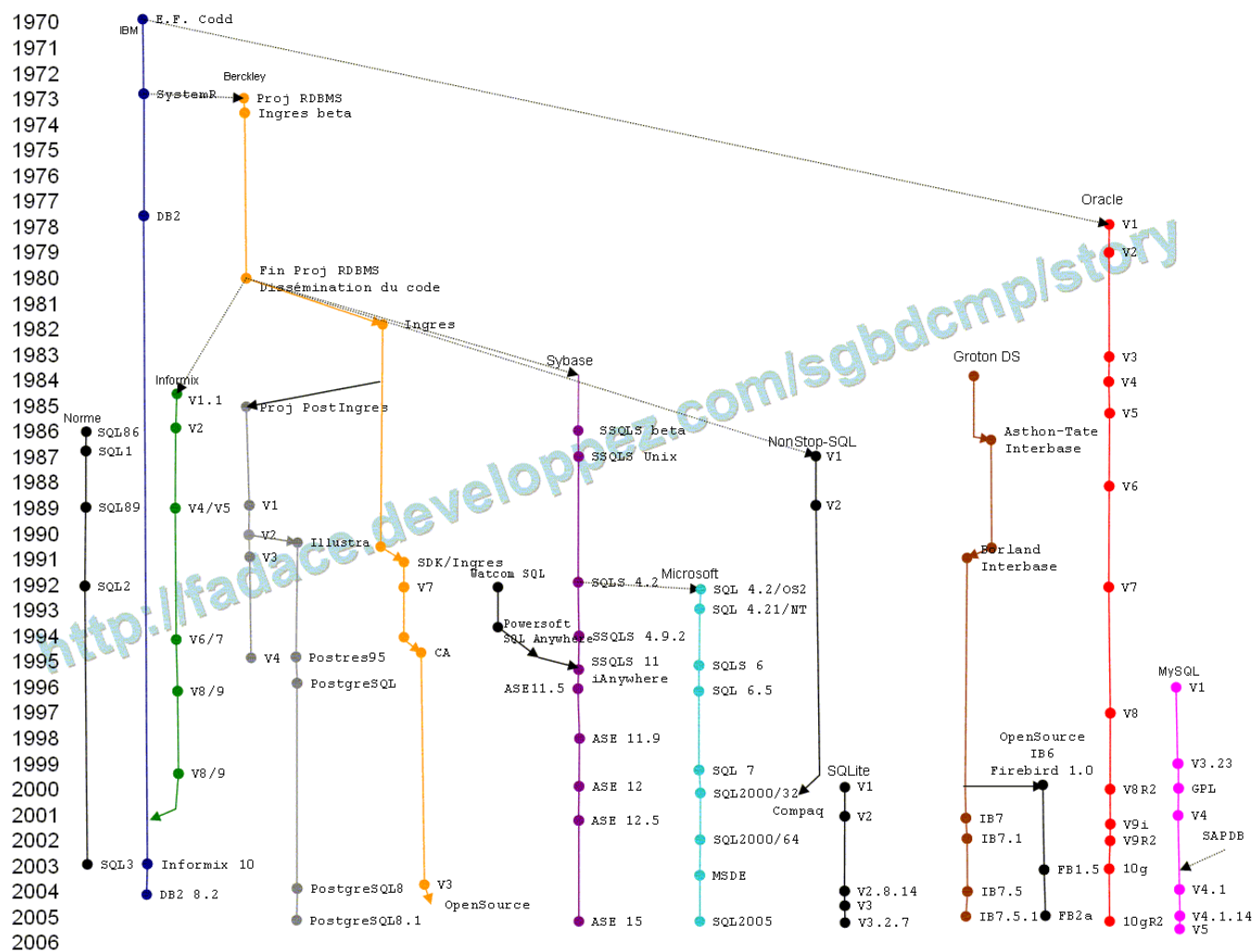
9.20. SQLManagerX : Comment assurer vous le développement et la non régression?

Avec la version 4 de SQLManagerX, les accès se sont vus dotés d'une nouvelle méthode transverse : `TestAll()`. Elle comprend un jeu d'essai type des fonctionnalités minimales que doit posséder l'accès pour être compatible. Si la trace fournie à l'issue de ce test est identique à la trace référence, l'évolution est validée.

10. Divers

10.1. Généalogie des SGBDR

Le graphique suivant reprend l'historique des différents SGBD. Vous pouvez trouver la dernière version sur [la page de fadace](http://fadace.developpeur.com/sgbdcmp/story).



On comprend mieux pourquoi les bases SQL-Serveur et les bases Sybase sont toutes deux compatibles avec la dlib... On comprend mieux d'où vient le nom de PostGreSQL.

10.2. Quel accès pour ma base ?

Le tableau suivant reprend les principaux SGBD du marché.

	API	ODBC4WD ⁷	ADO4WD	FB4WD	Oracle4WD	MySQL4WD	PG4WD	Php4WD ⁸	OTL4WD	SQLite4WD
AS400	<i>Non disponible</i>									
Borland Interbase Borland Firebird		+	+	+++				+++ (distant)		
Filemaker	<i>Non disponible</i>	+	+							
IBM Informix	<i>Non disponible</i>	+	+							
IBM DB2/UDB	<i>(DB2 CLI)</i>	+	+						<i>En cours</i>	
Microsoft SQL-Serveur	<i>ADO</i>	+	+++							
Microsoft MSDE	<i>ADO</i>	+	+							
Microsoft Access	<i>ADO</i>	+	+							
MySQL	<i>(MySQL C API)</i>	+	+			+++		+++ (distant)		
Oracle	<i>(OCI)</i>	+	+		+++			+++ (distant)	+++	
Pervasive	<i>ADO</i>	+	+							
PostgreSQL	<i>(libpq)</i>	+	+				+++	+++ (distant)		
Progress	<i>Non disponible</i>	+								
Sybase ASE, ASA, ASIQ	<i>Non disponible</i>	+	+							
SQLite										+++
XBase	<i>Non disponible</i>	+								

⁷ Si un accès ODBC existe pour cet éditeur de SGBD

⁸ L'accès se base sur les connecteurs disponibles par PHP. Implicitement toute base accessible via le langage PHP et accessible au travers de l'accès php4wd.

10.3. Les équivalences de types⁹

	WinDev	FireBird	Oracle	PostGreSQL	ADO	MySQL
0	<i>hRubInvalide</i>	Invalide	VARCHAR(4000)	VARCHAR2(4000)	TEXT	TEXT
1	<i>hRubIdAuto</i>	Identifiant automatique (8 octets)	INTEGER	NUMBER(20)	BIGSERIAL	INTEGER
2	<i>hRubTexte</i>	Texte	CHAR(Taille) BLOB	VARCHAR2(Taille)	CHAR(Taille) TEXT	CHAR(Taille) LONGTEXT
3	<i>hRubEntier2</i>	Entier sur 2 octets	INTEGER	NUMBER(3)	SMALLINT	SMALLINT
4	<i>hRubEntier1</i>	Entier sur 1 octet	SMALLINT	NUMBER(5)	INTEGER	INT
5	<i>hRubEntier4</i>	Entier sur 4 octets	INTEGER	NUMBER(10)	BIGINT	INTEGER
6	<i>hRubRéal4</i>	Réal sur 4 octets	FLOAT	NUMBER(15,3)	FLOAT	FLOAT
7	<i>hRubRéal8</i>	Réal sur 8 octets	DOUBLE PRECISION	NUMBER(15,3)	DOUBLE PRECISION	FLOAT
8	<i>hRubNumEnr</i>	Numéro d'enregistrement	INTEGER	NUMBER(4)	INTEGER	INTEGER
9	<i>hRubEntierNonSigné2</i>	Entier non signé sur 2 octets	INTEGER	NUMBER(4)	INTEGER	INTEGER
10	<i>hRubDate6</i>	Date	typeDate	TypeDate	TypeDate	typeDate
11	<i>hRubHeure</i>	Heure	typeHeure	TypeDate	typeHeure	typeHeure
12	<i>hRubEntierNonSigné1</i>	Entier non signé sur 1 octet	INTEGER	NUMBER(2)	INTEGER	INTEGER
13	<i>hRubRéalTurbo</i>	Réal turbo	FLOAT	NUMBER(15,3)	REAL	REAL
14	<i>hRubDate8</i>	Date	typeDate	TypeDate	TypeDate	typeDate
15	<i>hRubMémoTexte</i>	Mémo texte	BLOB	VARCHAR2(2000)	TEXT	TEXT
16	<i>hRubMémoBinaire4</i>	Mémo binaire (4 octets)	BLOB	LONG	BYTEA	IMAGE
17	<i>hRubMonétaire</i>	Monétaire	DOUBLE PRECISION	NUMBER(20,3)	DOUBLE PRECISION	FLOAT
18	<i>hRubMémoBinaire</i>	Mémo binaire	BLOB	LONG RAW	BYTEA	IMAGE
19	<i>hRubEntier8</i>	Entier sur 8 octets	INTEGER	NUMBER(20)	BIGINT	INTEGER
20	<i>hRubEntierNonSigné8</i>	Entier non signé sur 8 octets	INTEGER	NUMBER(20)	BIGINT	INTEGER
21	<i>hRubImage</i>	Image	BLOB	LONG RAW	BYTEA	IMAGE
22	<i>hRubEntierNonSigné4</i>	Entier non signé sur 4 octets	INTEGER	NUMBER(20)	BIGINT	INTEGER
23	<i>hRubBinaire</i>	Binaire	BLOB	LONG RAW	BYTEA	IMAGE
24	<i>hRubDateHeure</i>	Date/Heure	VARCHAR(14)	TypeDate	TEXT	TEXT
25	<i>hRubDurée</i>	Durée	CHAR(Taille)	VARCHAR2(taille)	CHAR(Taille) TEXT	CHAR(Taille) TEXT
26	<i>hRubCaractère</i>	Caractère	CHAR(Taille)	VARCHAR2(taille)	CHAR(Taille) TEXT	CHAR(Taille) TEXT
27	<i>hRubBooléen</i>	Booléen	SMALLINT	NUMBER(4)	SMALLINT	SMALLINT
28	<i>hRubIdAuto4</i>	Identifiant automatique (4 octets)	INTEGER	NUMBER(20)	BIGSERIAL	INTEGER

⁹ Ces équivalences sont utilisées dans l'outil SQLManagerX Converter. Elles sont données à titre indicatif. Vous pouvez les modifier en personnalisant le fichier contenant ces équivalences.

10.4. Les accès natifs de l'éditeur

L'éditeur fournit un ensemble d'accès natifs. Ils sont peut-être **payants** mais au contraire des accès [alter]natifs, ils sont **totalemtent intégrés** à l'environnement de développement : les outils tels que le visualisateur de fichiers WDMAP, le RAD, le WDETAT sont compatibles à 100%. Côté programmation, vous n'avez pas de classe à gérer, de code supplémentaire à produire : votre technique de programmation peut rester identique à celle pour accéder à HyperFile (utilisation des ordres HLit*).

Liste des accès disponibles :

- **AS/400 & iSeries**
- **DB2** (à partir de la version 7.1)
- **Informix** (à partir de la version 7)
- **MySQL** (à partir de la version 3.23.52)
- **Oracle** (à partir de la version 7)
- **Progress** (à partir de la version 7.3 C)
- **SQL Server** (à partir de la version 6)
- **Sybase** (ASE, à partir de la version 10)
- **xBase** (dBase 3+, dBase 4, Clipper5, Clipper 87, FoxBase)

10.5. Liens et références

Site référence : www.sqlmanagerx.com

Site de l'éditeur : <http://www.pcsoft.fr>

Site de l'association : <http://www.windasso.org>

Site partenaire généraliste : www.wdforge.org

Site partenaire spécifique WebDev : <http://www.wtablettes.net>

Site contenant des informations générales sur les SGBD : <http://fadace.developpez.com>

Site contenant des informations générales sur les SGBD et plus particulièrement le langage SQL : <http://sqlpro.developpez.com>

ADO4WD se base sur le travail de *Carlos Antolini* disponible sur la page <http://www.codeproject.com/database/aaaadoclass1.asp>

OTL4WD se base sur le travail de *Sergei Kuchin* disponible sur la page <http://otl.sourceforge.net>

FB4WD se base sur le travail de *Olivier Mascia* disponible sur la page <http://www.ibpp.org>

11. Index

A

Access, 7, 9, 42
 ADO, 7, 17, 42, 43, 44
 alter[natif], 36
 API, 7, 9, 17, 39, 42
 AS/400, 2, 9, 45
 avantage, 13, 31, 37

B

BLOB, 3, 30, 31, 40, 44

C

code unique, 13
 Connexion, 11

D

DAO, 7
 DB2, 19, 42, 45
 DLL, 3, 5, 9, 10, 11, 14, 17, 34, 37, 40

F

FireBird, 17, 37, 44

G

Généalogie, 3, 41

H

HyperFile, 2, 4, 7, 8, 9, 13, 14, 15, 27, 30, 37, 40, 45

I

Informix, 42, 45
 InterBase, 17

M

Mémo, 44
 MySQL, 3, 9, 13, 17, 18, 20, 25, 32, 37, 38, 42, 44, 45
 MySQLConnecte, 11, 18, 35
 mySQLExec, 12, 35, 39
 MySQLGetErrorMessage, 11, 12, 18, 35
 MySQLSetMode, 40
 MySQLTable, 12

O

ODBC, 2, 7, 8, 9, 42
 OLE DB, 2, 7, 8, 9
 Oracle, 2, 9, 11, 13, 17, 37, 42, 44, 45

P

Pervasive, 43
 PHP, 18, 32, 33, 34, 35, 42
 Postgresql, 17, 18
 Progress, 43, 45

R

RAD, 9, 10, 15, 45

S

SGBD, 2, 4, 5, 7, 9, 10, 14, 25, 37, 40, 41, 42, 46
 SQL Server, 2, 9, 45
 SQLCTable, 25
 SQLDelete, 23
 SQLDernier, 22, 23
 SQLEcranVersTable, 23, 31
 SQLFiltre, 25, 26, 27, 29, 31
 SQLite, 13, 17, 18, 37, 43
 SQLManagerX, 1, 2, 3, 4, 5, 6, 13, 14, 15, 16, 17, 18, 19, 20, 25, 28, 29, 30, 31, 36, 38, 39, 40
 SQLPrecedent, 21
 SQLPremier, 21, 31
 SQLSuivant, 21
 SQLTableConstruit, 4, 29
 SQLTableVersEcran, 13, 21, 22, 23
 SQLUpdate, 13, 23
 Sybase, 42, 43, 45

W

WDMMap, 3, 10, 37
 WDMAP, 45
 Wdscript, 13
 WebDev, 3, 4, 39, 46
 WinDev, 2, 4, 5, 7, 8, 9, 10, 14, 15, 31, 34, 38, 44

X

xBase, 45