DELPHI et INTERBASE,
Premiers pas

Traditionnellement, Delphi et les bases de données font bon ménage. De la à dire que Delphi est l'outil idéal pour développer des frontaux de bases de données, il n'y a qu'un pas que vous pouvez franchir. En réalité, ce serait une vue assez restrictive des très larges possibilités offertes par cet outil de développement.

Bien que le sujet ai été souvent abordé, le nombre de personnes qui ont encore du mal à utiliser le couple infernal Delphi et Interbase est assez important si on en croit la lecture des News Groups sur le sujet. Voici donc une modeste contribution pour aider tous ceux qui désirent aborder le sujet pas à pas, en évitant nombres d'écueils.

Nous considérerons que vous avez téléchargé Interbase (client et serveur), et la dernière version de IBExpress et que ceux ci sont correctement installés sur votre PC. Le serveur Interbase peut bien sûr être installé sur un Pc du réseau autre que votre machine de travail. Vous pouvez compléter la lecture de ce tutorial par les excellents tutoriaux de **Henry Cesbron Lavau** sur le Web à l'adresse : http://www.developpez.com/hcesbronlavau/IB6Delphi6.htm

Première étape : Création d'un nouveau projet

- Fichier / Nouveau / Application
- Fichier / Nouveau / Module de données

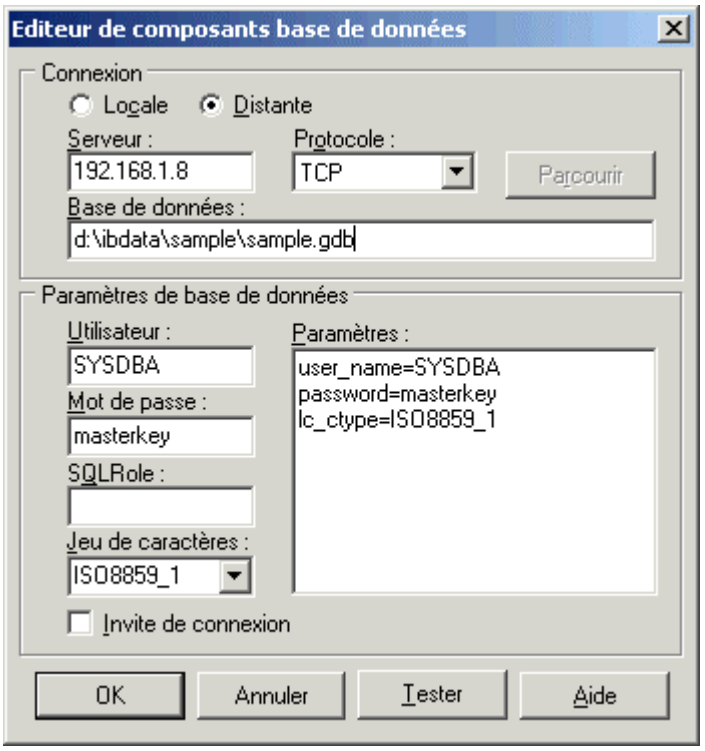
Sélectionnez l'onglet Interbase, puis le composant TIBDatabase et déposez le sur le Datamodule, que nous avons nommé DM et enregistré dans l'unité

uDm.pas. Nous renommons notre composant IBDB que nous trouvons plus pratique que IBDatabase1.

Double cliquez sur le composant que vous venez de déposer et renseignez les zones afin de vous connecter à la base sample.GDB que nous vous proposons avec ce tutoriel.

Comme vous pouvez le constater sur la photo ci-contre, nous avons choisi :

- Une connexion distante
- Indiqué notre adresse IP
- Sélectionné le Protocole TCP
- Indiqué le chemin et le nom de la base
- Renseigné le login **SYSDBA** et le mot de passe **masterkey** qui sont ceux par défaut
- Sélectionné un jeu de caractère par défaut.

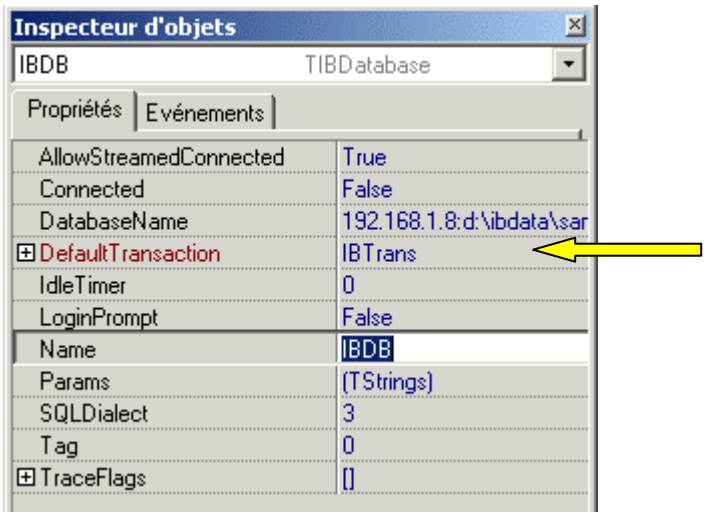


Avant de refermer, n'hésitez pas cliquer sur le bouton « Tester » afin de vous assurer que la connexion se passe bien. Si ce n'est pas le cas, un renseignement est faux.

Deuxième étape : Ajouter une transaction

Dans la palette de composant Interbase, sélectionnez un composant TIBTransaction (le cinquième en partant de la gauche) et déposez le sur le Datamodule.

Renseignez la propriété DefaultDatabase en sélectionnant le nom du composant TIBDatabase. Cliquez sur le composant de base de données IBDB (TIBDatabase) et renseignez la propriété DefaultTransaction en indiquant la transaction que nous venons de déposer et que nous avons nommé **IBTrans**.



Dans la photo ci-contre, le renseignement des propriétés du composant TIBDataBase.

Très bien mais « A quoi sert la transaction » ?

Merci de poser la question. Interbase est un moteur de base de données transactionnel. Les transactions supportent généralement 3 ordres

- Start Démarre la transaction
- Commit Approuve la transaction
- RollBack Annule la transaction

D'accord, mais à quoi ça sert ?
Supposons le cas suivant :

Vous êtes chargés (votre application) de débiter le compte de Monsieur A pour créditer le compte de monsieur B
Votre programme commence à débiter le compte A et soudain, panne de courant.
Dans un système de fichier non transactionnel, monsieur A est débité, alors que monsieur B n'est pas crédité, la somme à disparue…

Dans un système transactionnel, le compte de monsieur A ne sera réellement débité que lorsque la transaction aura été approuvée par un ***Commit***.

StartTransaction

Débit du compte A

Crédit du compte B

Commit

Si un incident se passe durant l'opération, le programme utilisera l'ordre ***RollBak*** pour annuler la transaction démarrée par ***StartTransaction***

Avant d'aller plus loin, notez que

- Une transaction globale ne peut satisfaire tous les besoins d'une application.
- Qu'une transaction doit être ouverte et refermée aussi tôt que possible.

Pour simplifier, sachez que la transaction conserve une copie de toutes les modifications. Si vous conservez vos transaction ouvertes du début à la fin de l'application, votre programme va devenir de plus en plus lent, Interbase va avoir de plus en plus de travail… Si vous utilisez la même transaction pour tous les composant Query ou Table, qu'allez vous ***commiter*** ou annuler ? toutes les modifications, sur n'importe quelle table ?

Il est important de dissocier les transactions en utilisant autant de composants que nécessaire en fonction des besoins de l'application.

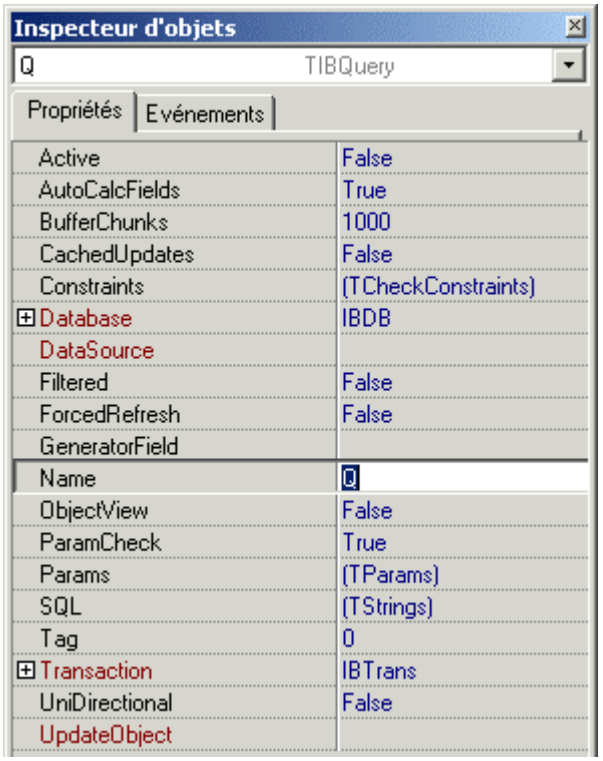
L'exemple ci-dessous démontre l'emploi d'une transaction indépendante de celle définie par défaut.

```
Procedure UpdateMatable(MaChaine : String);
Const
 cMaRequeteUpdate = 'UPDATE MaTable SET MonChamp=%S';
begin
   With TIbSQL.Create(Nil) do
   begin
     try
       Database := IBDB;
       Transaction := TibTransaction.Create(Self);
       Transaction.StartTransaction;
       SQL.Text := Format(cMaRequeteUpdate,[MaChaine]);
       try
        ExecQuery;
        Transaction.Commit;
       Except
         On E: Exception do
         begin
           ShowMessage('UpdateMatable ' + #13 + e.message);
           if Transaction.InTransaction then
            Transaction.RollBack;
         end;
       end;
     finally
       Free;
     end;
```

```
    end;
end;
```

Troisième étape : Ajouter une requête (TIBQuery)

De nouveau dans la palette Interbase, sélectionnez un composant TIBQuery (le second en partant de la gauche) et déposez le sur le DataModule.



Double cliquez sur le champ de la propriété DataBase, le nom de la connexion à la base apparaît. Si c'est pas le cas, sélectionnez dans la liste déroulante.
La transaction par défaut se renseigne toute seule

Cliquez sur le bouton de la propriété SQL (à droite) et saisissez votre requête SQL

Ex : Select * From MaTable ou MaTable correspond au nom d'une table de la base

Quatrième étape : Ajouter un TIBUpdateSQL

De nouveau dans la palette Interbase, sélectionnez un composant TIBUpdateSQL (le sixième en partant de la gauche) et déposez le sur le DataModule. Donnez lui un nom et double clichez sur la propriété **UpdateObject** de notre TIBQuery. Le nom de notre nouveau composant apparaît.

A quoi sert le UpdateSQL ?

A gérer automatiquement les opérations de modification / Ajout / Effacement et Rafraîchissement.

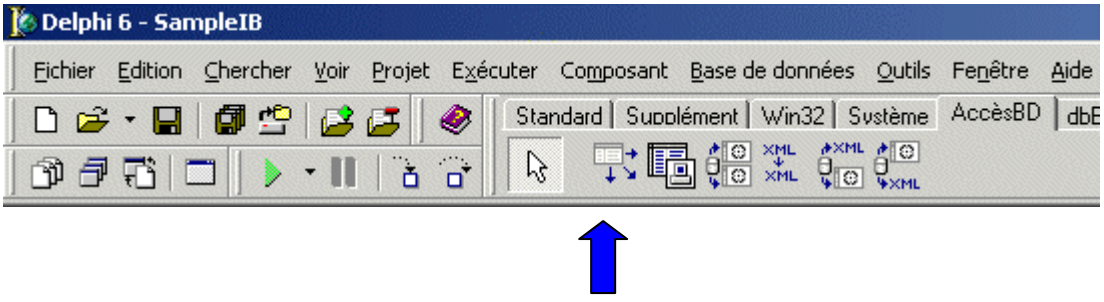
Si vous disposez d'une version entreprise de Delphi, un assistant vous aide à remplir les quatre requêtes. Sinon, voici un exemple de requête Update (vous trouverez les autres dans le projet exemple joint)

```
update TB_PLANNER
set
  PLA_PK = :PLA_PK,
  STARTTIME = :STARTTIME,
  ENDTIME = :ENDTIME,
  SUBJECT = :SUBJECT,
  COLOR = :COLOR,
  IMAGE = :IMAGE,
  CAPTION = :CAPTION,
  NOTES = :NOTES
where
  PLA_PK = :OLD_PLA_PK
```

Notez le double point devant dans l'expression **STARTTIME = :STARTTIME** Ceci indique à Delphi qu'il s'agît d'un paramètre qui sera renseigné automatiquement par la valeur correspondante.

Nb : Ne perdez pas de temps à renseigner les autres requêtes maintenant, vous aurez tout loisir de le faire ensuite en vous inspirant de l'exemple joint à télécharger.

Cinquième étape : Ajouter une un TDATASource



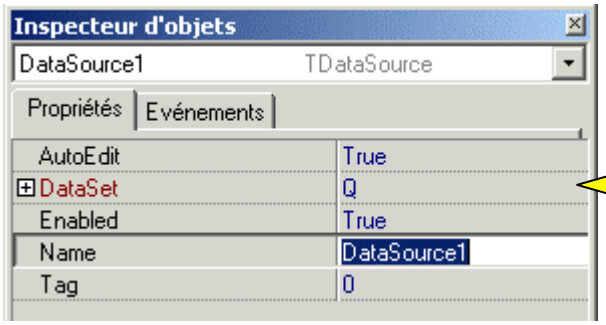
Qu'est ce donc que ce composant, à quoi sert-il ?

Delphi distingue les composants d'accès aux données des composants dits visuels (Ex : un champ de saisie). Pour relier les composants invisibles aux composants visuels, on utilise un composant tiers, le TDataSource.

Extrait de l'aide de Delphi

> **Utilisez TDataSource pour :**
>
> **Fournir un canal entre un ensemble de données et les contrôles orientés données d'une fiche permettant ainsi l'affichage, les déplacements, et la modification des données de l'ensemble de données sous-jacent.**

Maintenant que nous savons à quoi sert le composant, voyons comment l'employer.



Pour connecter le composant de base de données avec le DataSource indiquez simplement le nom du composant dans la propriété DataSet

Donnez un nom au DataSource

Notez que le composant TDataSource peut églement servir à connecter deux tables (requêtes) pour créer une relation Maître Détail.

Si vous utilisez deux requêtes

Exemple :

```
Requête 1 =
      SELECT ID, CHAMP1, CHAMP2 From TableMaitre
Requête 2  =
      SELECT IDMAITRE, CHAMP1, CHAMP2 From TableDETAIL WHERE IDMAITRE=:ID
```

Le renseignement de la relation `IDMAITRE=:ID` peut être automatiquement mis à jour par Delphi. Pour ce, il suffit d'indiquer le DataSource qui référence la requête1 dans la propriété DataSource de la requête 2.

Pour une relation entre deux tables, le principe est le même, Datasource qui référence Table1 dans propriété Datasource de table2 et renseignement du champ MasterField de Table2.

Sixième étape : Ajouter un composant Visuel

Au tout début du tutoriel, nous avons créé une nouvelle application qui a créé une fiche vierge.
Sélectionnez l'unité Unit1 et déclarez l'unité uDm dans la section i**mplementation** comme dans l'exemple ci-dessous

```
implementation
uses uDm;
{$R *.dfm}
```

Sélectionnez la fiche, puis un composant Grille dans la palette ContrôleBD (le premier en partant de la gauche) et déposez le sur la fiche.
Renseignez la propriété DataSource de la grille (choisissez avec la liste déroulante), la liaison du composant visuel avec la table ou requête (non visuelle) est faite.

Enregistrez le projet, les bases de votre première application Delphi Interbase sont posées.

Septième étape : Découverte des notions essentielles

En premier lieu, pour voir le résultat de votre requête, il est indispensable d'indiquer au programme que vous le souhaitez.

MonQuery.Active :=True ; Ouvre la requête
MonQuery.Active :=False ; ferme la requête

Le fait d'ouvrir une requête, ouvre automatiquement la connexion si celle–ci est fermée. Pour ouvrir explicitement une connexion, utilisez l'instruction suivante :

```
IBDB.Connected:=True;
```

Qu'est ce qui change par rapport à une base de donnée de type « fichier » ?

En premier lieu, en environnement Client Serveur on a pour habitude de ne faire transiter par le réseau que les informations essentielles.

En mode fichier, une requête sur une table Paradox via le BDE (borland Database Engine) rapatrie toutes les données vers votre application. Ceci s'explique par le fait que nous n'avons pas, dans ce cas, un programme dit « le serveur » qui fait la sélection pour nous.

En mode client Serveur, la requête est exécutée sur le serveur et seul le résultat est rapatrié.

Données partielles

Cette logique, simple à appréhender, joue parfois quelques tours au développeur qui n'a pas l'habitude. Un système de bufferisation automatique limite le nombre de lignes réellement retournés par une requête. Les lignes ne seront appelées que lorsque l'application les demandera explicitement (Déplacement du curseur dans une ligne de données = Fetch). Ceci s'explique aisément par le fait que, par nature, il est inutile de tenter d'afficher une information portant sur 10 000 lignes.

Le cas classique est la DBCombolistBox qui utilise une requête SQL et qui n'affiche qu'une seule ligne alors que la table en contient dix et plus.. La raison est assez simple à comprendre. Ce composant n'appelle pas les lignes automatiquement mais simplement la première. Le fetch ne porte que sur une ligne. Pour afficher toutes les lignes, utilisez l'instruction **`maTIBQuery.FecthAll ;`**

Mise à jour de l'information sur différents postes

Autre cas classique, les modification effectuée sur un poste ne sont pas visibles sur un autre. Ceci peut paraître bizarre, mais c'est entièrement normal.
La raison ? C'est la logique même du mode client Serveur. Les données affichées dans votre programme, le client, ne sont en réalité qu'une copie partielle à un instant T de la base qui réside dans le serveur. Pour vérifier que vous disposez de la dernière mise à jour, vous devez rafraîchir les données. Le seul vrai moyen de rafraîchir les données et de rejouer la requête. La fermer et la rouvrir.

L'exemple de code ci-dessous permet de fermer et de réouvrir une requête en gardant le curseur sur la même ligne. Généralement, cette action est invisible pour l'utilisateur.

```
procedure TDM.GenericRefresh(DataSet: TDataSet);
Var Bmk : TBookMark;
begin
  // Raffraichi via Fermeture  + Ouverture
  // sans cette option les maj ne sont pas visibles
  With DataSet do
   begin
     if Active then
     begin
       Bmk := GetBookmark;
       Try
         Active:=False;
         Active:=True;
         GotoBookMark(Bmk);
       finally
        FreeBookmark(Bmk);
       end;
     end;
   end;
end;
```

Les clés auto-incrémentées.

Dans une application classique, la gestion des clés auto-incrémentées sont assurées par le moteur auxiliaire (BDE, ODBC). Avec Interbase, la création de la clé doit être explicite.

En règle générale, la création d'une clé auto-incrémentée passe par les étapes suivantes :

- Un Trigger Before Insert (Déclencheur automatique avant insertion)
- Un générateur
- Une procédure stockée (pas obligatoire mais conseillé surtout dans les versions antérieures à Delphi6)
- Un événement dans le programme Delphi

Le déclencheur

```
CREATE TRIGGER TB_PLANNER_BI FOR TB_PLANNER
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.PLA_PKID IS NULL) THEN
      NEW.PLA_PKID = GEN_ID(GEN_PLA_PKID,1);
END
```

La procédure stockée

```
CREATE PROCEDURE TB_PLANNER_AUTOINC
RETURNS (
    NEW_VALUE INTEGER)
AS
BEGIN
  NEW_VALUE = GEN_ID(GEN_PLA_PKID, 1); SUSPEND;
END
```

L'appel de la procédure stockée depuis Delphi

```
function TDM.GetAutoInc(IBObjectName: String): Integer;
Var IBSQl : TIbSql;
begin
  // Récupération de la valeur d'un générateur autoInc
  IBSQl := TIbSql.Create(Nil);
  Try
    With IBSql do
    begin
      DataBase:=IBDB;
      Transaction:=TIbTransaction.Create(Nil);
      Transaction.DefaultdataBase:=IBDB;
      Try
        Transaction.Active:=True;
        SQL.Text:='SELECT NEW_VALUE FROM '+IBObjectName+'_AUTOINC';
        ExecQuery;
        Try
         Result := FieldByName('NEW_VALUE').asInteger;
        Finally
         CLose;
        end;
      finally
       Transaction.Active:=False;
       Transaction.Free;
      end;
    end;
  finally
   IbSql.Free;
```

```
  end;
end;

// * Evénnements pour gérer les actions sur la table * //

procedure TDM.IBQNewRecord(DataSet: TDataSet);
begin
  // renseigne la clé primaire de type Générateur
  IBQPLA_PKID.asInteger:=GetAutoInc('TB_PLANNER');
  // pour l'exemple, la date de début = maintenant
  IBQSTARTTIME.AsDateTime:=Now;
  IBQEndTIME.AsDateTime:=IncMinute(IBQSTARTTIME.AsDateTime,15);
end;
```

Notez qu'à partir de Delphi 6, la propriété GeneratorField des composants pour Interbase permet d'appeler le générateur directement, sans passer par la procédure stockée.

Mais au fait, pourquoi se préoccuper de récupérer la valeur de la clé auto-incrémentée puisque un déclencheur (Trigger) se charge de la renseigner ?
Dans une relation maître Détail, votre programme à besoin de connaître la clé de la table Maître pour renseigner la table détail.

Voici que s'achève ce tutorial. Nous espérons qu'il vous aura apporté ce que vous cherchiez..

Si vous n'êtes pas familiers avec Interbase,

- Vous pouvez consulter le tutorial Comment configurer Interbase : Premiers pas.

Pour aller plus loin :

- Les excellents tutoriaux de **Henry Cesbron Lavau** sur le Web à l'adresse :

http://www.developpez.com/hcesbronlavau/IB6Delphi6.htm

Fichier à télécharger : SampleIB.Zip

L'exemple est écrit en Delphi6.
Cet exemple dispose de 3 unités
uMain.pas source de l'interface IHM
uDM.Pas source pour manager la base IB (traditionnellent séparés du IHM)
uInitInterbase.pas : Sert dans l'exemple pour initialiser la base à partir d'un fichier ini (joint)
Vous trouverez également

- La base d'essai et Les scripts SQL pour générer la base d'essai.

Si vous ne parvenez pas à télécharger l'exemple, essayez sur le site http://www.plusfacile.com/

Auteur du tutorial : jjm@plusfacile.com

Configuration du serveur Interbase de Borland™

Ou trouver Interbase ?

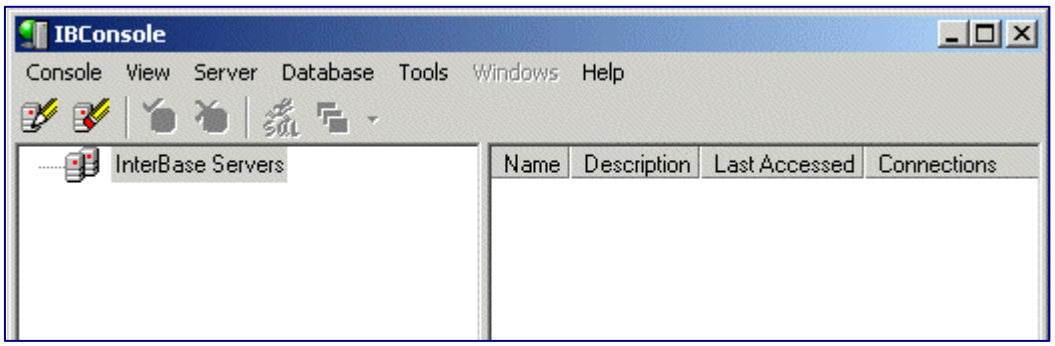
Sur le site de Borland

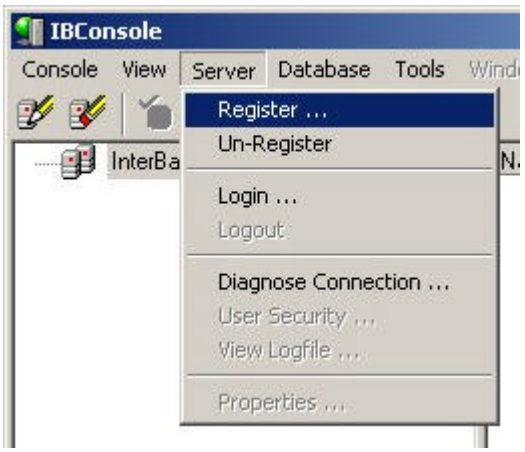
http://www.borland.com/devsupport/interbase/opensource/

Ou sur le site de IBPhoenix

http://www.ibphoenix.com/ibp_download.html#100B2

Le programme d'installation de Interbase installe différents outils et notamment IBConsole.

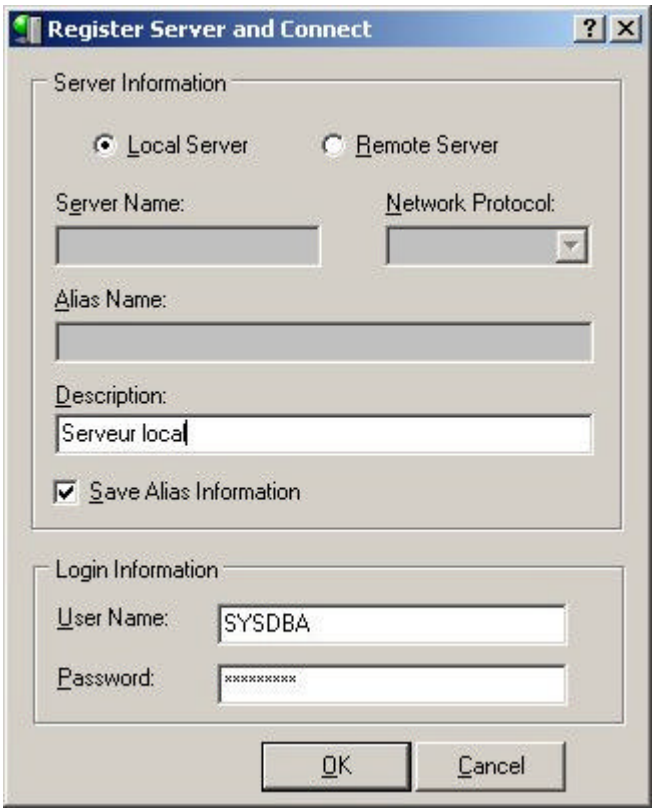




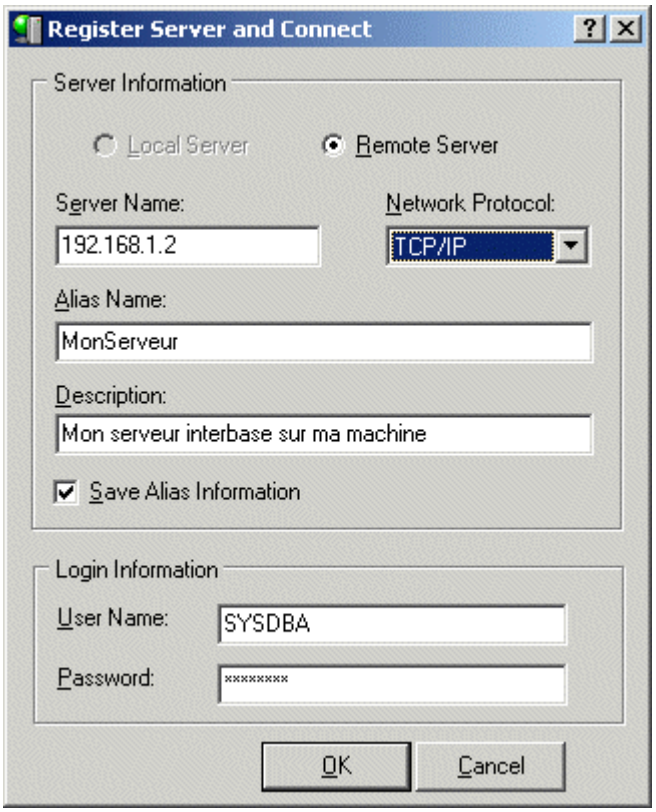
Cliquez sur le menu « Server » et choisissez l'option de menu « Register »

Configuration du serveur Local (si sur votre PC)

Rien de bien particulier, si ce n'est le fait que vous devez spécifier une description, le login et le password masterkey par défaut

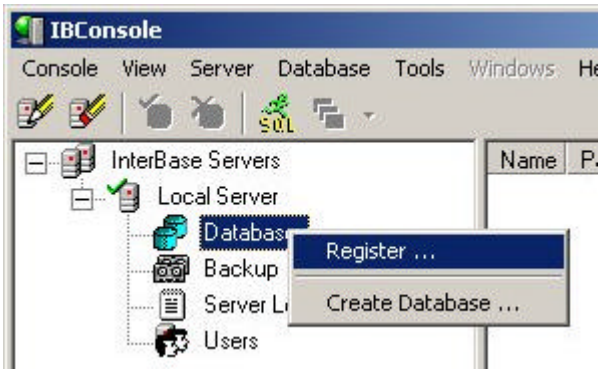


Configuration du serveur distant



Le serveur distant peut, sans aucune distinction, être votre PC ou un PC du réseau dont vous connaissez l'adresse IP

Enregistrement de la base dans IBConsole



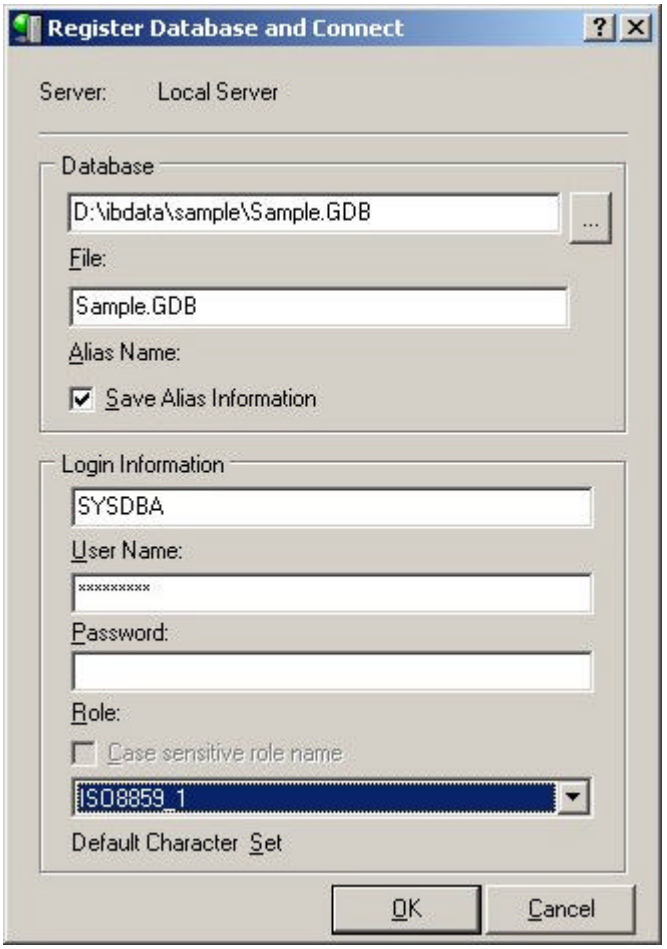
Cette étape n'a rien d'obligatoire. Delphi peut accéder à une base même si celle-ci n'est pas « registered » sur le serveur. Toutefois, si vous souhaitez accéder à la base depuis IBConsole, il est impératif de l'enregistrer.
Notez que dans ce cas, « signalé » la base est plus approprié que « Enregistrer » puisqu'il n'y a aucune action de sauvegarde attachée à l'opération.

Enregistrement sur serveur local ou distant

Indiquez le chemin complet de la base.
Indiquez un nom d'alias (nous avons laissé Sample.GDB, mais nous aurions pu mettre Mabase)

Indiquez le login, puis le password et le default Character Set.

Notez que la saisie des informations peut prêter à confusion en ce sens que les libéllés sont sous les champs à renseigner.



Pour mener à bien ces opérations, il faut évidemment que la base existe déjà. Vous pouvez la créer à partir de IBConsole, ou la générer à partir d'un script. C'est ce que nous allons voir dans le paragraphe qui suit.

Créer la base à partir d'un script.

Idéalement, il est préférable de créer deux scripts. Le premier pour créer la base, le second pour la renseigner.
Pourquoi ? Vous pouvez bien sûr créer un script global, mais Interbase va rechigner sur le Dialect 3 et vous proposer de créer un Dialect 1. En soit, ce n'est pas très grave. Sauf que dans certains cas, l'utilisation du Dialect 1 interdira certains ordres du script qui nécessitent le Dialect 3. Pour éviter œ genre de désagrément, autant prendre de bonnes habitudes en séparant la création de la base du script de renseignement.

Remarquez la ligne **`CREATE DATABASE '192.168.1.2:d:\ibdata\sample\sample.gdb'`**
Celle indique la chaîne de connection vers la base. Bine évidemment, si vous envisagez de mettre votre base ailleurs, vous devrez modifier cette chaîne. Dans le même esprit, l'adresse IP que vote indiquerez doit correspondre à l'adresse IP de la machine sur laquelle est installé votre Serveur Interbase

La ligne `USER` **`'SYSDBA' PASSWORD 'masterkey'`** Indique le login et password qui sera utilisé pour la connexion à la base. Ceux ci-doivent exister avant de lancer le script.
Nb : SYSDBA et masterkey est le couple par défaut de tout serveur interbase nouvellement installé.

Exemples de scripts (les scripts complets sont joints)

Script de création de la base

```
SET SQL DIALECT 3;

CREATE DATABASE '192.168.1.2:d:\ibdata\sample\sample.gdb'
USER 'SYSDBA' PASSWORD 'masterkey'
PAGE_SIZE 2048
DEFAULT CHARACTER SET ISO8859_1;
```

Script de création des éléments (domaines, tables, déclencheurs, données, ect)

```
CREATE DOMAIN DOMCAPTION AS VARCHAR(50) CHARACTER SET ISO8859_1;
…
CREATE GENERATOR GEN_PLA_PKID;
```

```
SET GENERATOR GEN_PLA_PKID TO 5;

CREATE TABLE TB_PLANNER (
    PLA_PKID DOMPKID NOT NULL,
    PLA_PK DOMGUI NOT NULL collate ISO8859_1,
    STARTTIME DOMDATETIME,
    ENDTIME DOMDATETIME,
    SUBJECT DOM_LARGE_TEXT100 collate ISO8859_1,
    COLOR DOMINTEGER,
    IMAGE DOMINTEGER,
    CAPTION DOMSMALLINT,
    NOTES DOM_VC4096 collate ISO8859_1);

INSERT INTO TB_PLANNER (PLA_PKID, PLA_PK, STARTTIME, ENDTIME, SUBJECT, COLOR,
IMAGE, CAPTION, NOTES) VALUES (3, '{B101CA0B-D4B8-45A8-9344-28741EFB4946}',
'10/24/2001 10:14:17', '10/24/2001 10:20:17', 'Rendez vous chez..', NULL, NULL,
NULL, 'Nouveau texte');

COMMIT WORK;
```

La suite du script est dans l'archive jointe

Fichiers à télécharger (sur www.developpez.com):

SampleIB.Zip

L'exemple est écrit en Delphi6.

Cet exemple dispose de 3 unités

uMain.pas source de l'interface IHM

uDM.Pas source pour manager la base IB (traditionnellent séparés du IHM)

uInitInterbase.pas : Sert dans l'exemple pour initialiser la base à partir d'un fichier ini (joint)

Vous trouverez également

- La base d'essai et
- Les scripts SQL pour générer la base d'essai.

Si vous ne parvenez pas à télécharger l'exemple, essayez sur le site
http://www.plusfacile.com/Didactiels/Interbase/

Auteur du tutorial : jjm@delphicenter.net