1. INTRODUCTION.

1.1. Objectifs.

Ce cours est destiné aux personnes qui souhaitent développer des applications avec DELPHI et se connecter avec ADO à MS SQL SERVER 2000. Je vais développer un exemple concret de programme clients-serveur depuis les desiderata du promoteur jusqu'au déploiement en passant par l'analyse, le développement de l'application et la stratégie de sécurité. L'essentiel du cours est basé sur les procédures stockées.

Le cours sera abondamment documenté par des saisies d'écrans et des commentaires détaillés.

Ma façon de faire ne rencontre aucun objectif de programmation orientée objet. Je ne créerai aucun composant et je n'utiliserai aucun composant autre que les composants livrés d'origine avec DELPHI 7.

J'ai l'intention de développer ensuite la même application mais en environnement WEB avec ASP et MS SQL SERVER. Ceci fera l'objet d'un cours distinct.

1.2. Souhaits du promoteur du projet.

Cette application très classique m'a été demandée par une connaissance qui en avait assez de ne pas récupérer les cassettes de DVD qu'il prêtait. Rien de bien original mais vous y trouverez tout de même l'insertion d'images et l'utilisation des codes à barres.

Suite à l'interview préalable du promoteur, voici le résumé de ses desiderata :

- possibilité de gérer sa collection de films (ajout, suppression, modification, recherches diverses) en incluant des photos du film, des commentaires et appréciations
- gestion des acteurs et metteurs en scène
- possibilité de disposer de supports différents pour ses films, supports qu'il détient parfois en plusieurs exemplaires
- gestion des prêts de ces exemplaires à des connaissances
- utilisation d'une douchette lectrice de codes à barres et d'une imprimante pour créer ses propres codes à barres collés sur les exemplaires
- possibilité d'imprimer des listes et même des formulaires de prêt
- possibilité d'utilisation sur un réseau local
- possibilité d'imprimer par pages entières les codes à barres des exemplaires pour ne plus avoir qu'à les lier et les coller aux exemplaires lors de leur encodage.

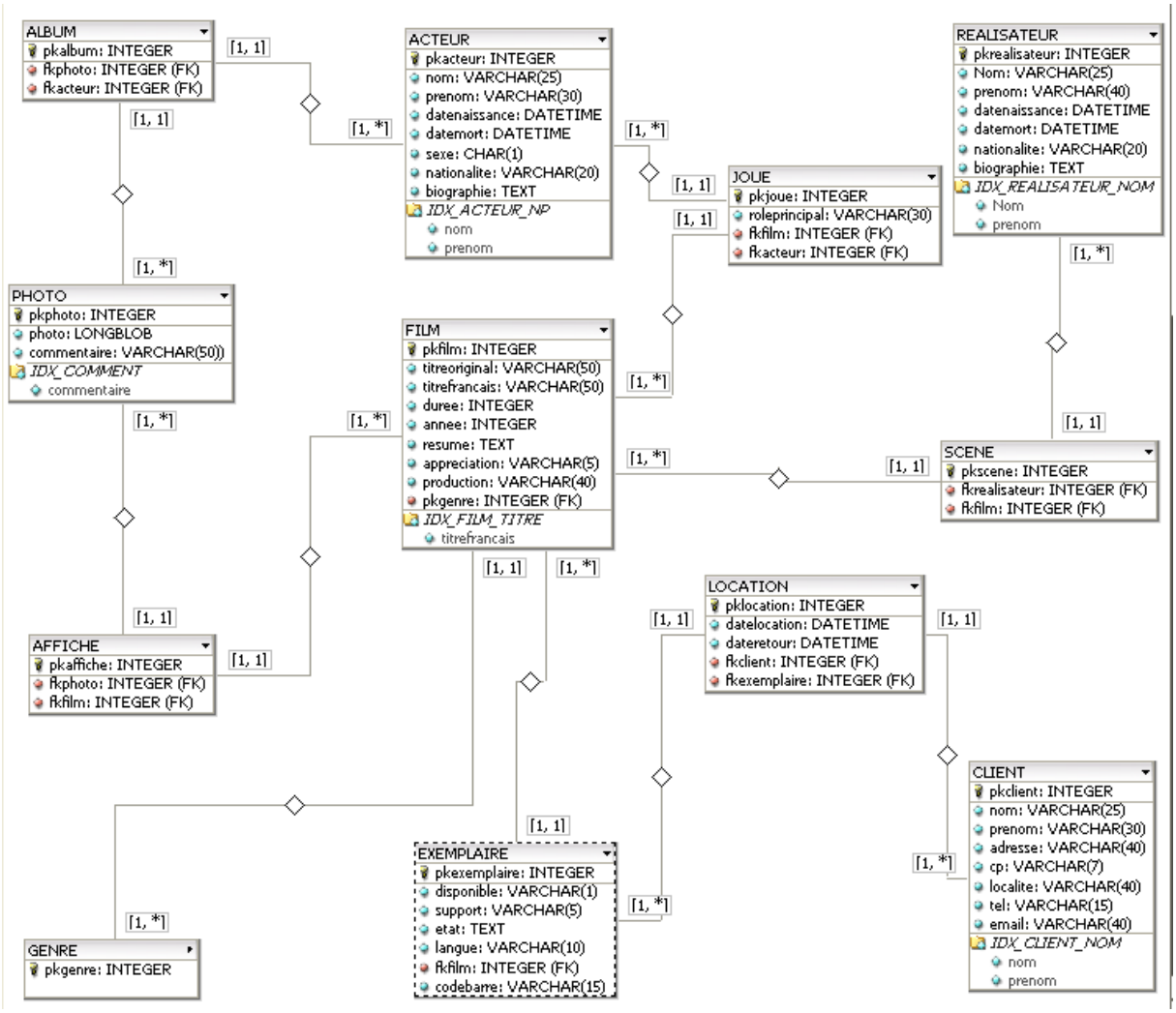
1.3. Schéma d'analyse du projet.

Ce projet que je nommerai MESVIDEOS partira du schéma suivant, réalisé sur FABFORCE DBDESIGNER 4. Il tiendra compte des remarques précisées ci-dessus.

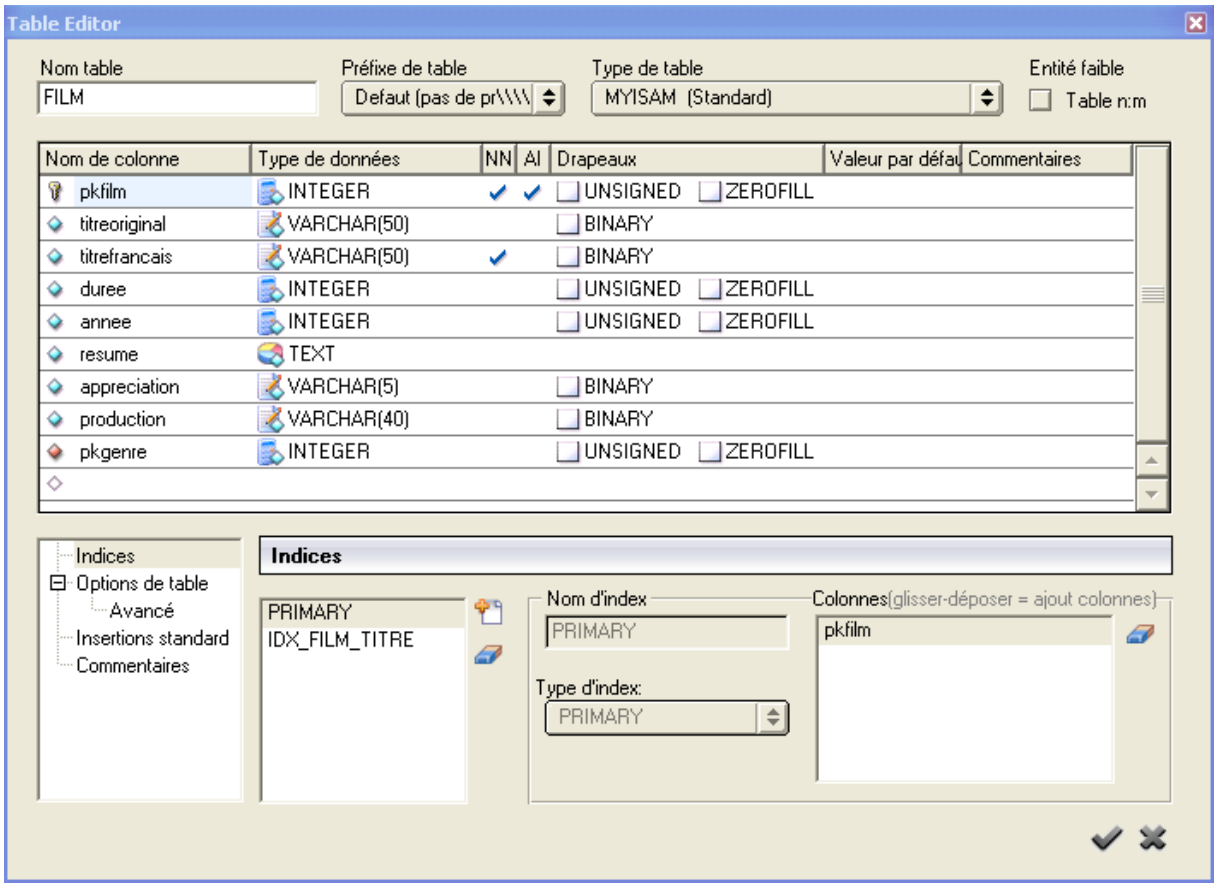
Des modifications pourraient être apportées plus tard en fonction de problèmes rencontrés. De toute façon, une table sera ajoutée à ce schéma : la table PARAMETRE qui contiendra entre autres le dernier numéro de code à barres imprimé, le nombre de documents imprimés par défaut et d'autres paramètres que nous découvrirons plus tard. Cette table ne sera pas liée aux autres ce qui ne nous posera pas de problème de clef étrangère.

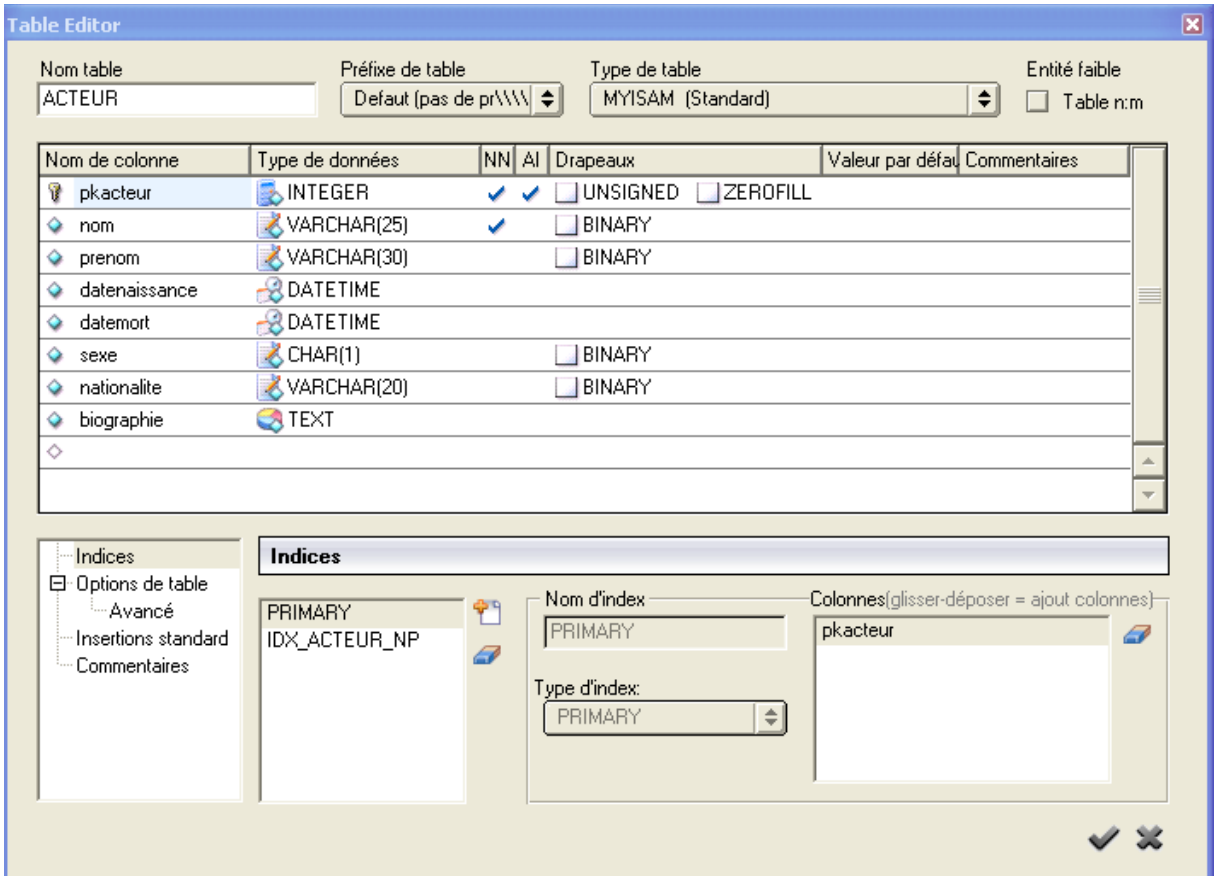
J'ai utilisé comme nom de tables CLIENT et LOCATION. Il ne s'agit pas de créer un programme de commerce de location ou vente mais ces noms sont apparus naturellement.

Certains affirmeront que je pouvais diminuer le nombre de tables en regroupant ACTEUR et REALISATEUR. C'est vrai, mais j'ai voulu utiliser des relations simples et pas deux relations entre les deux mêmes tables.


ALBUM
pkalbum: INTEGER
fkphoto: INTEGER (FK)
fkacteur: INTEGER (FK)
ACTEUR
pkacteur: INTEGER
nom: VARCHAR(25)
prenom: VARCHAR(30)
datenaissance: DATETIME
datemort: DATETIME
sexe: CHAR(1)
nationalite: VARCHAR(20)
biographie: TEXT
IDX_ACTEUR_NP
nom
prenom
REALISATEUR
pkrealisateur: INTEGER
Nom: VARCHAR(25)
prenom: VARCHAR(40)
datenaissance: DATETIME
datemort: DATETIME
nationalite: VARCHAR(20)
biographie: TEXT
IDX_REALISATEUR_NOM
Nom
prenom
JOUE
pkjoue: INTEGER
roleprincipal: VARCHAR(30)
fkfilm: INTEGER (FK)
fkacteur: INTEGER (FK)
PHOTO
pkphoto: INTEGER
photo: LONGBLOB
commentaire: VARCHAR(50))
IDX_COMMENT
commentaire
FILM
pkfilm: INTEGER
titreoriginal: VARCHAR(50)
titrefrancais: VARCHAR(50)
duree: INTEGER
annee: INTEGER
resume: TEXT
appreciation: VARCHAR(5)
production: VARCHAR(40)
pkgenre: INTEGER (FK)
IDX_FILM_TITRE
titrefrancais
SCENE
pkscene: INTEGER
fkrealisateur: INTEGER (FK)
fkfilm: INTEGER (FK)
LOCATION
pklocation: INTEGER
datelocation: DATETIME
dateretour: DATETIME
fkclient: INTEGER (FK)
fkexemplaire: INTEGER (FK)
AFFICHE
pkaffiche: INTEGER
fkphoto: INTEGER (FK)
fkfilm: INTEGER (FK)
CLIENT
pkclient: INTEGER
nom: VARCHAR(25)
prenom: VARCHAR(30)
adresse: VARCHAR(40)
cp: VARCHAR(7)
localite: VARCHAR(40)
tel: VARCHAR(15)
email: VARCHAR(40)
IDX_CLIENT_NOM
nom
prenom
EXEMPLAIRE
pkexemplaire: INTEGER
disponible: VARCHAR(1)
support: VARCHAR(5)
etat: TEXT
langue: VARCHAR(10)
fkfilm: INTEGER (FK)
codebarre: VARCHAR(15)
GENRE
pkgenre: INTEGER
[1, 1]
[1, *]

1.4. Structure des tables.





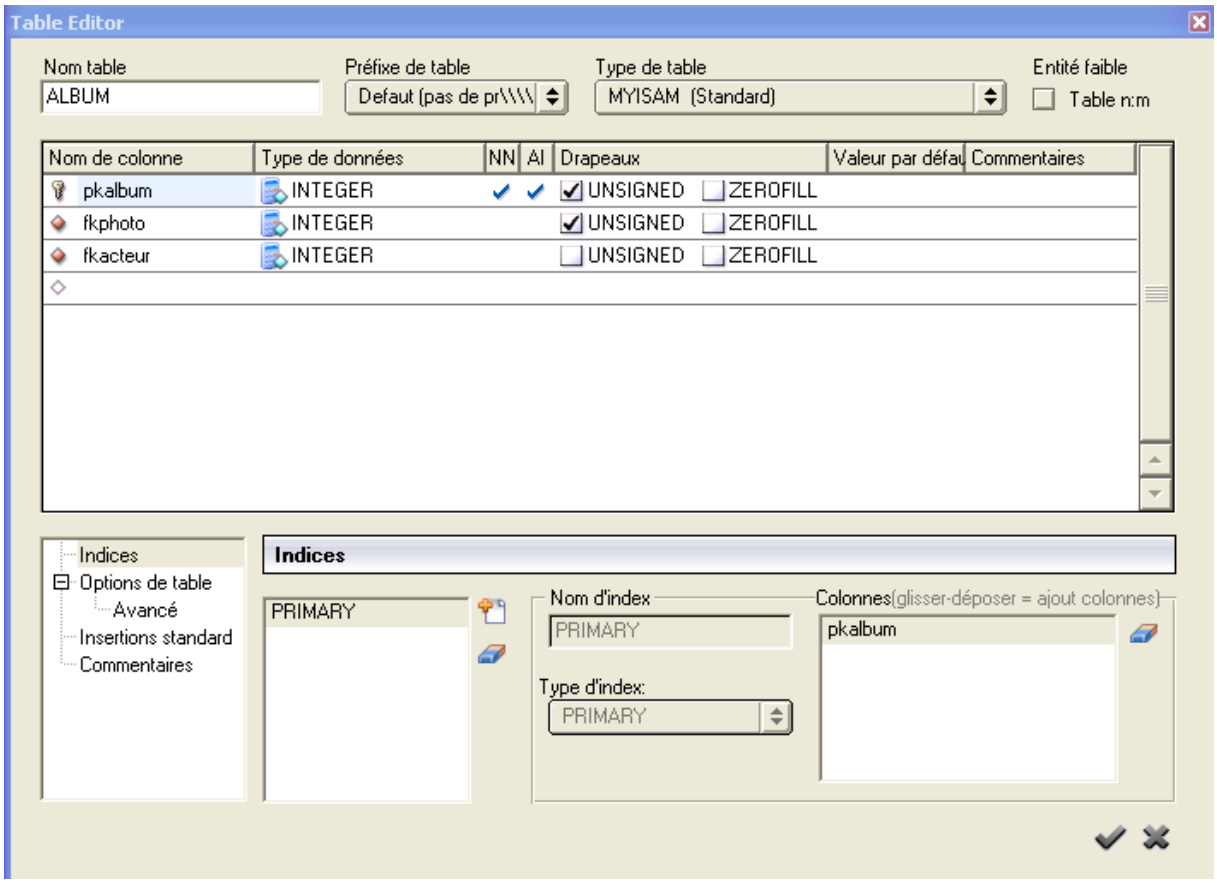


Table Editor
Nom table
ALBUM
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkalbum
INTEGER
UNSIGNED
ZEROFILL
fkphoto
INTEGER
UNSIGNED
ZEROFILL
fkacteur
INTEGER
UNSIGNED
ZEROFILL
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkalbum
Type d'index:
PRIMARY

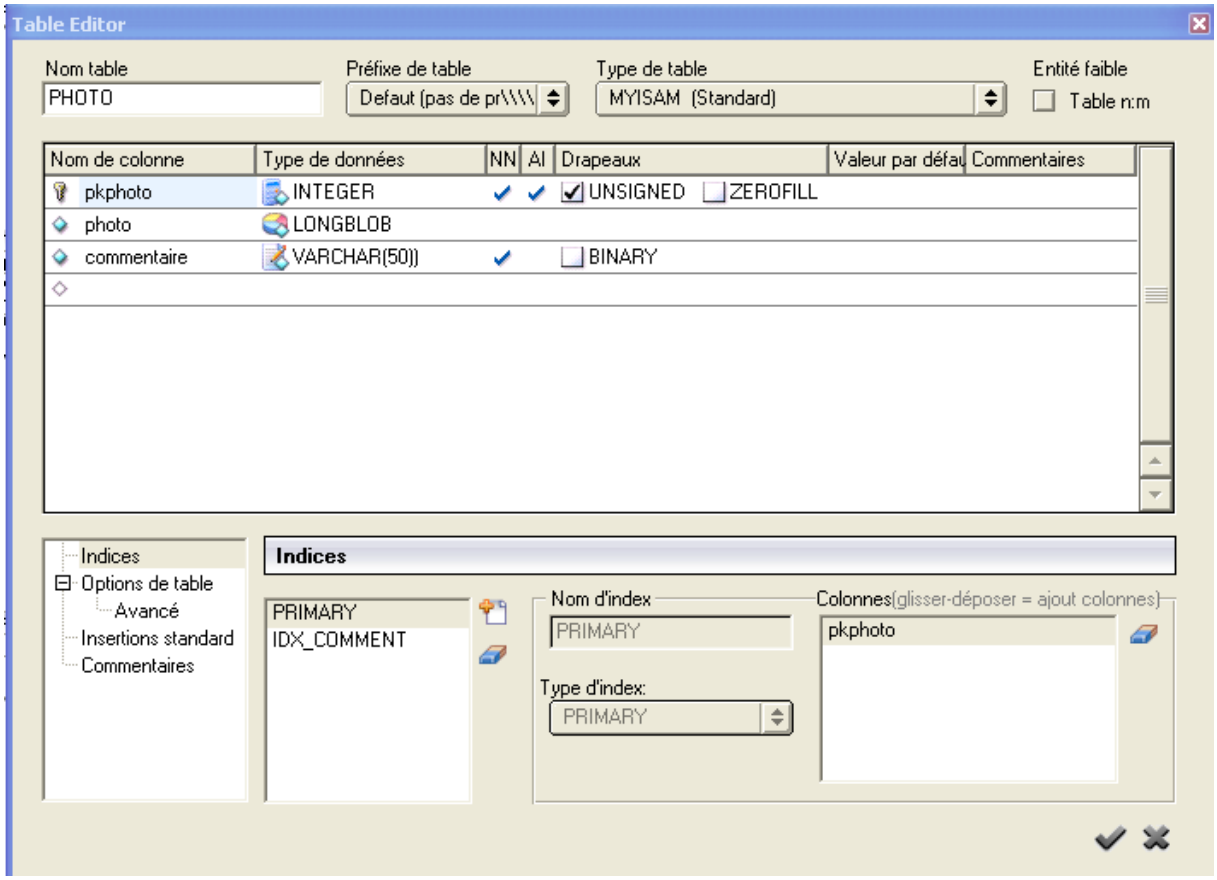


Table Editor
Nom table
PHOTO
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkphoto
INTEGER
UNSIGNED
ZEROFILL
photo
LONGBLOB
commentaire
VARCHAR(50))
BINARY
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
IDX_COMMENT
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkphoto
Type d'index:
PRIMARY

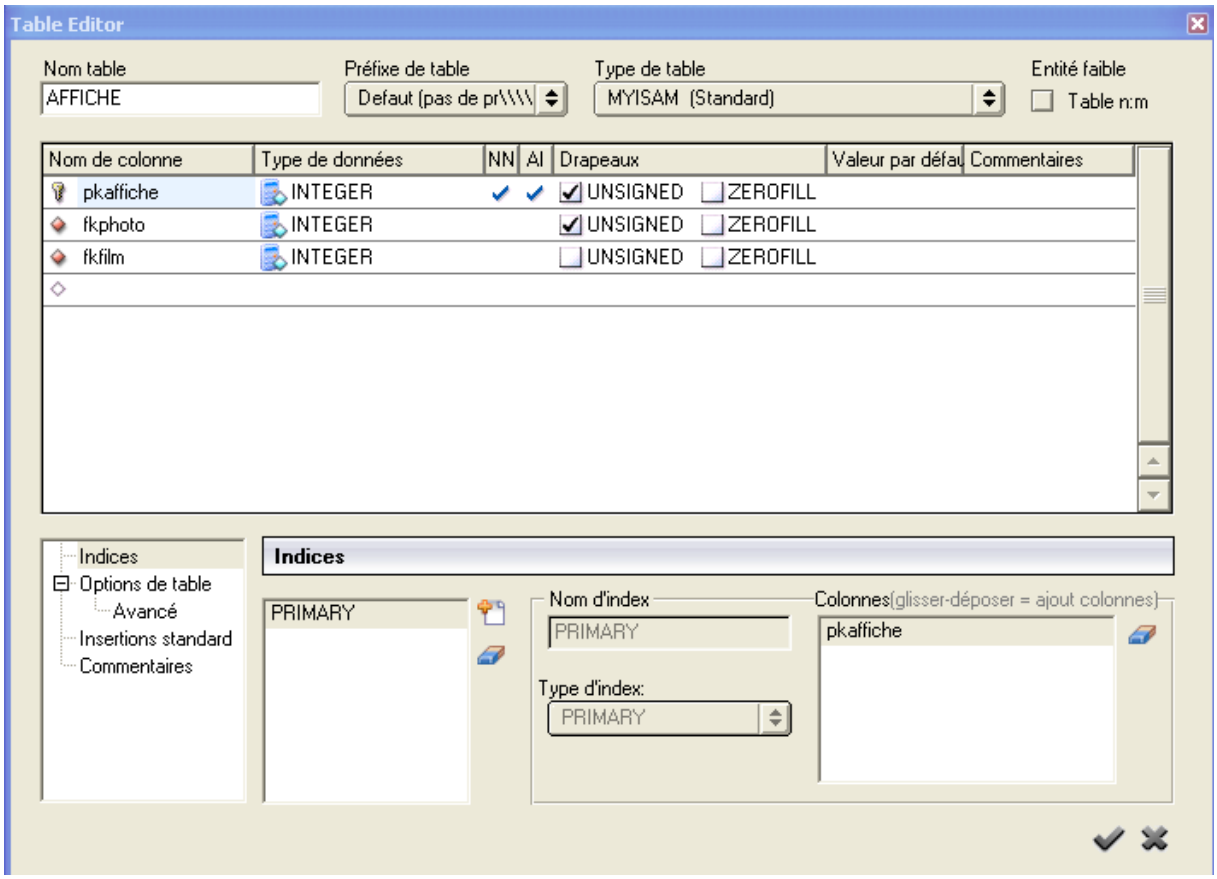


Table Editor
Nom table
AFFICHE
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkaffiche
INTEGER
UNSIGNED
ZEROFILL
fkphoto
INTEGER
UNSIGNED
ZEROFILL
fkfilm
INTEGER
UNSIGNED
ZEROFILL
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkaffiche
Type d'index:
PRIMARY

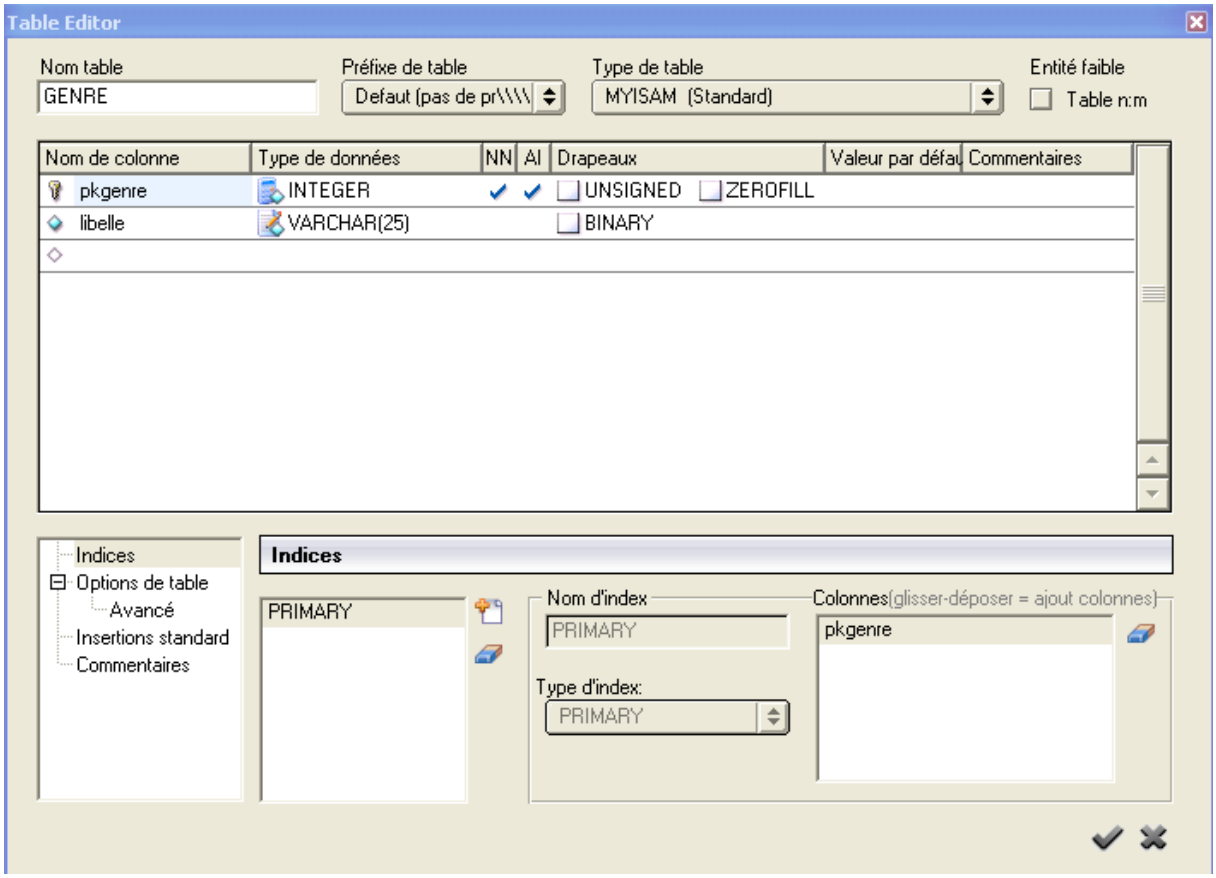


Table Editor
Nom table
GENRE
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkgenre
INTEGER
UNSIGNED
ZEROFILL
libelle
VARCHAR(25)
BINARY
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkgenre
Type d'index:
PRIMARY

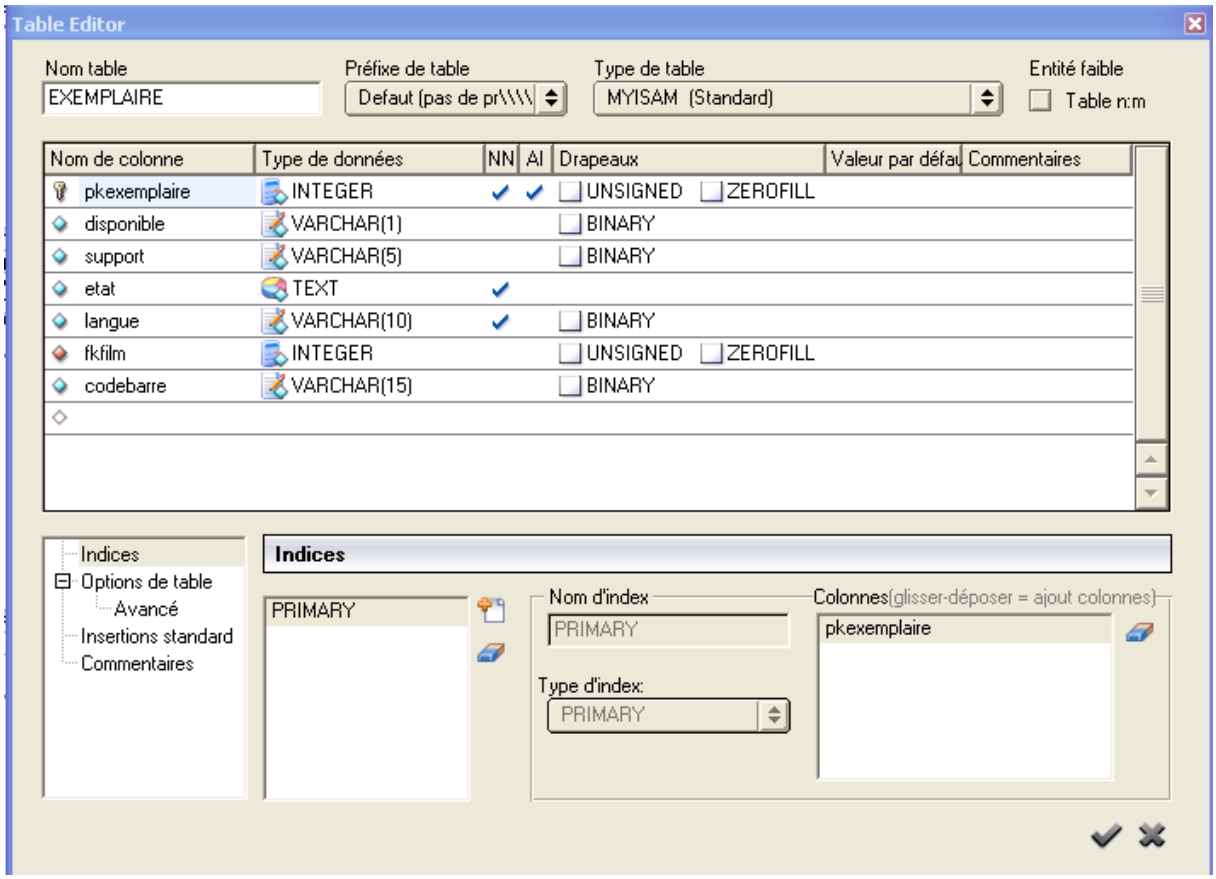


Table Editor
Nom table
EXEMPLAIRE
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkexemplaire
INTEGER
UNSIGNED
ZEROFILL
disponible
VARCHAR(1)
BINARY
support
VARCHAR(5)
BINARY
etat
TEXT
langue
VARCHAR(10)
BINARY
fkfilm
INTEGER
UNSIGNED
ZEROFILL
codebarre
VARCHAR(15)
BINARY
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkexemplaire
Type d'index:
PRIMARY

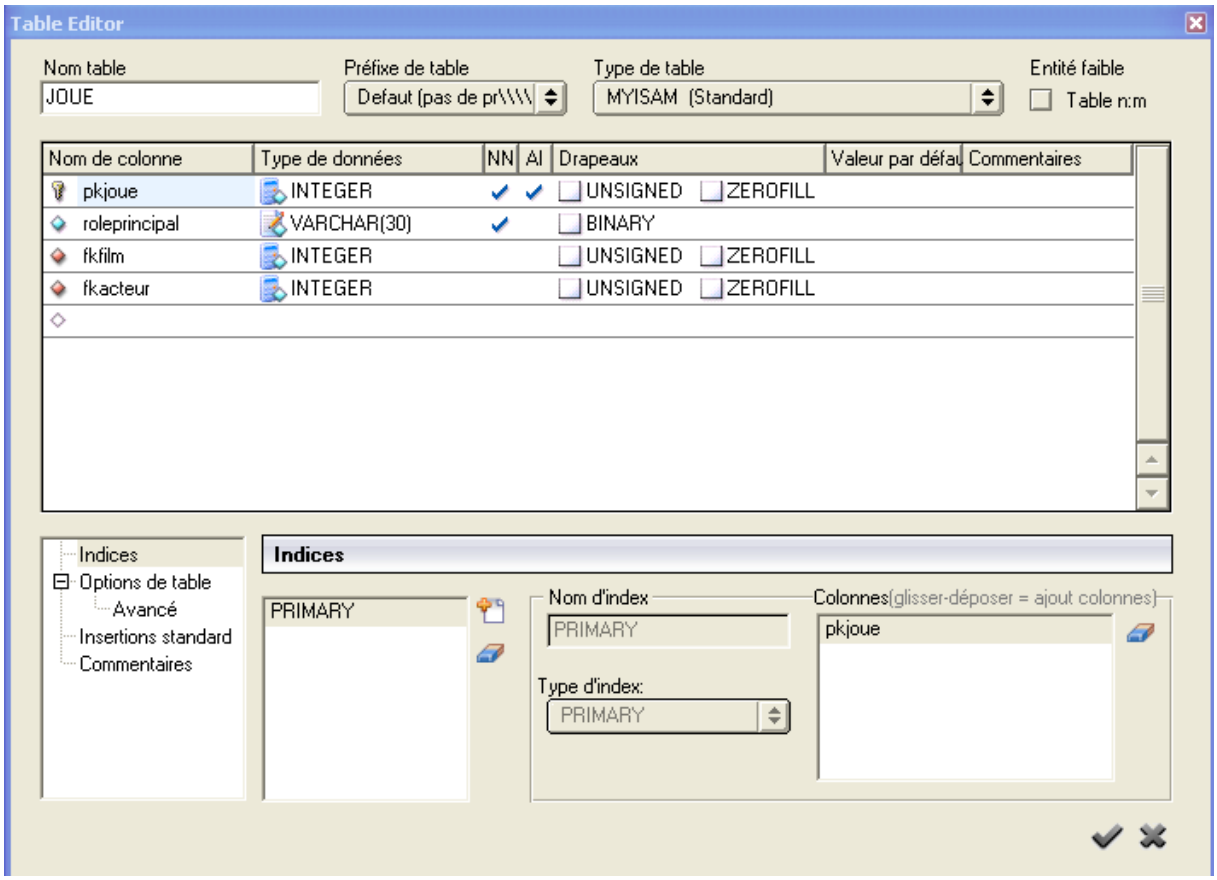


Table Editor
Nom table
JOUE
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkjoue
INTEGER
UNSIGNED
ZEROFILL
roleprincipal
VARCHAR(30)
BINARY
fkfilm
INTEGER
UNSIGNED
ZEROFILL
fkacteur
INTEGER
UNSIGNED
ZEROFILL
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkjoue
Type d'index:
PRIMARY

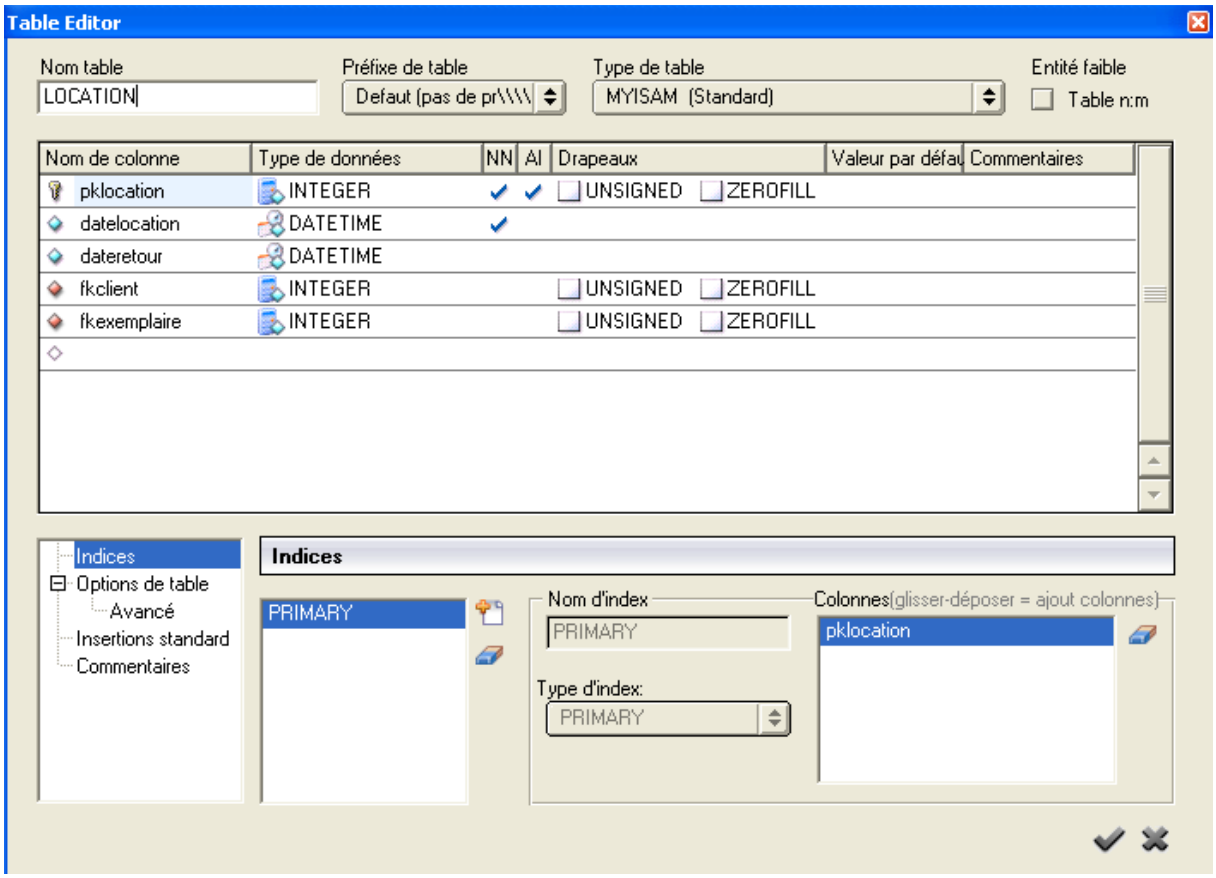


Table Editor
Nom table
LOCATION
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pklocation
INTEGER
UNSIGNED
ZEROFILL
datelocation
DATETIME
dateretour
DATETIME
fkclient
INTEGER
UNSIGNED
ZEROFILL
fkexemplaire
INTEGER
UNSIGNED
ZEROFILL
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
Nom d'index
PRIMARY
Type d'index:
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pklocation

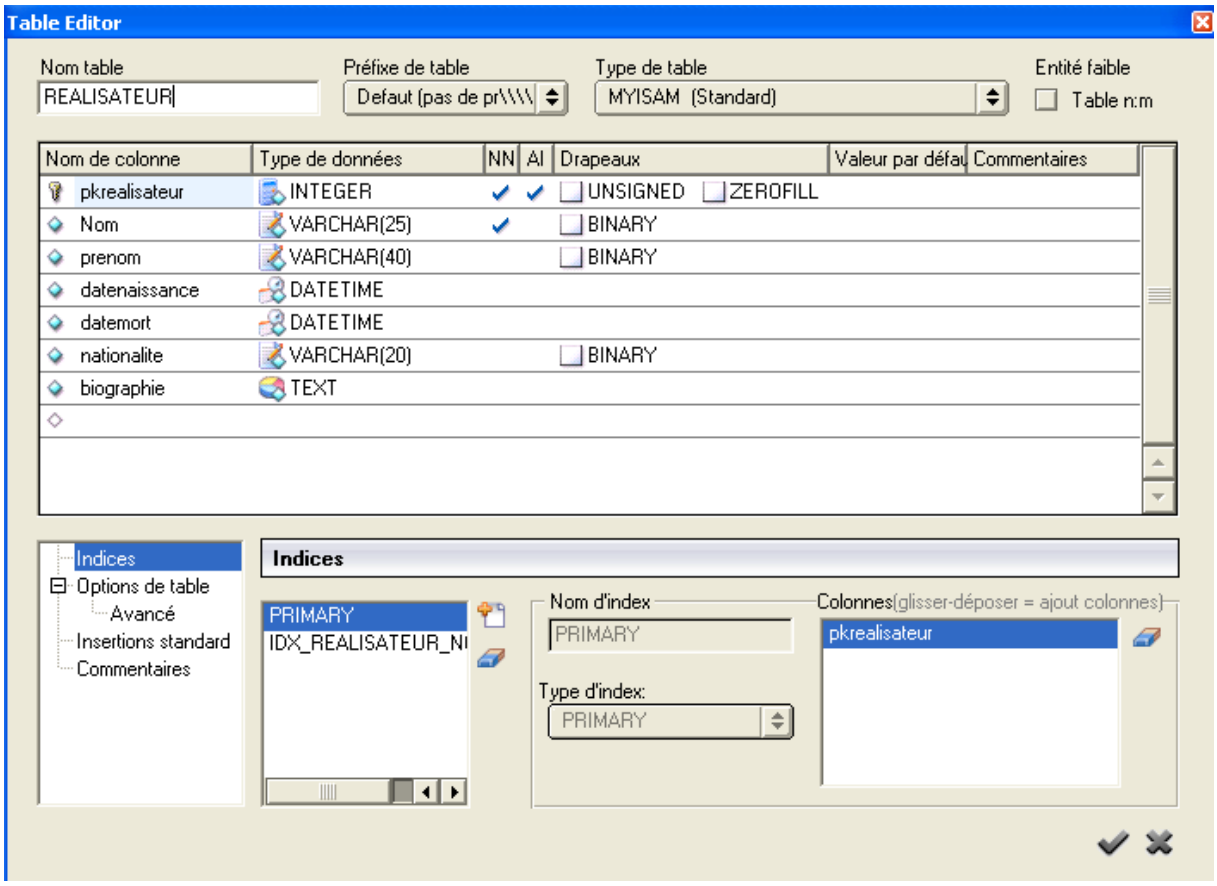


Table Editor
Nom table
REALISATEUR
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkrealisateur
INTEGER
UNSIGNED
ZEROFILL
Nom
VARCHAR(25)
BINARY
prenom
VARCHAR(40)
BINARY
datenaissance
DATETIME
datemort
DATETIME
nationalite
VARCHAR(20)
BINARY
biographie
TEXT
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
IDX_REALISATEUR_NI
Nom d'index
PRIMARY
Type d'index:
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkrealisateur

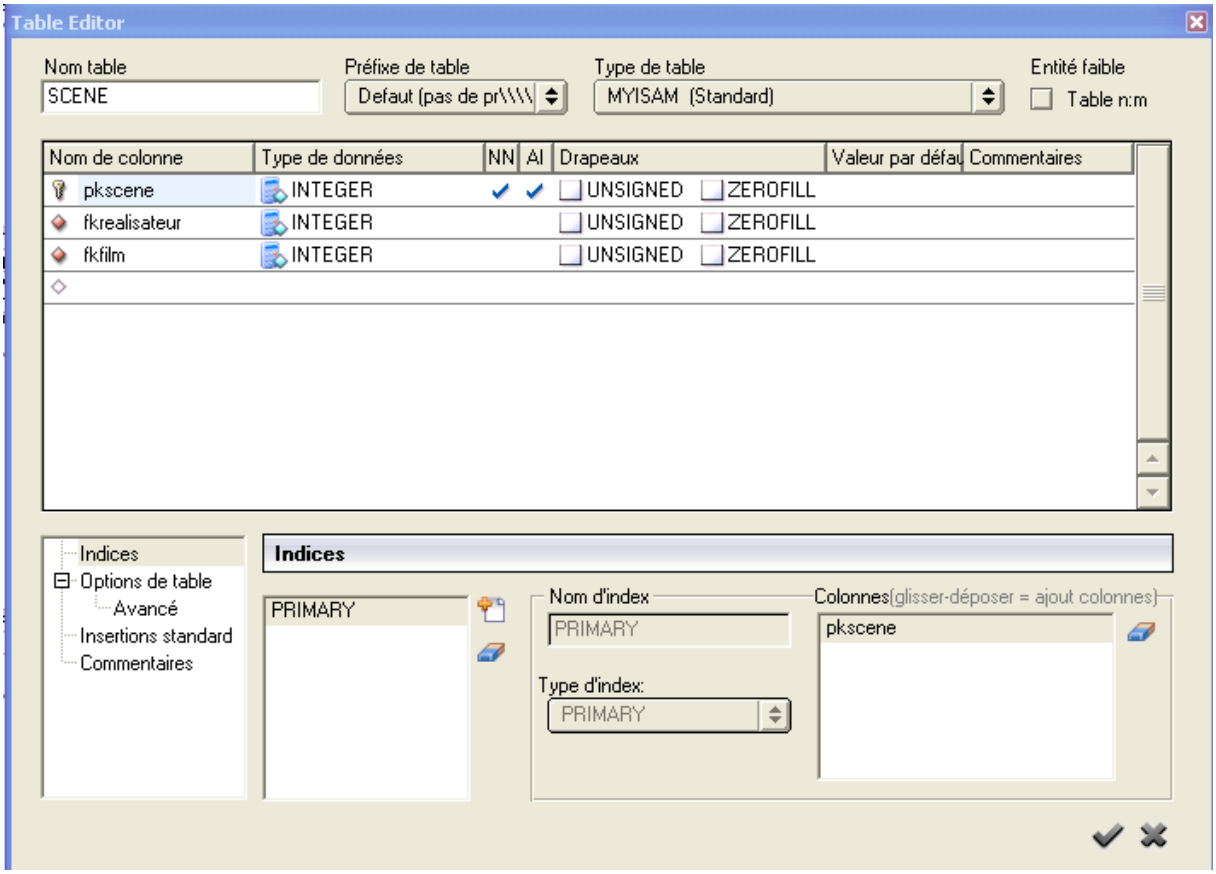


Table Editor
Nom table
SCENE
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkscene
INTEGER
UNSIGNED
ZEROFILL
fkrealisateur
INTEGER
UNSIGNED
ZEROFILL
fkfilm
INTEGER
UNSIGNED
ZEROFILL
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkscene
Type d'index:
PRIMARY

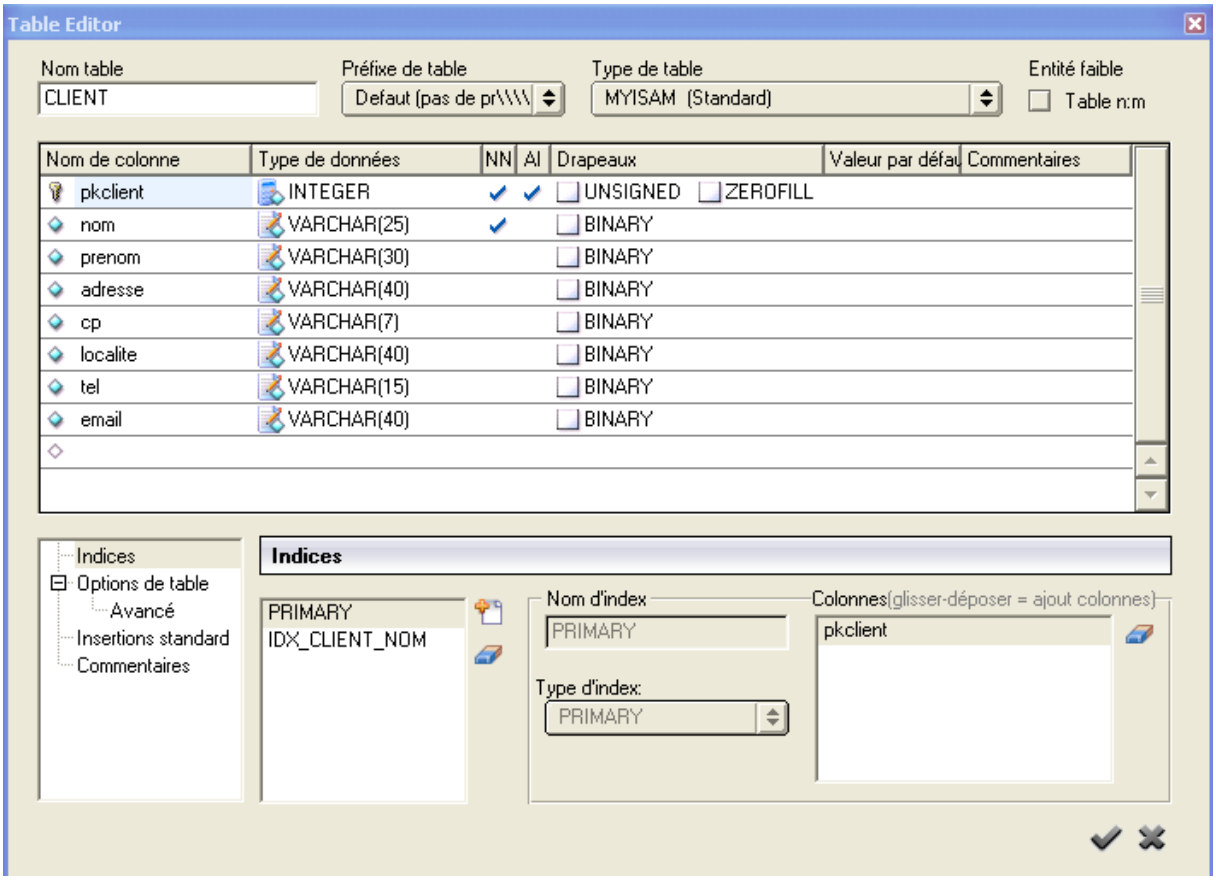


Table Editor
Nom table
CLIENT
Préfixe de table
Defaut (pas de pr\\\\
Type de table
MYISAM (Standard)
Entité faible
Table n:m
Nom de colonne
Type de données
NN
AI
Drapeaux
Valeur par défau
Commentaires
pkclient
INTEGER
UNSIGNED
ZEROFILL
nom
VARCHAR(25)
BINARY
prenom
VARCHAR(30)
BINARY
adresse
VARCHAR(40)
BINARY
cp
VARCHAR(7)
BINARY
localite
VARCHAR(40)
BINARY
tel
VARCHAR(15)
BINARY
email
VARCHAR(40)
BINARY
Indices
Options de table
Avancé
Insertions standard
Commentaires
Indices
PRIMARY
IDX_CLIENT_NOM
Nom d'index
PRIMARY
Colonnes(glisser-déposer = ajout colonnes)
pkclient
Type d'index:
PRIMARY

1.5. Code de la base de données.

USE MASTER
CREATE DATABASE mesvideos

Voici le code généré après les modifications pour être optimisé pour MS SQL SERVER.

```
USE mesvideos
CREATE TABLE CLIENT (
  pkclient INTEGER NOT NULL IDENTITY(0,1) ,
  nom VARCHAR(25) NOT NULL,
  prenom VARCHAR(30) NULL,
  adresse VARCHAR(40) NULL,
  cp VARCHAR(7) NULL,
  localite VARCHAR(40) NULL,
  tel VARCHAR(15) NULL,
  email VARCHAR(40) NULL,
  PRIMARY KEY(pkclient),
);

CREATE TABLE GENRE (
  pkgenre INTEGER NOT NULL IDENTITY(0,1) ,
  libelle VARCHAR(25) NULL,
  PRIMARY KEY(pkgenre)
);

CREATE TABLE PHOTO (
  pkphoto INTEGER    NOT NULL IDENTITY(0,1) ,
  photo IMAGE NULL,
  commentaire VARCHAR(50) NOT NULL,
  PRIMARY KEY(pkphoto),
);

CREATE TABLE ACTEUR (
  pkacteur INTEGER NOT NULL IDENTITY(0,1) ,
  nom VARCHAR(25) NOT NULL,
  prenom VARCHAR(30) NULL,
  datenaissance DATETIME NULL,
  datemort DATETIME NULL,
  sexe CHAR(1) NULL,
  nationalite VARCHAR(20) NULL,
  biographie TEXT NULL,
  PRIMARY KEY(pkacteur),

);

CREATE TABLE REALISATEUR (
  pkrealisateur INTEGER NOT NULL IDENTITY(0,1) ,
  Nom VARCHAR(25) NOT NULL,
  prenom VARCHAR(40) NULL,
  datenaissance DATETIME NULL,
  datemort DATETIME NULL,
  nationalite VARCHAR(20) NULL,
  biographie TEXT NULL,
  PRIMARY KEY(pkrealisateur),
);

CREATE TABLE FILM (
  pkfilm INTEGER NOT NULL IDENTITY(0,1) ,
```

```
  titreoriginal VARCHAR(50) NULL,
  titrefrancais VARCHAR(50) NOT NULL,
  duree INTEGER NULL,
  annee INTEGER NULL,
  resume TEXT NULL,
  appreciation VARCHAR(5) NULL,
  production VARCHAR(40) NULL,
  pkgenre INTEGER NULL,
  PRIMARY KEY(pkfilm),
  FOREIGN KEY(pkgenre)
    REFERENCES GENRE(pkgenre)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE EXEMPLAIRE (
  pkexemplaire INTEGER NOT NULL IDENTITY(0,1) ,
  disponible VARCHAR(1) NULL,
  support VARCHAR(5) NULL,
  etat TEXT NOT NULL,
  langue VARCHAR(10) NOT NULL,
  fkfilm INTEGER NULL,
  codebarre varchar(15) NULL,
  PRIMARY KEY(pkexemplaire),
  FOREIGN KEY(fkfilm)
    REFERENCES FILM(pkfilm)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE AFFICHE (
  pkaffiche INTEGER    NOT NULL IDENTITY(0,1) ,
  fkphoto INTEGER    NULL,
  fkfilm INTEGER NULL,
  PRIMARY KEY(pkaffiche),
  FOREIGN KEY(fkfilm)
    REFERENCES FILM(pkfilm)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(fkphoto)
    REFERENCES PHOTO(pkphoto)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE SCENE (
  pkscene INTEGER NOT NULL IDENTITY(0,1) ,
  fkrealisateur INTEGER NULL,
  fkfilm INTEGER NULL,
  PRIMARY KEY(pkscene),
  FOREIGN KEY(fkfilm)
    REFERENCES FILM(pkfilm)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(fkrealisateur)
    REFERENCES REALISATEUR(pkrealisateur)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);
```

```
CREATE TABLE ALBUM (
  pkalbum INTEGER    NOT NULL IDENTITY(0,1) ,
  fkphoto INTEGER    NULL,
  fkacteur INTEGER NULL,
  PRIMARY KEY(pkalbum),
  FOREIGN KEY(fkacteur)
    REFERENCES ACTEUR(pkacteur)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(fkphoto)
    REFERENCES PHOTO(pkphoto)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE JOUE (
  pkjoue INTEGER NOT NULL IDENTITY(0,1) ,
  roleprincipal VARCHAR(30) NOT NULL,
  fkfilm INTEGER NULL,
  fkacteur INTEGER NULL,
  PRIMARY KEY(pkjoue),
  FOREIGN KEY(fkacteur)
    REFERENCES ACTEUR(pkacteur)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(fkfilm)
    REFERENCES FILM(pkfilm)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE LOCATION (
  pklocation INTEGER NOT NULL IDENTITY(0,1) ,
  datelocation DATETIME NOT NULL,
  dateretour DATETIME NULL,
  fkclient INTEGER NULL,
  fkexemplaire INTEGER NULL,
  PRIMARY KEY(pklocation),
  FOREIGN KEY(fkclient)
    REFERENCES CLIENT(pkclient)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(fkexemplaire)
    REFERENCES EXEMPLAIRE(pkexemplaire)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE INDEX IDX_FILM_TITREORIG ON FILM (titreoriginal)
CREATE INDEX IDX_FILM_TITREFR ON FILM (titrefrancais)
CREATE UNIQUE INDEX IDX_FILM3 ON FILM (pkfilm)
CREATE INDEX IDX_FILM4 ON FILM (pkgenre)
CREATE INDEX IDX_NOMACT ON ACTEUR (nom, prenom)
CREATE UNIQUE INDEX IDX_ACTEUR2 ON ACTEUR (pkacteur)
CREATE INDEX IDX_JOUE1 ON JOUE (fkfilm)
CREATE INDEX IDX_JOUE2 ON JOUE (fkacteur)
CREATE INDEX IDX_NOMCLIENT ON CLIENT (nom, prenom)
CREATE UNIQUE INDEX IDX_CLIENT2 ON CLIENT (pkclient)
CREATE UNIQUE INDEX IDX_EXEMPLAIRE1 ON EXEMPLAIRE (pkexemplaire)
CREATE INDEX IDX_EXEMPLAIRE2 ON EXEMPLAIRE (fkfilm)
```

```
CREATE INDEX IDX_LOCATION1 ON LOCATION (fkclient)
CREATE INDEX IDX_LOCATION2 ON LOCATION (fkexemplaire)
CREATE UNIQUE INDEX IDX_GENRE1 ON GENRE (pkgenre)
CREATE INDEX IDX_NOMREA ON REALISATEUR (Nom, prenom)
CREATE UNIQUE INDEX IDX_REALISATEUR2 ON REALISATEUR (pkrealisateur)
CREATE INDEX IDX_SCENE1 ON SCENE (fkrealisateur)
CREATE INDEX IDX_SCENE2 ON SCENE (fkfilm)
CREATE UNIQUE INDEX IDX_AFFICHE1 ON AFFICHE (pkaffiche)
CREATE INDEX IDX_AFFICHE2 ON AFFICHE (fkphoto)
CREATE INDEX IDX_AFFICHE3 ON AFFICHE (fkfilm)
CREATE UNIQUE INDEX IDX_ALBUM1 ON ALBUM (pkalbum)
CREATE INDEX IDX_ALBUM2 ON ALBUM (fkphoto)
CREATE INDEX IDX_ALBUM3 ON ALBUM (fkacteur)
CREATE UNIQUE INDEX IDX_PHOTO1 ON PHOTO (pkphoto)
CREATE INDEX IDX_PHOTO2 ON PHOTO (commentaire)
ALTER TABLE ACTEUR
   ADD CONSTRAINT  C1       CHECK(datemort >= datenaissance)
ALTER TABLE REALISATEUR
   ADD CONSTRAINT  C2       CHECK(datemort >= datenaissance)
ALTER TABLE LOCATION
   ADD CONSTRAINT  C3       CHECK(dateretour >= datelocation)
CREATE INDEX IDX_EXEMPLAIRE3 ON EXEMPLAIRE (codebarre)
```

La dernière ligne crée un index sur le codebarre, ce qui nous permettra de retrouver facilement la pk et donc l'exemplaire.

2. PREMIERE TABLE : CLIENT.

2.1. Introduction.

Avant toute chose, il faut se préparer à travailler avec ordre pour bien structurer l'application et ne rien oublier. Mon habitude est de toujours commencer par une table qui ne nécessite pas de liaison (une table sans clé étrangère). CLIENT correspond à cette attente.

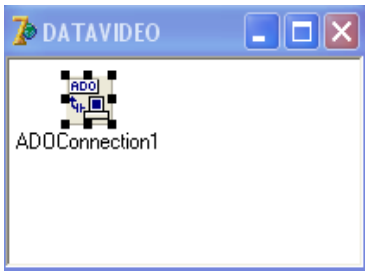
Avant tout, il faut créer le projet (extension dpr). Fichier > Nouveau > Application.

Je crée un répertoire mesvideos dans lequel je vais placer tous mes fichiers (y compris les icônes,...). J'y sauve le projet sous DBVIDEO.dpr et la première unité sous UCLIENT.pas. Je nomme (name) la forme FCLIENT et je lui donne comme titre (caption) Gestion des Emprunteurs.
Je sauvegarde le tout : icône

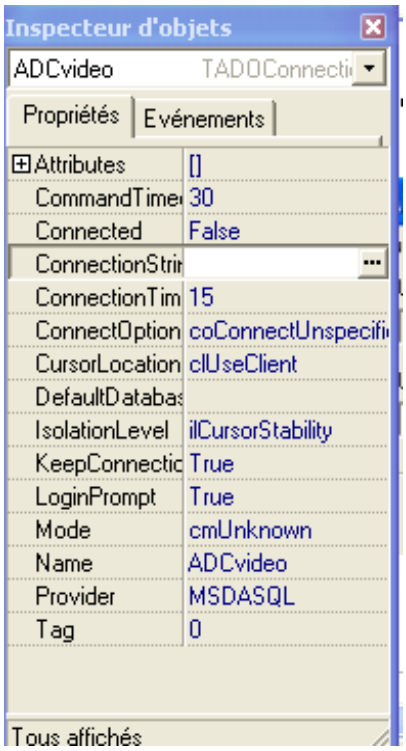
A partir de maintenant, chaque sauvetage se fera grâce à cette icône et tout sera sauvegardé.
Pour placer tout ce qui concerne la connexion et le travail avec la base de données, je crée un Module de Données. (datamodule en anglais). Fichier > Nouveau > Module de Données. Je le nomme DATAVIDEO (name dans l'Inspecteur d'objets) puis je sauvegarde le tout sous UDATA.pas.
Ce datamodule comprendra toutes les icônes de liaison avec la base de données et peut-être aussi des procédures et fonctions utiles.

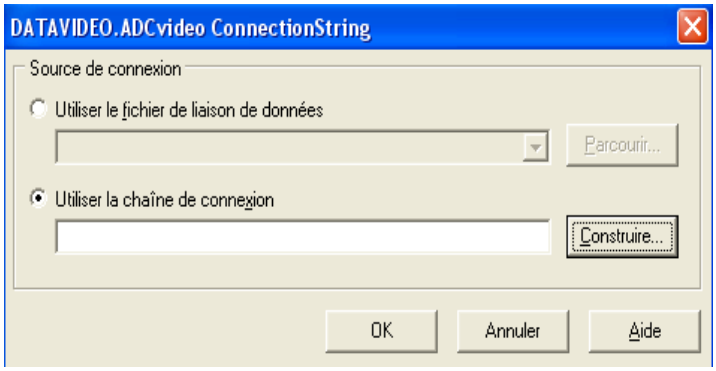
2.2. La connexion avec la base de données.

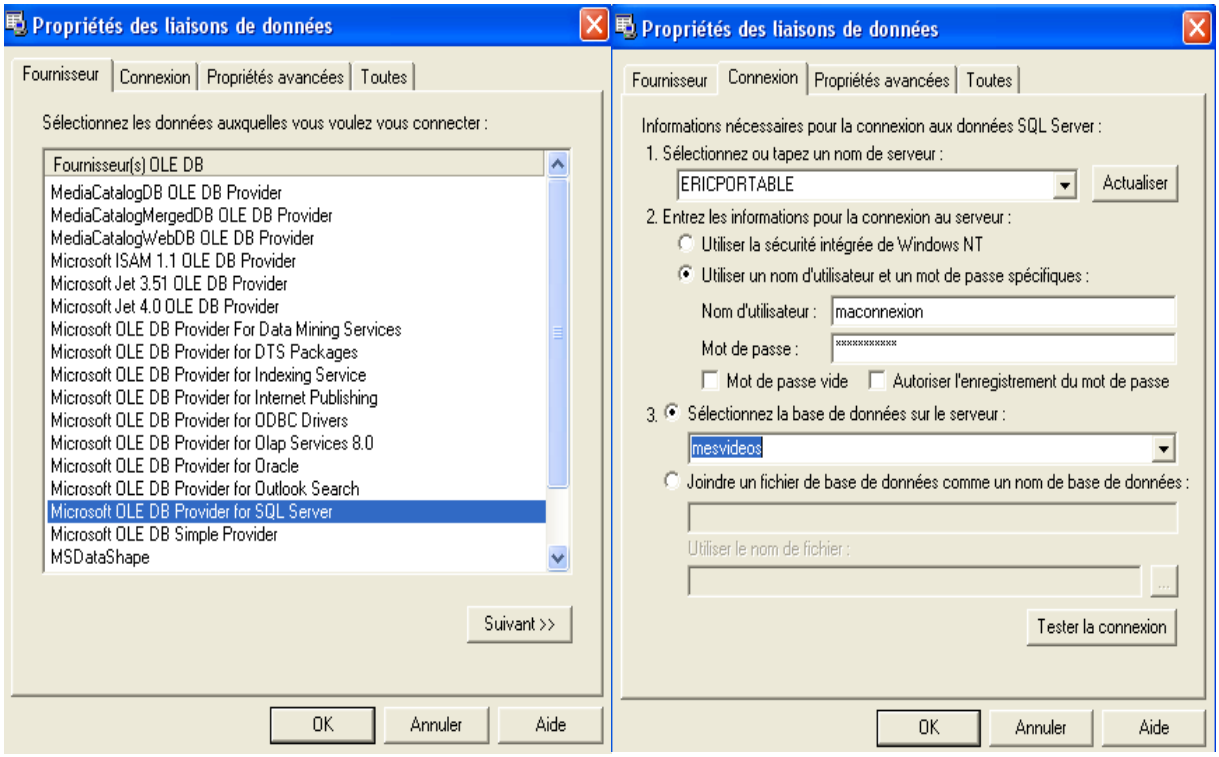
Je vais placer l'élément connexion dans DATAVIDEO. Cette connexion se trouve dans l'onglet ADO. Mon DATAVIDEO devient :



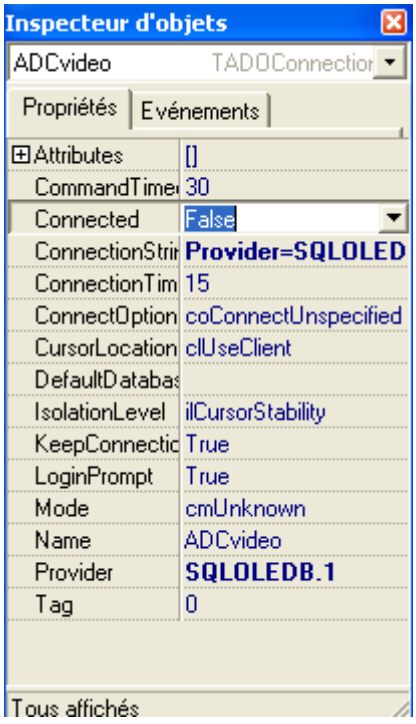
Vous cliquez sur les trois points à droite de ConnectionString et vous obtenez la fenêtre de dialogue où vous choisissez Construire. Vous obtenez ensuite la fenêtre suivante.



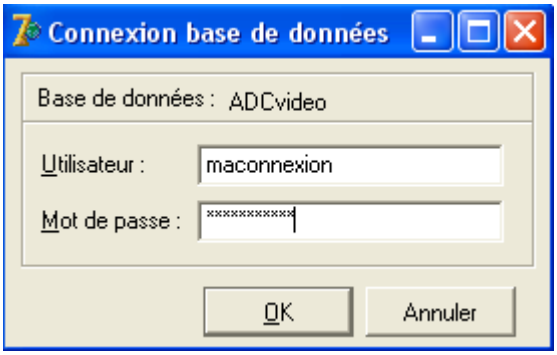




Microsoft OLE DB Provider for SQL Server et dans l'onglet connexion, sélectionnez votre serveur local et votre connexion authentifiée qui dispose de tous les pouvoirs (il est évident que dans la version déployée, nous utiliserons une connexion mois puissante). Testez la connexion. Vous devriez avoir un message OK. Une fois revenu dans l'inspecteur d'objet, prenez bien soin de laisser l'option LoginPrompt à true, ce qui impose d'entrer le mot de passe à chaque activation de la connexion. Quand vous modifierez la propriété connected à true, ce mot de passe vous sera demandé.



Remarque : quand vous quittez delphi, pensez à déconnecter chaque fois (mettez la propriété connected à false) car si un problème de connexion avec le serveur arrivait, vous ne pourriez entrer dans l'application. Lors du début du travail, vous vous reconnectez en mettant la propriété à true.



Encore un conseil : il est assez intéressant de voir apparaître les différentes unités. Tapez CTRL F12 et ajoutez DBVIDEO. Vous avez ainsi accès à chaque unité. Ensuite, pour retourner sur une forme, vous cliquez sur l'unité correspondante puis sur F12.

UCLIENT | UDATA | DBVIDEO

```
unit UCLIENT;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Co
  Dialogs;

type
  TFCLIENT = class(TForm)
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  FCLIENT: TFCLIENT;

implementation

{$R *.dfm}

end.
```

1: 1 Insertion Code Diagramme

2.3. La procédure stockée de listage des clients.

Dans une forme, j'apprécie tout particulièrement de disposer d'une grille reprenant quelques enregistrements et des zones de saisie ou d'affichage des contenus des champs. Il faut choisir soigneusement les champs à afficher pour éviter d'utiliser des ascenseurs horizontaux. Dans la table client, les colonnes intéressantes sont :
nom – prénom – localité – email.
Nous construirons la procédure stockée en fonction de ces choix.

Avec une procédure comme celle-ci, vous pouvez lister par ordre croissant ou décroissant sur une colonne. Il suffira de demander qu'un clic sur le titre de colonne relance cette procédure avec le nom de colonne et l'ordre demandé.

Attention : ceci est gourmand en temps car la transaction est relancée chaque fois et parfois sur des champs non indexés.

C'est à vous à décider si oui ou non on autorise ce type de propriété.

A nous de voir si cette procédure fonctionne depuis notre application.

Requête - ericportable.mesvideos.ERICPORTABLE\Eric ...

```
CREATE proc listeclient
@nomdecolonne varchar(30),@ordretri char(1)
as
if @ordretri = '-' goto decroissant
select client.pkclient,client.nom,client.prenom,
       client.adresse,client.cp,client.localite,
       client.tel,client.email
from client
order by
   case @nomdecolonne
        when 'nom'          then nom
        when 'prenom'       then prenom
        when 'localite'     then localite
        when 'email'        then email
   end
goto fin
decroissant:
select client.pkclient,client.nom,client.prenom,
       client.adresse,client.cp,client.localite,
       client.tel,client.email
from client
order by
   case @nomdecolonne
        when 'nom'          then nom
        when 'prenom'       then prenom
        when 'localite'     then localite
        when 'email'        then email
   end
desc
fin:
```

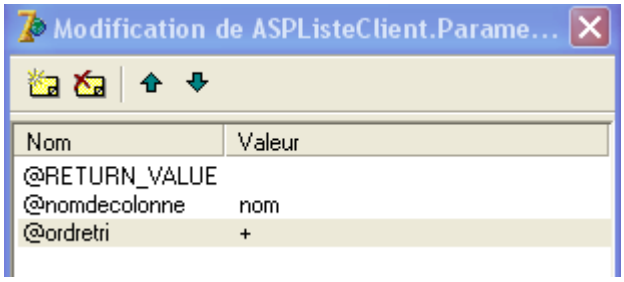
La ou les commandes ont réussi.

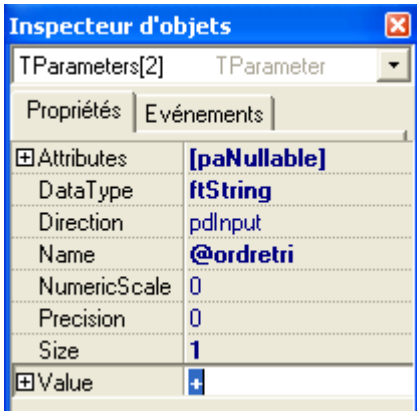
Grilles Messages

ericportat ERICPORTABLE\Eric (55) mesvideos 0:00:00 0 lignes Lg 1, Col 32

Dans le Module de Données, nous allons ajouter un composant ADO : ADOStoredProc. Propriétés :

- name : ASPListeClient
- connection : ADCVideo
- procedurename : listeclient
- parameters : pour @ordretri, nous aurons :



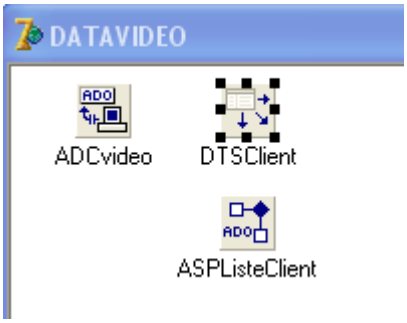


- parameters : pour @nomdecolonne : nom

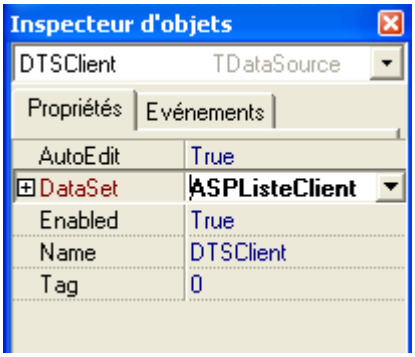
N'oubliez pas de tout enregistrer.

Pour tester la procédure stockée, nous allons placer un dbgrid sur la forme FCLIENT. Mais il nous faut d'abord placer un Datasource (onglet Accès BD) dans le Datamodule.

Rappel : le Datasource distribue le contenu de la source de données vers tous les composants bases de données placés sur une forme. Mon Datamodule devient :



et le Datasource a pour propriétés :



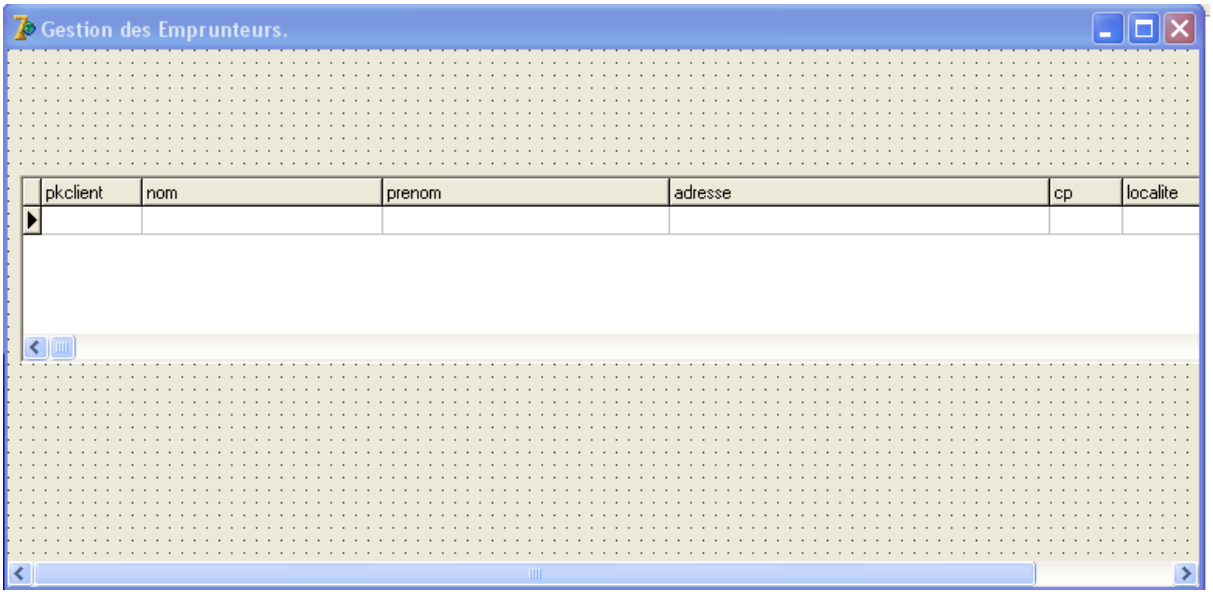
Ne pas oublier de tout sauvegarder.

2.4. Visualisation dans une grille (dbgrid).

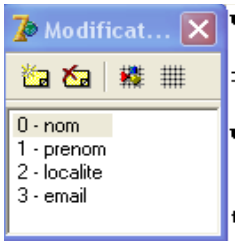
Cette fois, nous allons placer le Dbgrid (onglet Contrôle BD) sur la forme DCLIENT. Etendez la grille pour qu'elle occupe la largeur de la forme. Laissez de la place au-dessus pour les menus et icônes et de la place en-dessous pour d'autres composants. Voici les propriétés de cette grille :

name : DBGClient
datasource : DATAVIDEO.DTSClient
readonly : true (ce qui interdit l'écriture dans la grille).

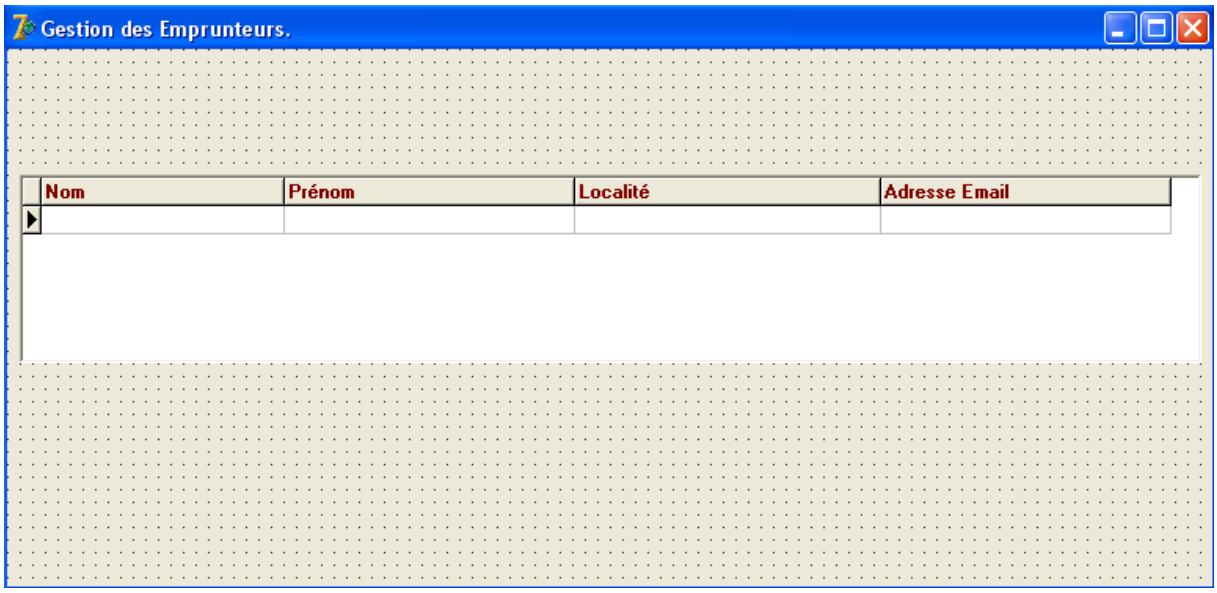
Pour tester si la liaison est effective, retourner dans le Module de données et forcez à true la propriété active d'ASPListeClient. Quand vous aurez procédé au lancement du projet, vous verrez apparaître les colonnes de la base de données dans la grille. On voit ainsi :



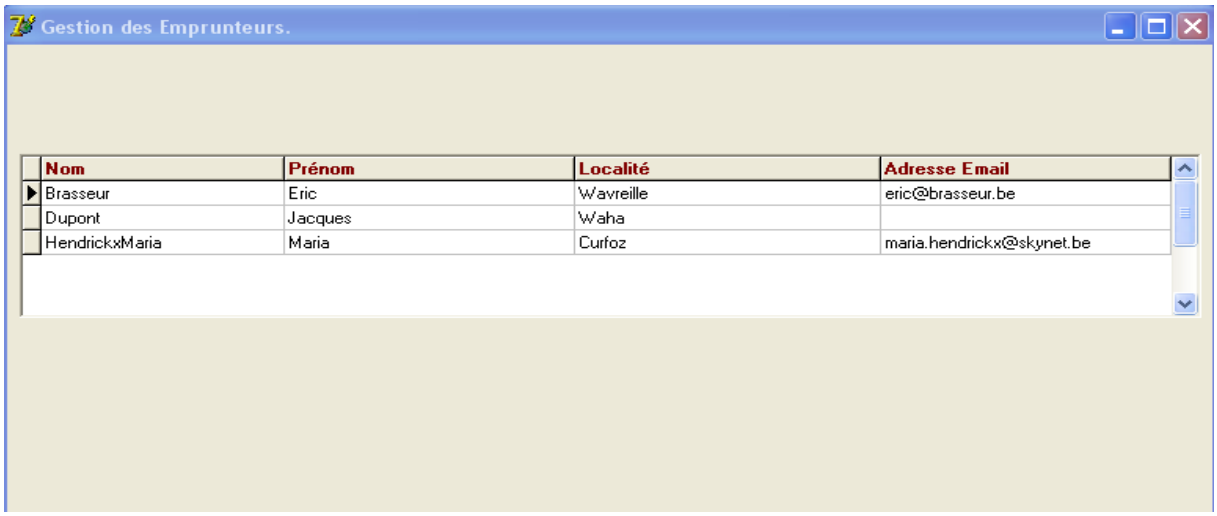
Nous allons limiter le nombre de colonnes à quatre et changer les titres.
Clic droit sur la grille > Editeur de Colonnes > cliquez sur l'icône Ajouter tous les champs > sélectionnez les colonnes à supprimer > supprimez ces colonnes.



Un clic droit sur nom ouvre son Inspecteur d'Objets. > cliquez sur title > la propriété name devient Nom > profitez-en pour changer la couleur du titre et mettre en gras. Faîtes de même avec les autres champs.
On obtient la fenêtre ci-dessous :

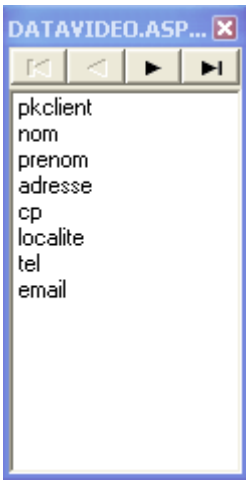


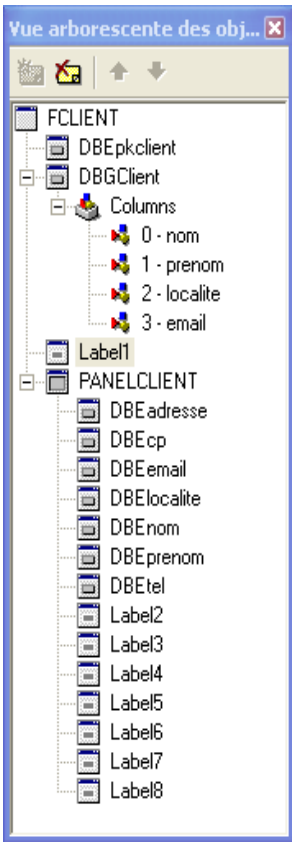
Pour remplir un peu la grille, rendons-nous dans l'Analyseur de Requêtes et ouvrons la table Client. Ajoutons-y quelques clients. Une fois lancé, le programme nous donne :



2.5. Zones de saisies.

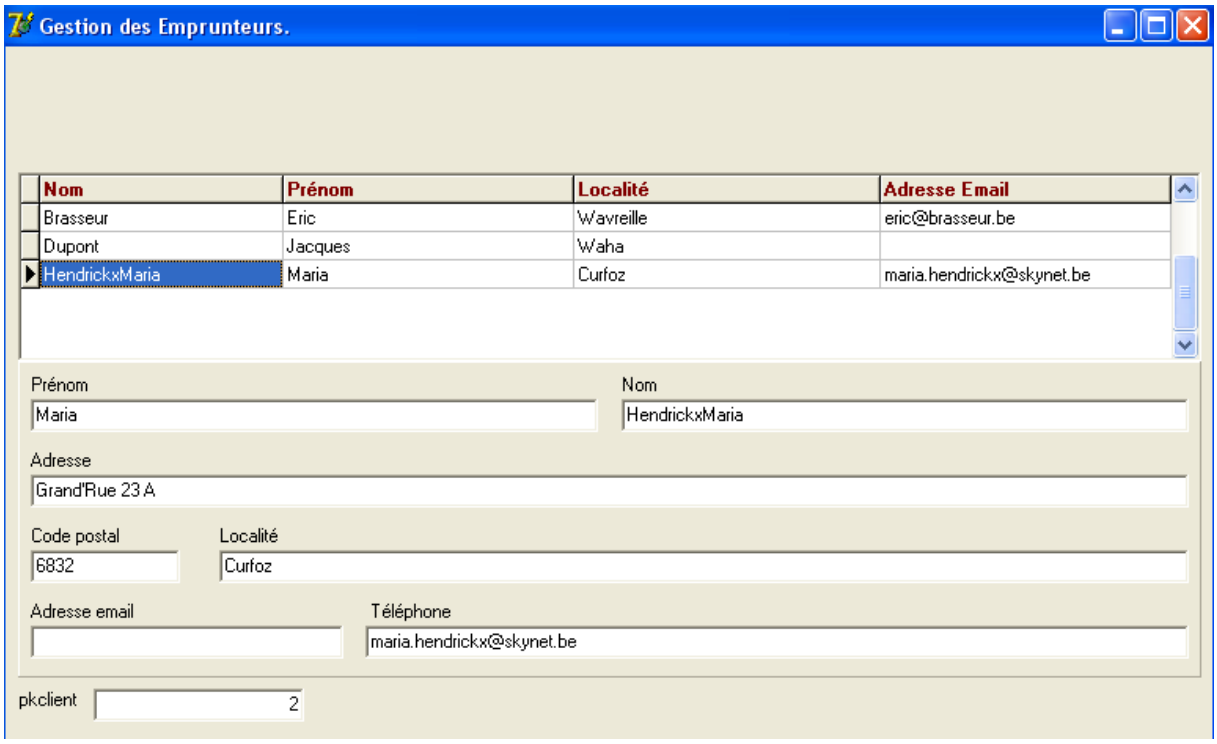
Nous allons maintenant ajouter des zones de saisie pour chaque champ de la table CLIENT, y compris les champs qui ne figurent pas dans la table. Voici la technique automatisée.
1° Agrandissez la forme FCLIENT pour qu'elle occupe un maximum de la hauteur de l'écran.
2° Placez un panel (onglet standard) sous la grille et agrandissez-le pour qu'il occupe tout le bas. Nommez le PANELCLIENT et mettez son caption à blanc.
3° Dans DATAVIDEO, un clic droit sur ASPListeClient > éditeur de champs > de nouveau un clic droit dans la fenêtre qui vient d'être ouverte > ajouter tous les champs
4° Sélectionnez tous les champs
5° Effectuez un drag and drop vers FCLIENT dans PANELCLIENT.
6° Replacez les éléments pour obtenir une figure bien arrangée. Le champ pkclient peut être mis à l'écart car il sera caché à l'utilisateur quand nous terminerons le développement de l'application.
7° Renommez les DBEDIT par DBE + le nom du champ associé.
8° Pas la peine de renommer les labels.
9° Profitez-en pour modifier les taborder pour suivre l'ordre des zones de saisie.
10° Visualisez sur l'arborescence des objets votre forme FCLIENT.





Il est pratique de disposes ses composants dbedit et labels dans un panel car vous pourrez déplacer tous les composants en déplaçant le panel. Tous les composants d'un panel y sont ancrés. De plus, ceci vous permet de changer la couleur de fond ou la police pour tous les composants en une seule opération.

Vous devriez avoir quelque chose du genre :

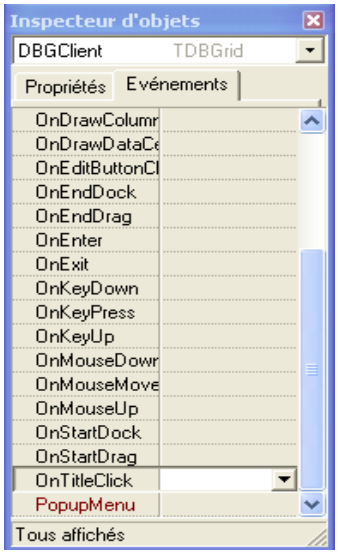


2.6. Rendre plus lisible et plus efficace une grille de données.

Jusque maintenant, nous n'avons pas encore modifié le code. Tout a été généré par DELPHI lui-même. Voici deux procédures qui vont nous permettre d'effectuer le tri en cliquant sur l'en-tête de colonne et aussi de dessiner les lignes en couleurs alternées. Commençons par les titres de colonnes.

Si je clique une fois sur la grille, je vois apparaître son inspecteur d'objets. Je sélectionne la colonne événements et je double clique à droite dans l'événement OnTitleClick, ce qui ouvre le code avec le nom de la procédure.

Vous trouverez ci-dessous le code de cette procédure et quelques explications.



1° Dans les variables (juste au-dessus de l'implémentation), insérez le ligne suivante :
valnom, valprenom, vallocalite, valemail : char ; Nous allons nous servir de ces variables pour retenir le sens du tri sur les différentes colonnes (tri croissant ou décroissant).

2° Si l'ensemble de données est vide, ma procédure ne s'exécute pas car son but est de trier suivant l'en^tête de colonne puis de se replacer sur l'enregistrement actif avant le tri. Recordcount me permet de savoir combien d'enregistrements figurent dans mon ensemble de données.

3° La position active avant le tri est sauvegardée dans la variable cle. La valeur provient de dbepkclient.

4° Il faut désactiver l'ensemble de données pendant la manipulation (active := false).

5° Je passe le paramètre @nomdecolonne avec comme valeur le nom de colonne (column.fieldname) sur laquelle j'ai effectué un clic sur le titre.

6° Je teste la valeur de l'ordre de tri associé à cette colonne, je l'inverse et je transmets le nouvel ordre à la procédure stockée par le paramètre @ordretri.

7° Je réactive mon ensemble de données (dataset en anglais).

8° Je me positionne sur la valeur sauvegardée dans cle.

UCLIENT | UDATA

```
procedure TFCLIENT.DBGClientTitleClick(Column: TColumn);
var
   cle : integer;
begin
   with DATAVIDEO.ASPlisteclient do
      begin
        if (recordcount <> 0) then
          begin
            cle := strtoint(dbepkclient.text);
            active := false;    // retour du curseur
            parameters.parambyname('@nomdecolonne').value :=  column.FieldName;
            if column.FieldName = 'localite' then
              begin
                 if vallocalite = '+' then vallocalite := '-'
                                      else vallocalite := '+';
                 parameters.ParamByName('@ordretri').value := vallocalite;
              end;
            if column.FieldName = 'email' then
              begin
                 if valemail = '+' then valemail := '-'
                                   else valemail := '+';
                 parameters.ParamByName('@ordretri').value := valemail;
              end;
            if column.FieldName = 'nom' then
              begin
                 if valnom = '+' then valnom := '-'
                                 else valnom := '+';
                 parameters.ParamByName('@ordretri').value := valnom;
              end;
            if column.FieldName = 'prenom' then
              begin
                 if valprenom = '+' then valprenom := '-'
                                    else valprenom := '+';
                 parameters.ParamByName('@ordretri').value := valprenom;
              end;
            active := true;
            locate('pkclient',cle,[]);
          end;
      end;
end;
```

Je vais maintenant utiliser une procédure pour dessiner les lignes de ma grille dans des couleurs alternées.

1° Cliquez une fois sur la grille.

2° Choisir la colonne événement dans l'inspecteur d'objet.

3° Double cliquez à droite dans l'événement OnDrawColumnCell.

4° Vous encodez ensuite le code suivant.

UCLIENT

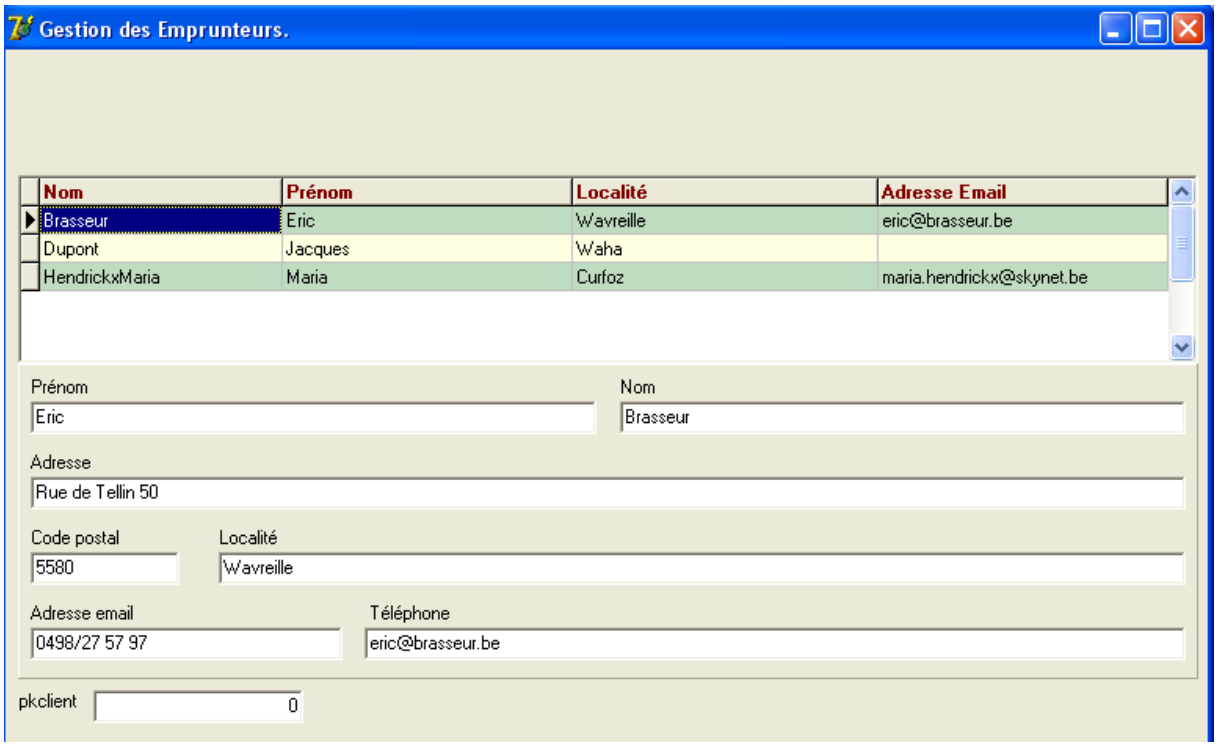
```
      end;
end;

procedure TFCLIENT.DBGClientDrawColumnCell(Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumn;
  State: TGridDrawState);
var
   grille : tdbgrid;
begin
   grille := (sender as tdbgrid);
   if(grille.DataSource.DataSet.RecNo mod 2) =0
      then grille.Canvas.Brush.Color := clinfobk
      else grille.Canvas.Brush.Color := clmoneygreen;
   if state = [gdselected,gdfocused] then grille.Canvas.Brush.Color := clnavy;
   grille.DefaultDrawColumnCell(rect,datacol,column,state);
end;

end.
```

Explications :

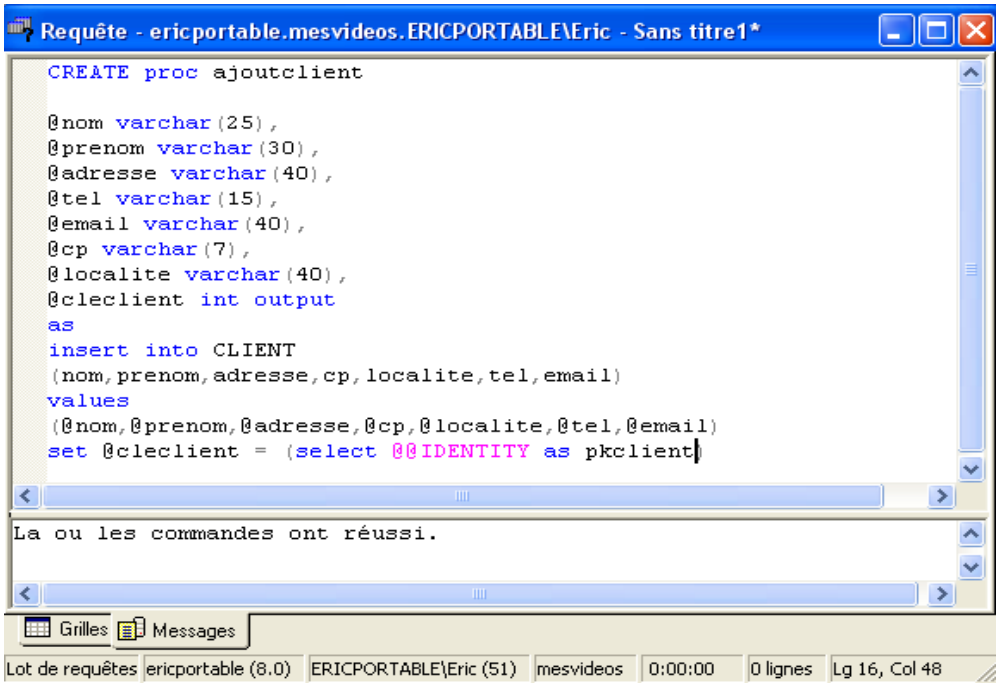
1° grille est une variable de type tdbgrid
2° cette variable reçoit le nom de la grille émettrice
3° si le numéro d'enregistrement (recno) est divisible par 2, alors la couleur de fond est clinfobk. Vous êtes libre de choisir une autre couleur ou même d'en définir une vous-même.
4° si le numéro d'enregistrement est impair, la couleur de fond est clmoneygreen.
5° l'élément qui a le focus (celui sur lequel vous avez cliqué en dernier) sera bleu marine.



Notre application commence à avoir un look plus professionnel.
Ne vous découragez pas en lisant ces premières lignes de code : elles figurent parmi les plus difficiles.

2.7. Ajouter un client.

L'ajout d'un client se fera uniquement dans les zones de saisie et par l'intermédiaire d'une procédure stockée que nous allons encoder dans l'analyseur de requêtes. Bien entendu, ceci n'est possible que pour une connexion autorisée à le faire.



Je déclare 7 variables qui recevront 7 paramètres du programme. @cleclient sera au contraire un paramètre renvoyé au programme pour lui donner le numéro de clé primaire généré par la procédure. C'est la dernière ligne qui effectue cette opération. Le reste est connu.
Nous allons ajouter le composant ASPajoutclient dans le module de données.
Name : ASPAjoutClient
Connection : ADCVideo
ProcedureName : ajoutclient
Ne pas activer : nous l'activerons juste au moment de l'ajout.

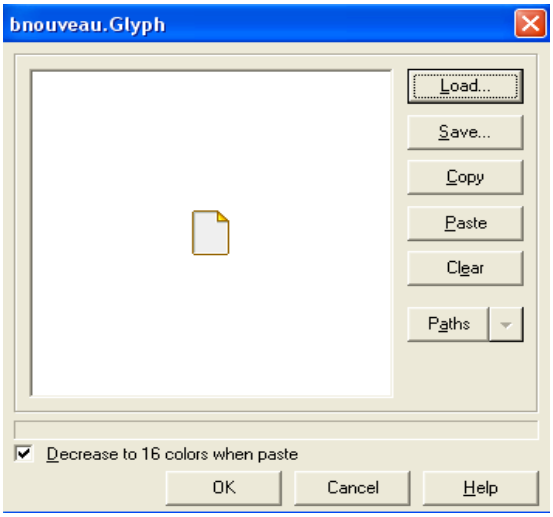
Dans FCLIENT, nous allons ajouter un panel (appelé PANELICONE) qui contiendra les icônes d'ajout, modification, suppression et recherche d'enregistrements. Plaçons-le juste au dessus à gauche de la grille.
Il nous faut aussi trouver des icônes à placer sur les formes. Voici PANELICONE avec 4 icônes Les icônes sont des speedbutton (onglet Supplément).

1° Vous les placez sur le panel.

2° Vous cliquez sur la propriété glyph et avec load, vous cherchez l'image bitmap qui sera superposée au bouton.

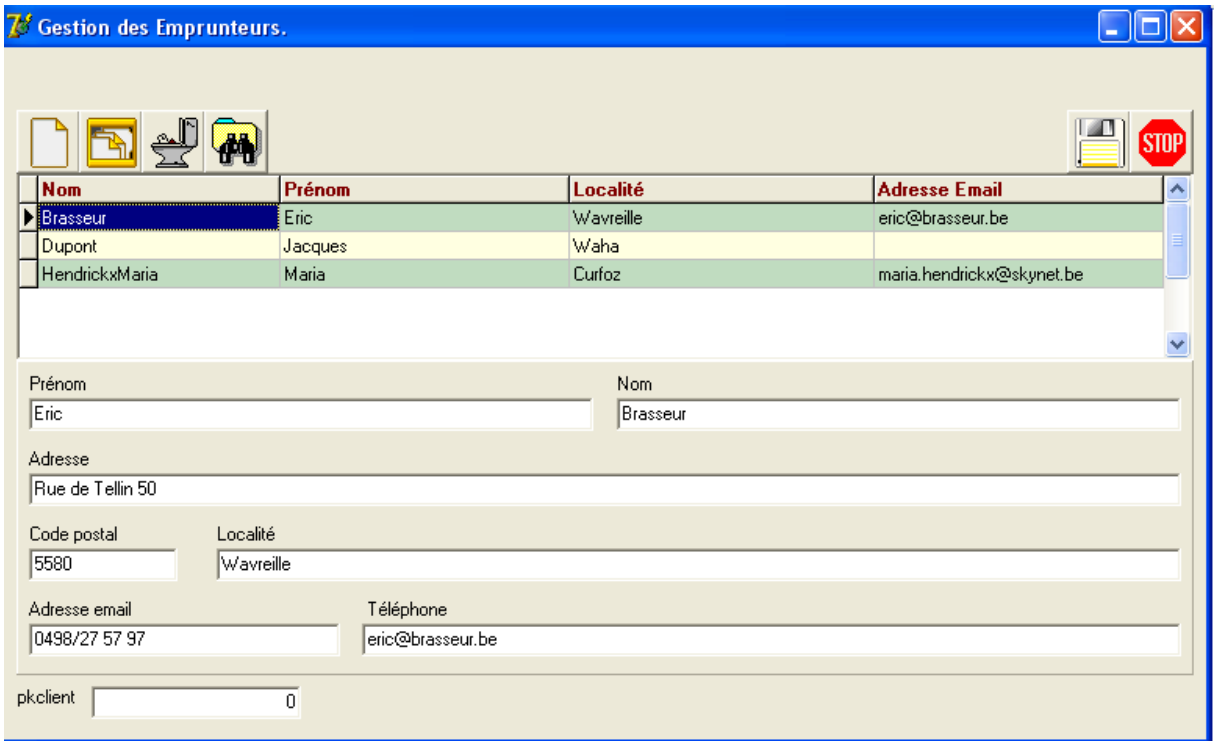
3° Vous les nommez successivement :

- bnouveau
- bmodifie
- bsupprime
- bcherche



N'oubliez pas de sauvegarder. Nous allons maintenant faire de même avec un nouveau panel que nous nommerons PANELVALIDATION avec 2 icônes pour enregistrer ou annuler.

Vous nommez les boutons benregistre et bannule. Puis vous placez PANELVALIDATION au-dessus à droite de la grille. Nous devrions avoir quelque chose du genre :



Avant de créer l'événement d'ajout, il faut penser aux implications : pendant un ajout ou une modification, je conseille :

- d'interdire l'accès à la grille
- d'ouvrir les zones de saisie en écriture seulement au moment des modifications ou ajouts et de les laisser en lecture seule autrement
- de faire disparaître le PANELICONE pendant l'ajout et la modification et de faire apparaître le PANELVALIDATION à ce moment.
- Autrement, la situation de ces deux panels doit être inversée.

Je vous conseille de créer les quatre procédures suivantes (non liées à des événements).

```
procedure tfclient.ouvreoption;
begin
  panelicone.Visible := true;
  panelvalidation.Visible := false;
end;

procedure tfclient.fermeoption;
begin
  panelicone.Visible := false;
  panelvalidation.Visible := true;
end;
```

Fermeoption interdit d'utiliser les quatre icônes d'action de PANELICONE et active PANELVALIDATION
Ouvreoption agit dans le sens inverse

```
procedure tfclient.ecrireoui;
begin
  dbenom.ReadOnly := false;
  dbeprenom.ReadOnly := false;
  dbeadresse.ReadOnly := false;
  dbecp.readonly := false;
  dbeemail.ReadOnly := false;
  dbetel.ReadOnly := false;
  dbelocalite.ReadOnly := false;
end;

procedure tfclient.ecrirenon;
begin
  dbenom.ReadOnly := true;
  dbeprenom.ReadOnly := true;
  dbeadresse.ReadOnly := true;
  dbecp.readonly := true;
  dbelocalite.ReadOnly := true;
  dbeemail.ReadOnly := true;
  dbetel.ReadOnly := true;
end;
```

Ecrireoui permet l'écriture dans les zones de saisie.
Ecrirenon interdit l'écriture dans les zones de saisie.

Pour que ces quatre procédures fonctionnent et qu'elles ne sont liées à aucun événement direct du programme, nous devons les signaler dans la partie INTERFACE du code. Les voici avec les procédures événementielles :

```
    procedure DBGClientTitleClick(Column: TColumn);
    procedure DBGClientDrawColumnCell(Sender: TObject; const Rect: TRect;
      DataCol: Integer; Column: TColumn; State: TGridDrawState);
    procedure bnouveauClick(Sender: TObject);
    procedure ouvreoption;
    procedure fermeoption;
    procedure ecrireoui;
    procedure ecrirenon;
```

Il faut les ajouter manuellement. Voici maintenant le code de la procédure d'ajout. A encoder après un double-clic sur le bouton bnouveau.

```
procedure TFCLIENT.bnouveauClick(Sender: TObject);
begin
   dbeprenom.SetFocus;
   datavideo.ASPlisteclient.insert;
   dbgclient.Enabled := false;
   fermeoption;
   ecrireoui;
end;
```

Je place le focus sur dbeprenom ce qui permet d'écrire immédiatement dans la zone.
Je signale à l'ensemble de données ASPlisteclient situé dans datavideo qu'il doit se mettre en mode d'insertion (d'ajout).
Je bloque l'accès à la grille.
Je modifie la visibilité des panels.
Je permets l'écriture dans les zones de saisie.
Nous sommes en mesure d'ajouter un enregistrement mais pas encore de le sauver . Il va falloir donner du code au bouton d'enregistrement.

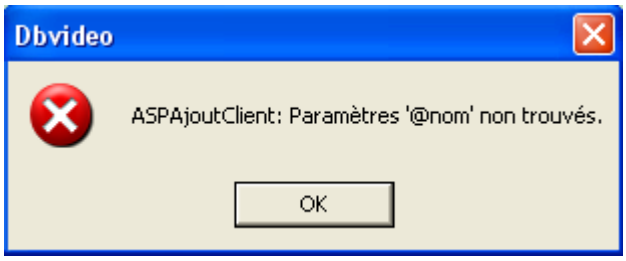
Auparavant, dans la rubrique var au-dessus de l'implémentation, ajoutez la variable cleclient : integer;

Cette variable récupérera la clé primaire générée par MS SQL SERVER et permettra de se situer sur l'enregistrement ajouté.

```
procedure TFCLIENT.benregistreClick(Sender: TObject);
begin
   with datavideo.ASPajoutclient do
      begin
         parameters.parambyname('@nom').value :=  dbenom.text;
         parameters.ParamByName('@prenom').Value := dbeprenom.Text;
         parameters.parambyname('@adresse').value :=  dbeadresse.text;
         parameters.parambyname('@tel').value :=  dbetel.text;
         parameters.parambyname('@email').value :=  dbeemail.text;
         parameters.ParamByName('@localite').Value := dbelocalite.text;
         parameters.ParamByName('@cp').Value := dbecp.text;
         execproc;
         cleclient := parameters.parambyname('@cleclient').value;
      end;
  benregistre.Enabled := false;
  datavideo.ASPlisteclient.Locate('pkclient',cleclient,[]);
  dbgclient.Enabled := true;
  ecrirenon;
  ouvreoption;
end;
```

Nous envoyons à la procédure stockée les paramètres qu'elle attend. Nous lançons l'exécution de la procédure (execproc) et nous récupérons la clé générée. Nous nous plaçons sur l'enregistrement ajouté et nous rétablissons la grille et la situation d'avant le début de l'ajout.

Si vous lancez l'application et que vous enregistrez, vous aurez un message d'erreur.



Ceci est facile à corriger : dans DATAVIDEO, cliquez sur ASPajoutclient et allez dans les paramètres (dans l'inspecteur d'objets).

Pour les variables varchar, dans la partie value, complétez type par OleStr.
Pour @cleclient, complétez type par integer et value par 0.

Cette action n'est pas réalisée par défaut au moment de l'enregistrement du composant.

Recommencez l'ajout et cette fois, cela fonctionne.

Toutefois, si vous ne complétez pas les champs obligatoires (required ou not null), votre application vous autorise à le faire alors que MS SQL SERVER vous donnera un message d'erreur. Il vaut mieux prévenir ce type d'erreur car le message n'est pas très compréhensible par la majorité des utilisateurs.
La solution la plus efficace est de ne rendre le bouton enregistrer utilisable (enabled) que si les zones non nulles sont remplies.
Il suffit de cliquer une fois sur la zone dbenom et de double cliquer à droite de l'événement onchange puis d'encoder :

UCLIENT

```
procedure TFCLIENT.DBEnomChange(Sender: TObject);
begin
   benregistre.Enabled := (trim(dbenom.Text) <> '');
end;

end.
```

Le bouton d'enregistrement ne sera utilisable que si la zone dbenom est non vide.
Il faut encore initialiser les variables val... à l'ouverture de la forme et cacher le PANELVALIDATION.
Nous allons utiliser l'événement FORMCREATE de la forme.
Double cliquez dans une zone libre de la forme FCLIENT > encodez :

```
procedure TFCLIENT.FormCreate(Sender: TObject);
begin
  panelvalidation.Visible := false;
  benregistre.Enabled := false;
  valnom := '-';
  valprenom := '-';
  valemail := '-';
end;
```

Et le code du bouton d'annulation :

```
procedure TFCLIENT.bannuleClick(Sender: TObject);
begin
   datavideo.ASPlisteclient.Cancel;
   benregistre.Enabled := false;
   ecrirenon;
   ouvreoption;
end;
```

Il faut arrêter le mode d'insertion de ASPlisteclient et rendre inutilisable le bouton d'enregistrement.

2.8. Modifier les renseignements d'un client.

Il nous faut d'abord une procédure stockée créée dans l'analyseur de requêtes.

```
CREATE proc modifclient
@nom varchar(25),
@prenom varchar(30),
@adresse varchar(40),
@cp varchar(7),
@localite varchar(40),
@tel varchar(15),
@email varchar(30),
@cleclient int
as
update CLIENT
set
nom = @nom,prenom = @prenom,adresse = @adresse,tel = @tel,
email = @email,cp = @cp,localite = @localite
where
PKclient = @cleclient
```

```
La ou les commandes ont réussi.
```

Ajoutons une ADOStoredProc dans DATAVIDEO avec les propriétés :
name = ASPModifClient
connection : ADCvideo
procedurename = modifclient
parameters : voir paragraphe précédent

Cette fois, la procédure stockée ne renvoie pas de valeur de clé, au contraire elle reçoit le numéro de clé à modifier. Il s'agit donc pour la procédure stockée d'un paramètre d'entrée.

Nous pouvons double cliquer sur l'icône bmodifie pour lancer le code :

```
procedure TFCLIENT.bmodifieClick(Sender: TObject);
begin
   if (trim(dbepkclient.Text) <> '') then
      begin
        cleclient := strtoint(dbepkclient.text);
        dbenom.SetFocus;
        datavideo.ASPlisteclient.edit;
        fermeoption;
        ecrireoui;
      end;
end;
```

Ici, nous sauvegardons la position courante dans la table puis nous mettons l'ensemble de données en position de modification (edit). Il faut maintenant adapter le bouton d'enregistrement pour lui dire qu'il doit faire intervenir la procédure de modification et non celle d'ajout.

```
procedure TFCLIENT.benregistreClick(Sender: TObject);
begin
   if (datavideo.ASPlisteclient.State = dsinsert) then
      begin
        with datavideo.ASPajoutclient do
          begin
            parameters.parambyname('@nom').value :=  dbenom.text;
            parameters.ParamByName('@prenom').Value := dbeprenom.Text;
            parameters.parambyname('@adresse').value :=  dbeadresse.text;
            parameters.parambyname('@tel').value :=  dbetel.text;
            parameters.parambyname('@email').value :=  dbeemail.text;
            parameters.ParamByName('@localite').Value := dbelocalite.text;
            parameters.ParamByName('@cp').Value := dbecp.text;
            execproc;
            cleclient := parameters.parambyname('@cleclient').value;
          end;
      end
   else
      begin
        cleclient := strtoint(dbepkclient.Text);
        with datavideo.ASPmodifclient do
          begin
             parameters.ParamByName('@cleclient').value := cleclient;
             parameters.parambyname('@nom').value :=  dbenom.text;
             parameters.ParamByName('@prenom').Value := dbeprenom.Text;
             parameters.parambyname('@adresse').value :=  dbeadresse.text;
             parameters.parambyname('@tel').value :=  dbetel.text;
             parameters.parambyname('@email').value :=  dbeemail.text;
             parameters.ParamByName('@localite').Value := dbelocalite.text;
             parameters.ParamByName('@cp').Value := dbecp.text;
             execproc;
          end;
      end;
  benregistre.Enabled := false;
  datavideo.ASPlisteclient.Locate('pkclient',cleclient,[]);
  ecrirenon;
  ouvreoption;
end;
```

.

Je commence par tester si je suis en mode insertion ou édition grâce à la propriété dsinsert. Je sais donc quelle procédure stockée je dois lancer. On constate que dans la création, c'est la procedure ASPAjoutClient qui me fournit la clé où me placer après ajout, tandis que dans la modification, la clé est celle de l'enregistrement qu'on vient de modifier, clé sauvegardée au préalable.
Après avoir sauvegardé, vous constaterez un message d'erreur lors de l'exécution.

```
[Erreur] UCLIENT.pas(167): Identificateur non déclaré : 'dsinsert'
[Erreur fatale] DBVIDEO.dpr(6): Impossible de compiler l'unité utilisée 'UCLIENT.pas'
```

Il ne connaît pas dsinsert. Vérifions dans l'aide . Il faut ajouter la librairie DB pour pouvoir l'utiliser.
Dans la ligne uses de l'interface, vous insérer ,DB à la fin. C'est en ordre.

Une autre erreur se produit parfois : quand vous effectuez une modification de la base de données, il arrive que les tables ne soient plus synchronisées (modification par autrui, ...). Il vaut donc mieux actualiser l'état. Nous allons créer une petite procédure qui désactive et réactive la procédure qui liste (ListeClient). C'est une procédure qui n'est pas événementielle. La voici :

```
procedure tfclient.actualise;
  begin
    datavideo.ASPlisteclient.active := false;
    datavideo.ASPlisteclient.active := true
  end;
```

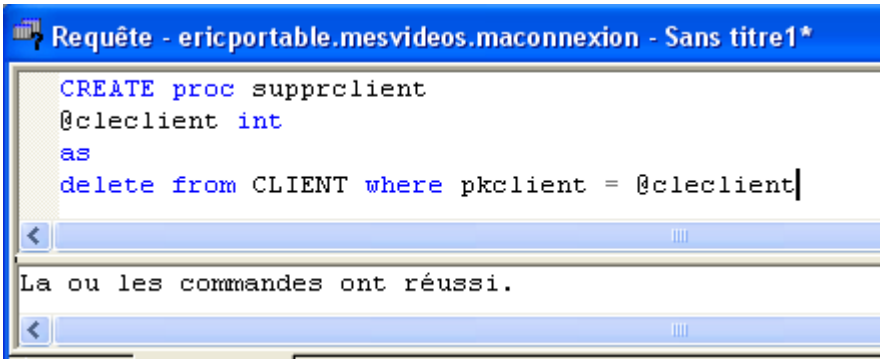
Ne pas oublier de copier la ligne procedure actualise; dans l'interface.

Ajoutons maintenant cette ligne de code juste avant le locate de la la procédure d'enregistrement.

```
benregistre.Enabled := false;
actualise;
datavideo.ASPlisteclient.Locate('pkclient',cleclient,[]);
ecrirenon;
ouvreoption;
```

2.9. Suppression d'un client.

Commençons par la procédure stockée.



Il faudra donc transmettre la clé de l'enregistrement à supprimer. Une question : où se placer après suppression ? Nous en discuterons plus tard.

Plaçons un ADOStoredProc dans le DATAVIDEO avec les propriétés
name : ASPsupprClient
connection : ADCvideo
procedurename : supprclient
parameters : cleclient : integer
Le code associé au bouton bsupprime sera :

```
procedure TFCLIENT.bsupprimeClick(Sender: TObject);
begin
   if (messagedlg('Confirmez-vous la suppression de cet emprunteur ? ',
         mtconfirmation,[mbok,mbcancel],0) = mrok) then
      begin
         with datavideo.ASPsupprclient do
           begin
              parameters.ParamByName('@cleclient').Value := strtoint(dbepkclient.Text);
              execproc;
           end;
         actualise;
      end;
end;
```

Un message de confirmation est lancé avant la suppression. La clé client est passée comme paramètre d'entrée de la procédure stockée. Enfin on actualise.

2.10 Actualisation de la table.

Je propose d'ajouter un bouton sur le PANELICONE pour actualiser la table si un autre utilisateur a effectué des changements sur un autre poste. On dispose déjà de la procédure. Insérons une icône entre la suppression et la recherche.

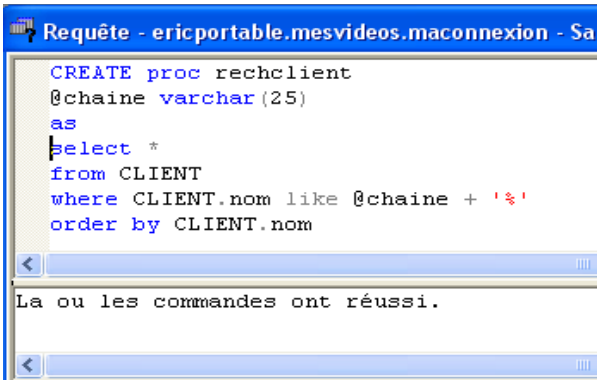
Le code est simple :

```
procedure TFCLIENT.bactualiseClick(Sender: TObject);
begin
  actualise;
end;
```

Cette procédure nous fera perdre la ligne en cours. C'est plus prudent en cas de suppression par un autre de cette ligne.

2.11. Recherche dans la table.

Voici la méthode proposée : nous allons créer une zone de saisie (appelée erecherche) où l'utilisateur tapera le début du nom. Chaque nouvelle lettre réduira le nombre d'enregistrements qui apparaîtront dans la grille. Il faudra écrire une nouvelle procédure stockée de recherche sur le début du nom et lier la grille à cette procédure. A la fin, il faudra rétablir la situation initiale.
Créons un panel (PANELRECHERCHE) qui occupera l'espace entre PANELICONE et PANELVALIDATION. Dans ce panel, la zone de saisie erecherche et un bouton de retour à la situation initiale.
Commençons par la procédure stockée dans l'analyseur de requêtes.



Voici le panel PANELRECHERCHE :

La zone de saisie est ici un edit simple (onglet standard). L bouton appelé bretour est un speedbutton.

Il faut utiliser un événement qui lance la procédure stockée rechclient. Nous utiliserons l'événement onchange de erecherche.

Attention : si vous travaillez sur une grosse base de données sur un réseau, cette procédure est très gourmande.
Plaçons dans le module de données une ADOStoredProc
name : ASPrechclient
connection : ADCvideo
procedurename : rechclient
parameters : OleStr pour la chaîne.

Voici le code qui ouvre PANELRECHERCHE (par clic sur le bouton de recherche) :

```
procedure TFCLIENT.bchercheClick(Sender: TObject);
begin
   fermeoption;
   panelvalidation.Visible := false;
   cleclient := strtoint(dbepkclient.Text);
   panelrecherche.Visible := true;
   datavideo.DTSclient.DataSet := datavideo.ASPrechclient;
   erecherche.SetFocus;
end;
```

Je cache PANELVALIDATION et je sauve ma position dans la table. Je rends visible PANELRECHERCHE et je change le pointage de la grille pour que la grille affiche les résultats de ma recherche. Enfin je prépare l'écriture dans erecherche.

Voici le code de erecherche :

```
procedure TFCLIENT.erechercheChange(Sender: TObject);
begin
  with datavideo.ASPrechclient do
    begin
      active := false;
      parameters.ParamByName('@chaine').Value := erecherche.text;
      execproc;
      active := true;
    end;
end;
```

Il faut désactiver la recherche avant de luis passer le paramètre puis réactiver.

Voici le code du bouton de retour :

```
procedure TFCLIENT.bretourClick(Sender: TObject);
begin
  if (trim(dbenom.Text) <> '') and (trim(dbepkclient.Text) <> '')
        then cleclient := strtoint(dbepkclient.text);
   datavideo.ASPrechclient.Active := false;
   datavideo.DTSclient.DataSet := datavideo.ASPlisteclient;
   panelicone.Visible := false;
   actualise;
   datavideo.ASPlisteclient.Locate('pkclient',cleclient,[]);
   ouvreoption;
   ecrirenon;
   panelrecherche.Visible := false;
end;
```

Attention : si la recherche n'a pas fourni de résultat, il sera impossible de se placer à la position donnée. Ici, on réattache la grille à la procédure de listage et plus à celle de recherche.

Au départ, le PANELRECHERCHE doit être invisible. Prévoyons-le dans FORMCREATE en ajoutant :

panelrecherche.Visible := false;

2.12. Points de sécurité.

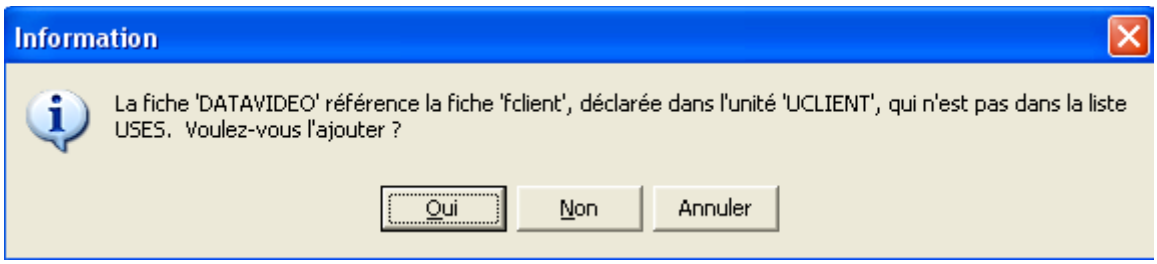
Il faut interdire certaines opérations quand la table est vide : modification, suppression, recherche. En effet, pour intercepter des erreurs, il vaut mieux tester la présence d'enregistrements dans la table et en cas d'absence, rendre inutilisables les boutons liés à ces opérations.

Nous allons ajouter une procédure liée directement à la procédure stockée ASMListeClient. Dans son événement AfterOpen, nous inscrivons la procédure suivante qui compte les enregistrements et bloque les boutons dangereux si ce nombre d'enregistrements est 0.

UCLIENT UDATA

```
procedure TDATAVIDEO.ASPListeClientAfterOpen(DataSet: TDataSet);
begin
   fclient.bmodifie.Enabled := ASPlisteclient.recordcount > 0;
   fclient.bsupprime.Enabled := ASPlisteclient.recordcount > 0;
   fclient.bcherche.Enabled := ASPlisteclient.recordcount > 0;
end;
```

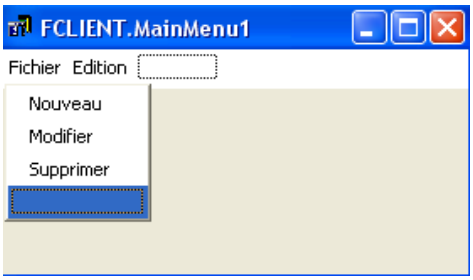
Un message vous parviendra. Vous acceptez car il répertorie fclient dans datavideo.

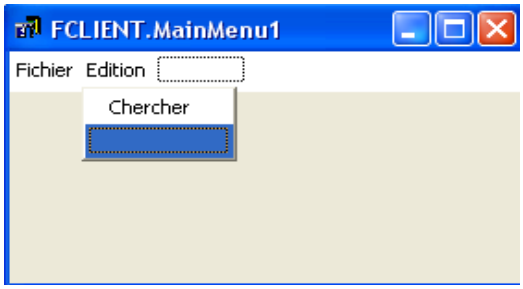


2.13. Ajout d'un menu déroulant.

Il est très utile de disposer d'un menu déroulant. En général, les options du menu sont couvertes par les icônes déjà présentes sur la forme.

Plaçons un composant mainmenu (onglet standard) au-dessus.
Double-cliquez sur ce menu et vous obtenez ceci :





Les titres et les rubriques porteront le nom qu'ils affichent (caption = name).

Pour chaque point du menu, nous allons attacher une action (événement).

Il s'agit de l'événement onclick.
Pour Nouveau, vous choisissez avec la flèche de droite l'événement bnouveauclick.
Pour Modifier, vous choisissez l'événement bmodifieclick.
Pour Supprimer, vous choisissez l'événement bsupprime.
Pour Chercher, vous choisissez l'événement bcherche.

Attention : il faut maintenant penser à désactiver ces options en même temps que les icônes qui y sont associées.
Heureusement, nous disposons des petites procédures personnelles ouvreoption et fermeoption.

```
procedure tfclient.ouvreoption;
begin
  panelicone.Visible := true;
  panelvalidation.Visible := false;
  nouveau.Enabled := true;
  modifier.Enabled := true;
  supprimer.Enabled := true;
  chercher.Enabled := true;
end;

procedure tfclient.fermeoption;
begin
  panelicone.Visible := false;
  panelvalidation.Visible := true;
  nouveau.Enabled := false;
  modifier.Enabled := false;
  supprimer.Enabled := false;
  chercher.Enabled := false;
end;
```

Il faut aussi penser à la procédure Tdatavideo.asplisteclientafteropen.

```
procedure TDATAVIDEO.ASPListeClientAfterOpen(DataSet: TDataSet);
begin
   fclient.bmodifie.Enabled := ASPlisteclient.recordcount > 0;
   fclient.bsupprime.Enabled := ASPlisteclient.recordcount > 0;
   fclient.bcherche.Enabled := ASPlisteclient.recordcount > 0;
   fclient.modifier.Enabled := ASPlisteclient.recordcount > 0;
   fclient.supprimer.Enabled := ASPlisteclient.recordcount > 0;
   fclient.chercher.Enabled := ASPlisteclient.recordcount > 0;
end;
```

2.14. Et l'aide à l'utilisateur ?

Nous allons créer le minimum d'aide au moyen des infobulles. Pour chaque icône, nous allons activer l'option showhint de leur inspecteur d'objets. Puis nous taperons dans la case hint un résumé de l'action de ces boutons . Une infobulle apparaîtra alors en survolant ces boutons. Ce n'est pas un tutoriel ni une aide détaillée mais cela apporte des informations à l'utilisateur. Nous cacherons la clé primaire et nous fixerons les composants définitivement à la fin du développement de l'application.

2.15. Essai et correction.

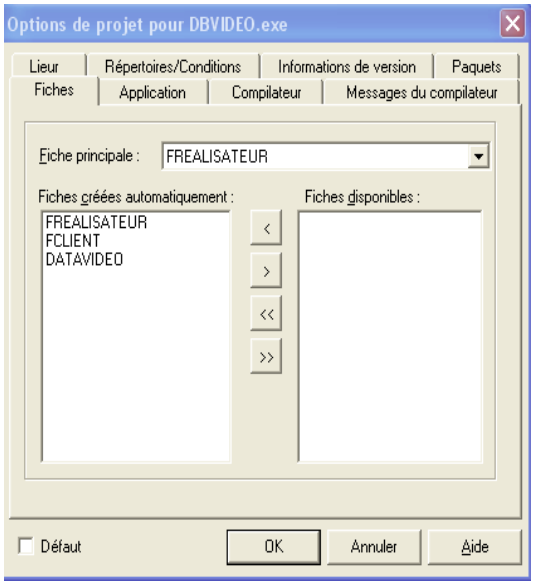
Lors de l'essai, j'ai constaté que la grille ne se réactivait pas après un ajout d'emprunteur. J'i réactivé la grille dans la procédure ouvreoption.

```
procedure tfclient.ouvreoption;
begin
  panelicone.Visible := true;
  panelvalidation.Visible := false;
  nouveau.Enabled := true;
  modifier.Enabled := true;
  supprimer.Enabled := true;
  chercher.Enabled := true;
  dbgclient.Enabled := true;
end;
```

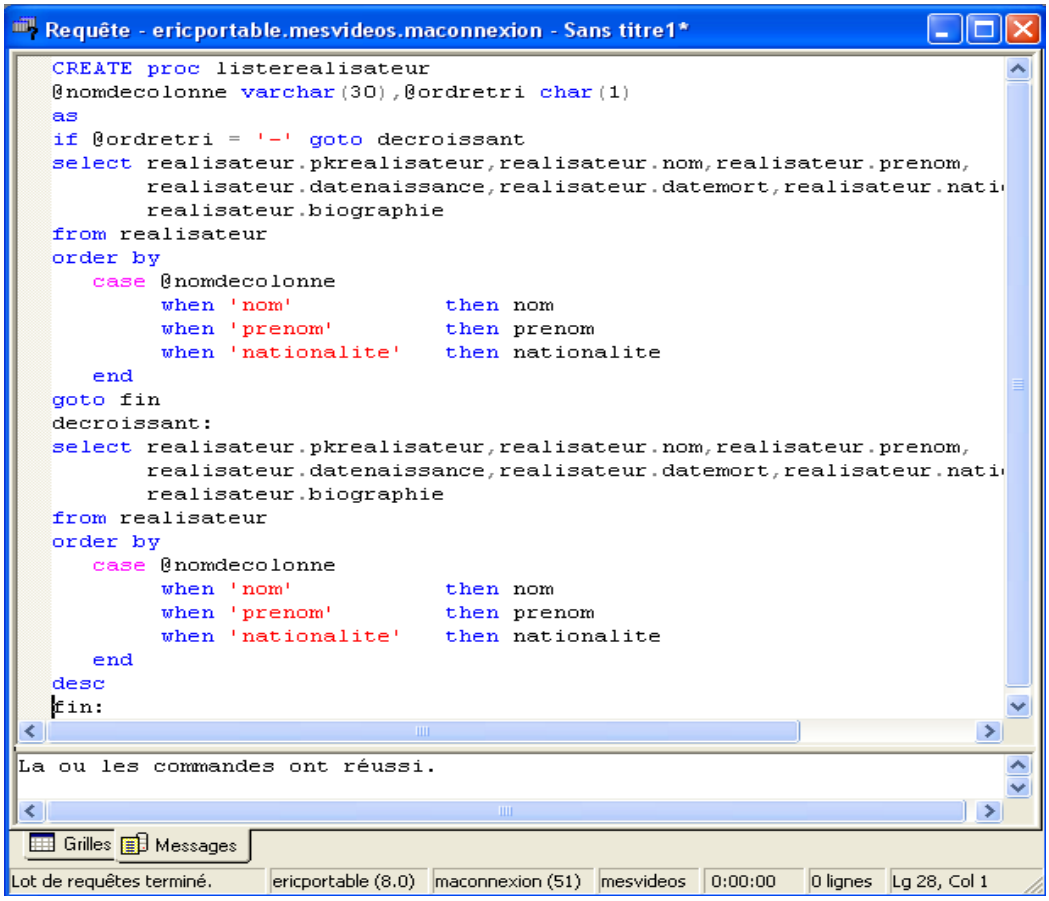
3. LA TABLE DES REALISATEURS.

3.1. Remarques.

Cette table est fort semblable à la précédente. Mais on y rencontre des types de champs différents. (texte, date).
Le principe de construction est le même que pour la table des clients. Je résumerai donc au minimum le développement de la nouvelle fenêtre.
Commençons par la nouvelle forme et son unité :
Name : FREALISATEUR
Caption : Gestion des Réalisateurs.
Unit : UREALISATEUR.
On peut maintenant commencer à travailler mais si vous lancez l'application, vous ne verrez pas la bonne fenêtre. Il faut signaler à DELPHI que c'est la fenêtre des réalisateurs qui est la fenêtre de démarrage.
Projet > Options > Fiche pricipale : FREALISATEUR

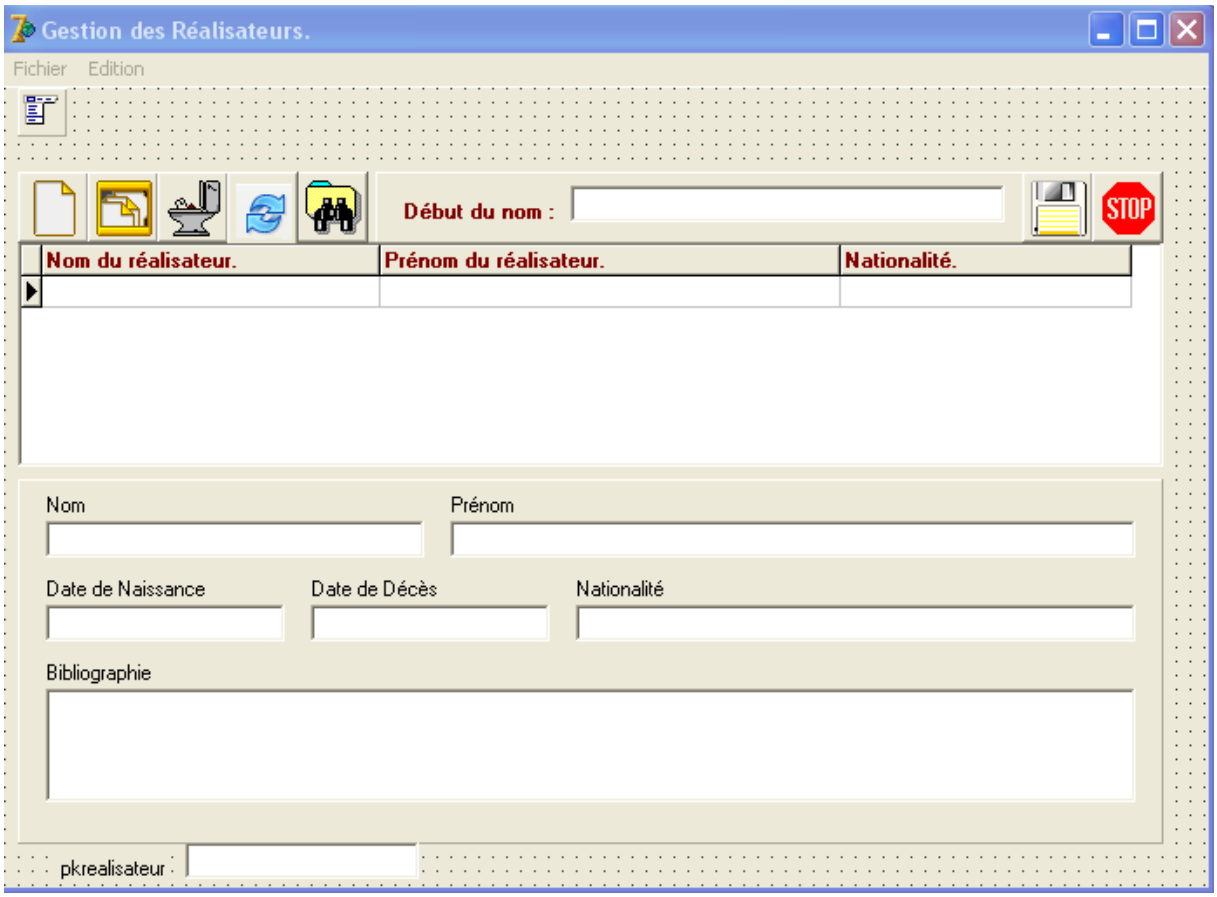


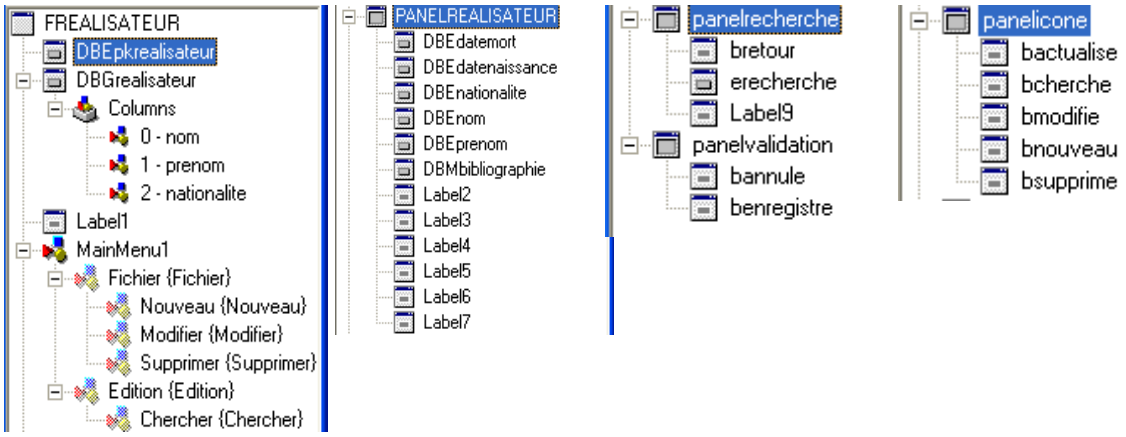
3.2. Procédure stockée de listage.



Pas besoin de faire apparaître d'autres champs dans la grille que les trois qui sont repris dans la clause order by.

Construisons la fenêtre comme nous l'avons fait pour les clients : grille et zones de saisie. N'oublions pas le panelrealisateur.
Nous pouvons ensuite copier et coller les 3 panels : PANELICONE, PANELRECHERCHE et PANELVALIDATION. Ainsi tous les boutons arrivent en même temps . Profitez-en pour copier coller le menu. Vous trouverez ci-dessous l'arborescence des composants et l'image de la fenêtre.

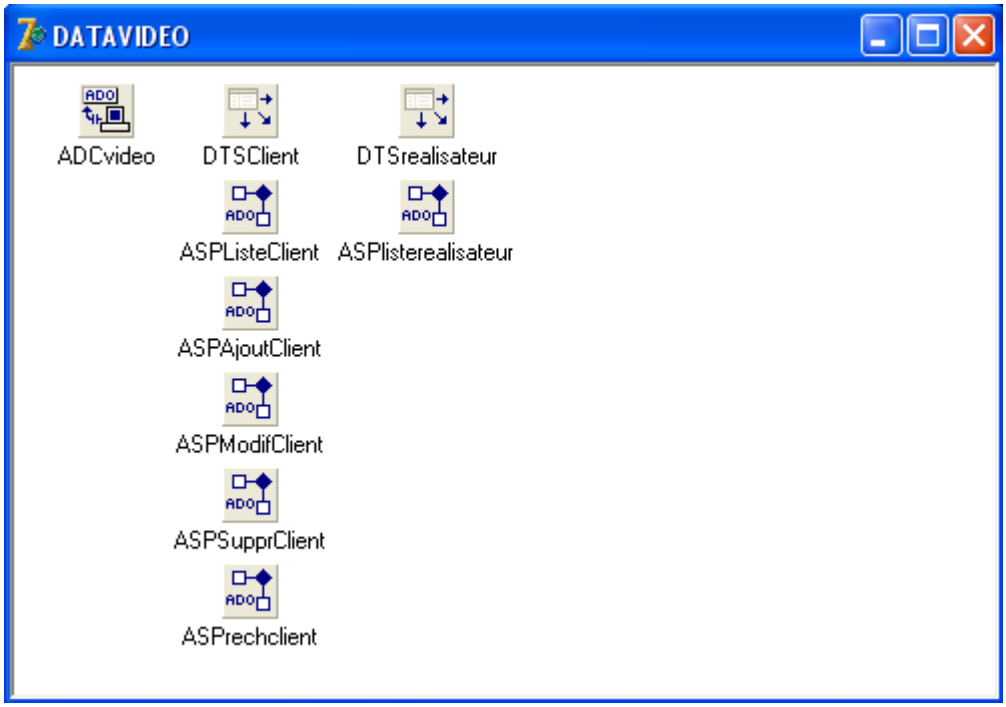




Vous remarquerez que :

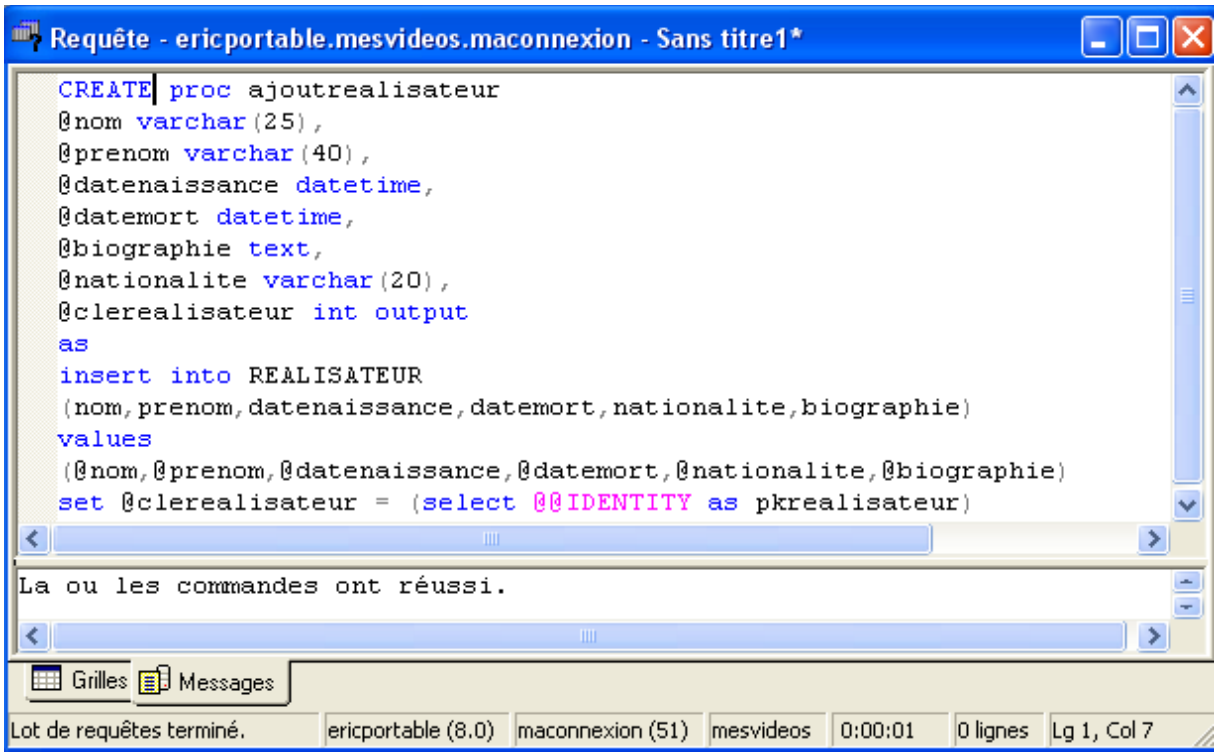
- DBMbibliographie : c'est un DBMEMO et pas un DBEdit (sa taille est différente et son comportement aussi)
- il a fallu redimensionner un peu les composants
- qu'en copiant les panels, vous copiez ses boutons et autres composants (noms compris, dessins,...)
- la mise en page s'en trouve considérablement raccourcie.
- Ne pas oublier les taborder

Voici DATAVIDEO après l'adjonction de la procédure stockée et du datasource.

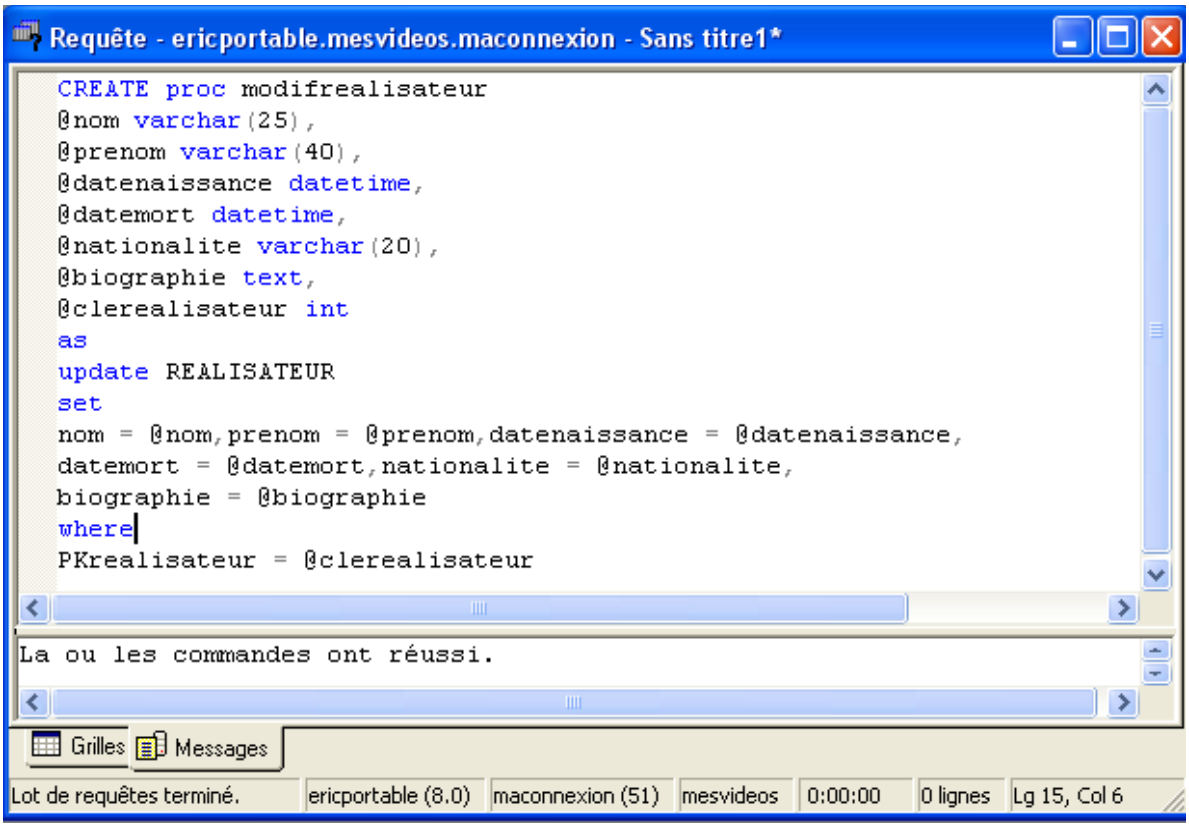


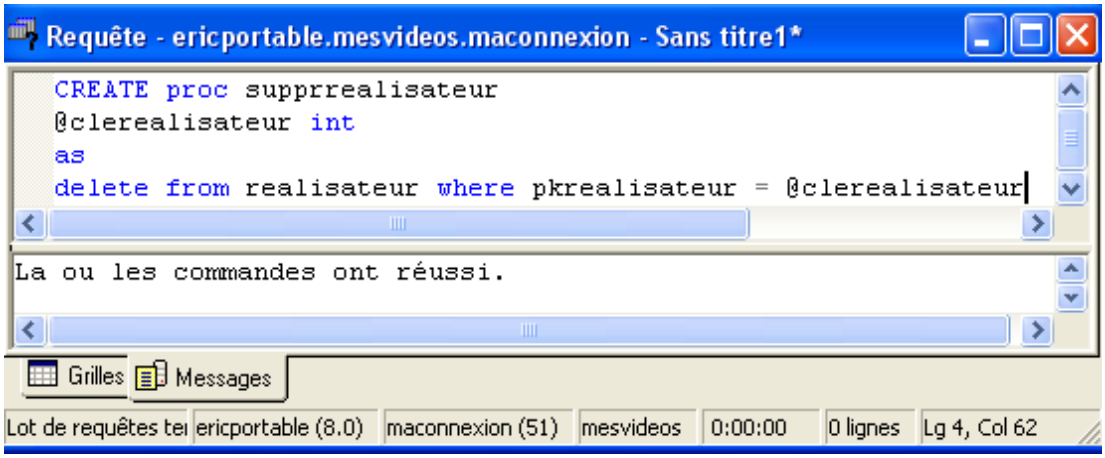
3.3. Les autres procédures stockées.

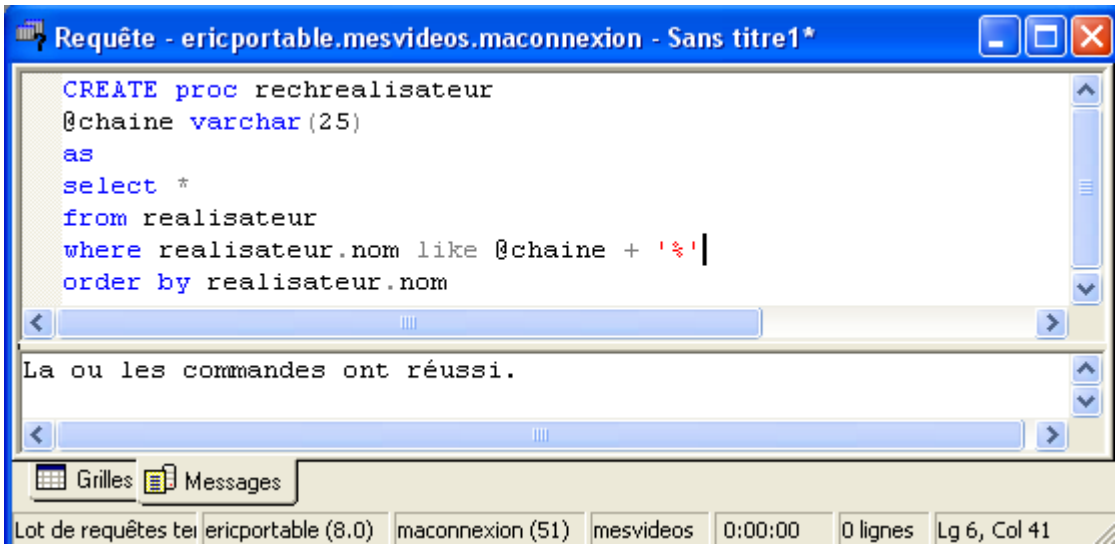
Nous allons créer immédiatement les autres procédures stockées (4) nécessaires à la gestion de la table.



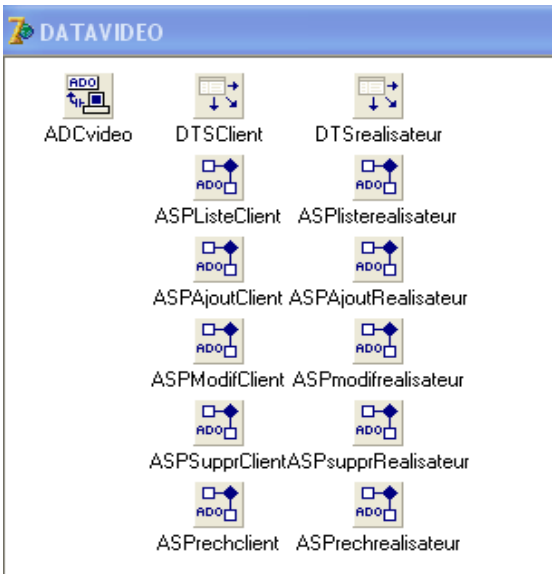
Pas de problème majeur. Attention aux types datetime et text.


Requête - ericportable.mesvideos.maconnexion - Sans titre1*
CREATE proc modifrealisateur
@nom varchar(25),
@prenom varchar(40),
@datenaissance datetime,
@datemort datetime,
@nationalite varchar(20),
@biographie text,
@clerealisateur int
as
update REALISATEUR
set
nom = @nom,prenom = @prenom,datenaissance = @datenaissance,
datemort = @datemort,nationalite = @nationalite,
biographie = @biographie
where
PKrealisateur = @clerealisateur
La ou les commandes ont réussi.
Grilles
Messages
Lot de requêtes terminé.
ericportable (8.0)
maconnexion (51)
mesvideos
0:00:00
0 lignes
Lg 15, Col 6


Requête - ericportable.mesvideos.maconnexion - Sans titre1*
CREATE proc supprrealisateur
@clerealisateur int
as
delete from realisateur where pkrealisateur = @clerealisateur
La ou les commandes ont réussi.
Grilles
Messages
Lot de requêtes tei
ericportable (8.0)
maconnexion (51)
mesvideos
0:00:00
0 lignes
Lg 4, Col 62


Requête - ericportable.mesvideos.maconnexion - Sans titre1*
CREATE proc rechrealisateur
@chaine varchar(25)
as
select *
from realisateur
where realisateur.nom like @chaine + '%'
order by realisateur.nom
La ou les commandes ont réussi.
Grilles
Messages
Lot de requêtes tei
ericportable (8.0)
maconnexion (51)
mesvideos
0:00:00
0 lignes
Lg 6, Col 41

Nous pouvons les placer dans le DATAVIDEO. Vous êtes prudents avec les paramètres de chacune des procédures stockées.



3.4. Ajouter un réalisateur et annuler l'action.

Différence par rapport à la table des clients, nous avons des dates à saisir. Il est essentiel que les dates saisies soient correctes (configuration de la saisie, format, date valide,...). Il existe dans DELPHI un type particulier de composant qui saisit correctement les dates : le DateTimePicker de l'onglet WIN32. Il permet de saisir la date dans un calendrier ou au moyen d'un ascenseur.
Ce composant ne doit être utilisé qu'en mode insertion ou édition et pas en mode affichage. La solution consiste à superposer ce DateTimePicker sur le dbedit et d'inverser la visibilité suivant la situation. Il faut juste que les deux composants soient de la même dimension. Voici leurs propriétés :
name : DTPnaissance et DTPmort
kind : dtkdate (ils vont afficher la date seule)
dateformat :dflong (ils vont afficher la date en long et avec le jour de la semaine)
datemode : dmupdown (ils n'affichent pas un calendrier mais un ascenseur)

Je vous conseille d'écrire d'abord la procédure formcreate pour cacher ces deux composants au démarrage.

```
var
  FREALISATEUR: TFREALISATEUR;
  valnom, valprenom, valnationalite : char;
  clerealisateur : integer;

implementation

uses UDATA;

{$R *.dfm}

procedure TFREALISATEUR.FormCreate(Sender: TObject);
begin
  panelvalidation.Visible := false;
  benregistre.Enabled := false;
  valnom := '-';
  valprenom := '-';
  valnationalite := '-';
  panelrecherche.Visible := false;
  DTPnaissance.Visible := false;
  DTPmort.Visible := false;
end;
```

On en profite pour ajouter les variables val... et la clé réalisateur.

Dans la procédure d'ajout, il faudra penser à rendre visibles les DTP et invisibles les DBE des dates ou encore plus efficace, faisons-le dans nos procédures ecrireoui et ecrirenon que nous pouvons encoder tout de suite. Toutefois, revoyez les taborder pour que ce soient les DTP qui soient prioritaires. Attribuons aussi par défaut la date du jour à DTPnaissance et la plus grande date possible à DTPmort. (pourquoi ?) mais dans la procédure ajout.

```
procedure TFREALISATEUR.bnouveauClick(Sender: TObject);
begin
   dbeprenom.SetFocus;
   datavideo.ASPlisterealisateur.insert;
   dbgrealisateur.Enabled := false;
   fermeoption;
   ecrireoui;
   dtpnaissance.Date := date();
   dtpmort.Date := strtodate('31/12/2099');
end;

procedure tfrealisateur.ouvreoption;
begin
  panelicone.Visible := true;
  panelvalidation.Visible := false;
  nouveau.Enabled := true;
  modifier.Enabled := true;
  supprimer.Enabled := true;
  chercher.Enabled := true;
  dbgrealisateur.Enabled := true;
end;
```

```
procedure tfrealisateur.fermeoption;
begin
  panelicone.Visible := false;
  panelvalidation.Visible := true;
  nouveau.Enabled := false;
  modifier.Enabled := false;
  supprimer.Enabled := false;
  chercher.Enabled := false;
end;

procedure tfrealisateur.ecrireoui;
begin
  dbenom.ReadOnly := false;
  dbeprenom.ReadOnly := false;
  dbenationalite.ReadOnly := false;
  dbmbibliographie.readonly := false;
  dbedatenaissance.visible := false;
  dbedatemort.visible := false;
  dtpnaissance.visible := true;
  dtpmort.visible := true;
end;
```

```
procedure tfrealisateur.ecrirenon;
begin
  dbenom.ReadOnly := true;
  dbeprenom.ReadOnly := true;
  dbenationalite.ReadOnly := true;
  dbmbibliographie.readonly := true;
  dbedatenaissance.visible := true;
  dbedatemort.visible := true;
  dtpnaissance.visible := false;
  dtpmort.visible := false;
end;
```

Attention à la conversion de string en date dans bnouveauclick.
Pour tester tout cela, nous allons écrire tout de suite l'annulation .

```
procedure TFREALISATEUR.bannuleClick(Sender: TObject);
begin
   datavideo.ASPlisterealisateur.Cancel;
   benregistre.Enabled := false;
   ecrirenon;
   ouvreoption;
end;
```

Il faut maintenant écrire la procédure qui permet de faire afficher le bouton d'enregistrement quand les zones non nulles sont replies. Il s'agit ici de nom et datenaissance. La seconde sera toujours non nulle du fait qu'on utilise un composant qui remplit la zone. La procédure est donc la même que celle des clients. (onchange sur dbenom).

```
procedure TFREALISATEUR.DBEnomChange(Sender: TObject);
begin
   benregistre.Enabled := (trim(dbenom.Text) <> '');
end;
```

3.5. La modification et l'enregistrement.

Lors de la modification, il faudra récupérer les dates de la base de données pour les injecter dans les DTP car ce ne sont pas des composants bases de données. Ils ne se mettent pas à jour automatiquement avec l'ensemble de données. Ceci nous donne :

```
procedure TFREALISATEUR.bmodifieClick(Sender: TObject);
begin
   if (trim(dbepkrealisateur.Text) <> '') then
      begin
        clerealisateur := strtoint(dbepkrealisateur.text);
        dbenom.SetFocus;
        datavideo.ASPlisterealisateur.edit;
        fermeoption;
        ecrireoui;
        dtpnaissance.Date := datavideo.ASPlisterealisateurdatenaissance.Value;
        dtpmort.Date := datavideo.ASPlisterealisateurdatemort.value;
      end;
end;
```

Les deux dernières lignes fournissent aux dtp le contenu de la base de données.

Voici la procédure actualise et le celle du bouton d'actualisation.

```
procedure tfrealisateur.actualise;
  begin
    datavideo.ASPlisterealisateur.active := false;
    datavideo.ASPlisterealisateur.active := true
  end;
procedure TFREALISATEUR.bactualiseClick(Sender: TObject);
begin
  actualise;
end;
```

Vous trouverez ensuite la procédure d'enregistrement.

```
procedure TFREALISATEUR.benregistreClick(Sender: TObject);
begin
   if (datavideo.ASPlisterealisateur.State = dsinsert) then
      begin
        with datavideo.ASPajoutrealisateur do
          begin
            parameters.parambyname('@nom').value :=  dbenom.text;
            parameters.ParamByName('@prenom').Value := dbeprenom.Text;
            parameters.ParamByName('@datenaissance').Value := dateof(dtpnaissance.Date);
            if (datetostr(dtpmort.Date) <> '31/12/2099')
               then parameters.ParamByName('@datemort').Value := dateof(dtpmort.date)
               else parameters.ParamByName('@datemort').Value := null;
            parameters.ParamByName('@nationalite').Value := dbenationalite.text;
            parameters.ParamByName('@biographie').Value := dbmbibliographie.Text;
            execproc;
            clerealisateur := parameters.parambyname('@clerealisateur').value;
          end;
      end
   else
      begin
        clerealisateur := strtoint(dbepkrealisateur.Text);
        with datavideo.ASPmodifrealisateur do
          begin
            parameters.ParamByName('@clerealisateur').value := clerealisateur;
            parameters.parambyname('@nom').value :=  dbenom.text;
            parameters.ParamByName('@prenom').Value := dbeprenom.Text;
            parameters.ParamByName('@datenaissance').Value := dateof(dtpnaissance.Date);
            if (datetostr(dtpmort.Date) <> '31/12/2099')
               then parameters.ParamByName('@datemort').Value := dateof(dtpmort.date)
               else parameters.ParamByName('@datemort').Value := null;
            parameters.ParamByName('@nationalite').Value := dbenationalite.text;
            parameters.ParamByName('@biographie').Value := dbmbibliographie.Text;
            execproc;
          end;
      end;
  benregistre.Enabled := false;
  actualise;
  datavideo.ASPlisterealisateur.Locate('pkrealisateur',clerealisateur,[]);
  ecrirenon;
  ouvreoption;
end;
```

Remarques : on attribue par défaut la date 31/12/2099 pour la mort et si cette date est conservée, on encode dans la table la valeur null. Il se fait que null correspond à la date 30/12/1899. Si bien qu'en modification, je dois tester le 30/12/1899 pour tester la nullité et dans ce cas, remplacer par 31/12/2099 dans le DTP. Etrange et pas évident à gérer quand on ne le sait pas.

Pendant qu'on parle des cas spéciaux, pensons à rendre les boutons supprimer, modifier et chercher inutilisables en cas de table vide. Dans UVIDEO :

```
procedure TDATAVIDEO.ASPlisterealisateurAfterOpen(DataSet: TDataSet);
begin
   frealisateur.bmodifie.Enabled := ASPlisterealisateur.recordcount > 0;
   frealisateur.bsupprime.Enabled := ASPlisterealisateur.recordcount > 0;
   frealisateur.bcherche.Enabled := ASPlisterealisateur.recordcount > 0;
   frealisateur.modifier.Enabled := ASPlisterealisateur.recordcount > 0;
   frealisateur.supprimer.Enabled := ASPlisterealisateur.recordcount > 0;
   frealisateur.chercher.Enabled := ASPlisterealisateur.recordcount > 0;
end;
```

Il s'agit de afteropen de ESPlisterealisateur.

3.6. La suppression et la recherche.

```
procedure TFREALISATEUR.bsupprimeClick(Sender: TObject);
begin
   if (messagedlg('Confirmez-vous la suppression de ce réalisateur ? ',
         mtconfirmation,[mbok,mbcancel],0) = mrok) then
      begin
         with datavideo.ASPsupprrealisateur do
           begin
              parameters.ParamByName('@clerealisateur').Value := strtoint(c
              execproc;
           end;
         actualise;
      end;
end;
```

```
procedure TFREALISATEUR.bchercheClick(Sender: TObject);
begin
   fermeoption;
   panelvalidation.Visible := false;
   clerealisateur := strtoint(dbepkrealisateur.Text);
   panelrecherche.Visible := true;
   datavideo.DTSrealisateur.DataSet := datavideo.ASPrechrealisateur;
   erecherche.SetFocus;
end;
```

```
procedure TFREALISATEUR.erechercheChange(Sender: TObject);
begin
  with datavideo.ASPrechrealisateur do
    begin
      active := false;
      parameters.ParamByName('@chaine').Value := erecherche.text;
      execproc;
      active := true;
    end;
end;
```

```
procedure TFREALISATEUR.bretourClick(Sender: TObject);
begin
  if (trim(dbenom.Text) <> '') and (trim(dbepkrealisateur.Text) <> '')
        then clerealisateur := strtoint(dbepkrealisateur.text);
   datavideo.ASPrechrealisateur.Active := false;
   datavideo.DTSrealisateur.DataSet := datavideo.ASPlisterealisateur;
   panelicone.Visible := false;
   actualise;
   datavideo.ASPlisterealisateur.Locate('pkrealisateur',clerealisateur,[]);
   ouvreoption;
   ecrirenon;
   panelrecherche.Visible := false;
end;
```

Je n'insiste sur aucune de ces procédures qui correspondent exactement à celles des clients.

3.7. La gestion de la grille.

Comme pour les clients, nous allons colorier et permettre les tris sur les titres de colonnes.

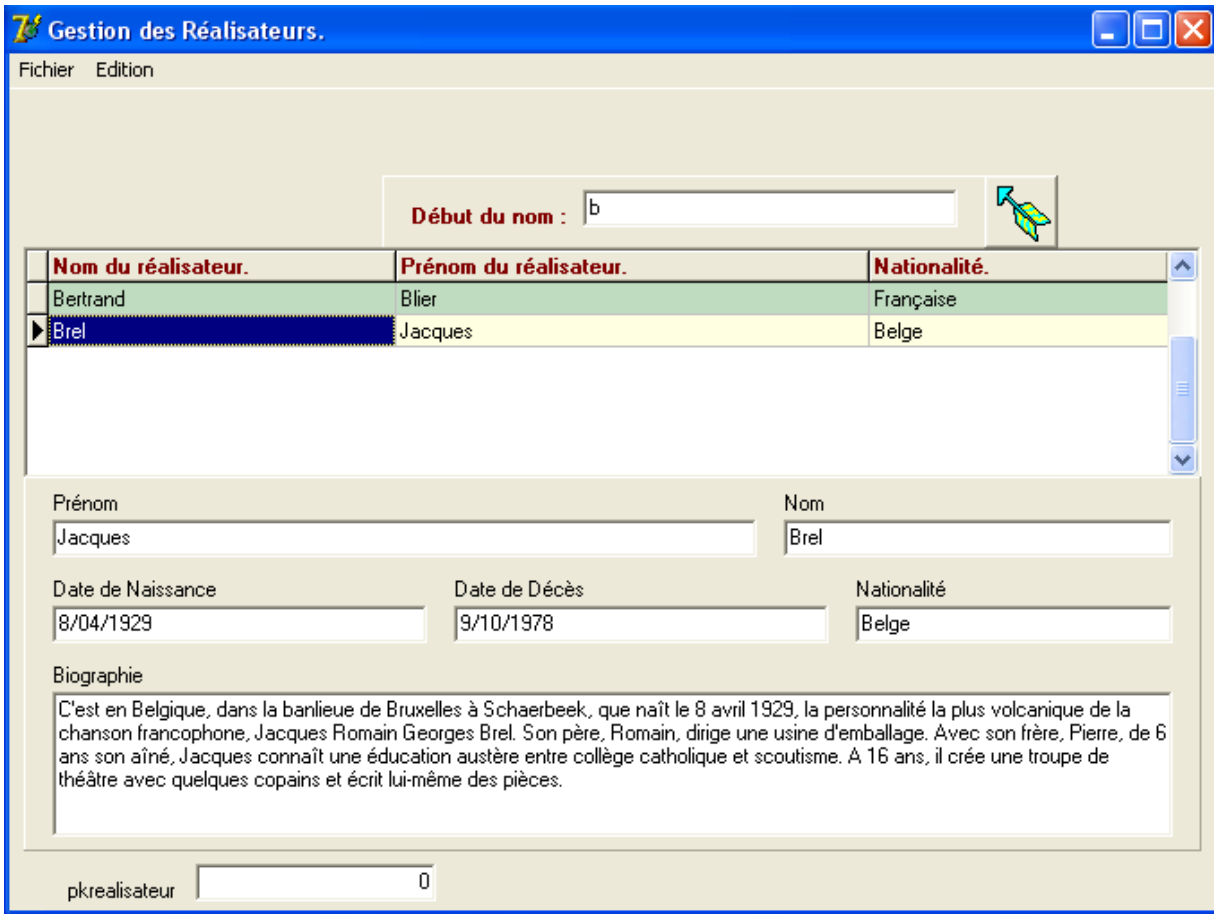
```
procedure TFREALISATEUR.DBGrealisateurTitleClick(Column: TColumn);
var
   cle : integer;
begin
   with DATAVIDEO.ASPlisterealisateur do
      begin
        if (recordcount <> 0) then
          begin
            cle := strtoint(dbepkrealisateur.text);
            active := false;   // retour du curseur
            parameters.parambyname('@nomdecolonne').value :=  column.FieldName;
            if column.FieldName = 'nationalite' then
              begin
                 if valnationalite = '+' then valnationalite := '-'
                                         else valnationalite := '+';
                 parameters.ParamByName('@ordretri').value := valnationalite;
              end;
            if column.FieldName = 'nom' then
              begin
                 if valnom = '+' then valnom := '-'
                                 else valnom := '+';
                 parameters.ParamByName('@ordretri').value := valnom;
              end;
            if column.FieldName = 'prenom' then
              begin
                 if valprenom = '+' then valprenom := '-'
                                    else valprenom := '+';
                 parameters.ParamByName('@ordretri').value := valprenom;
              end;
            active := true;
            locate('pkrealisateur',cle,[]);
          end;
      end;
end;
```

```
procedure TFREALISATEUR.DBGrealisateurDrawColumnCell(Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumn;
  State: TGridDrawState);
var
   grille : tdbgrid;
begin
   grille := (sender as tdbgrid);
   if(grille.DataSource.DataSet.RecNo mod 2) =0
      then grille.Canvas.Brush.Color := clinfobk
      else grille.Canvas.Brush.Color := clmoneygreen;
   if state = [gdselected,gdfocused] then grille.Canvas.Brush.Color := clnavy;
   grille.DefaultDrawColumnCell(rect,datacol,column,state);
end;
```

3.8. Finition.

Ne pas oublier la gestion du menu et les infobulles. Vous remarquerez d'ailleurs que les textes des infobulles ont été conservés lors du copier coller ainsi que la propriété showhint.

Voici pour clore cette partie une image du programme pendant son exécution.



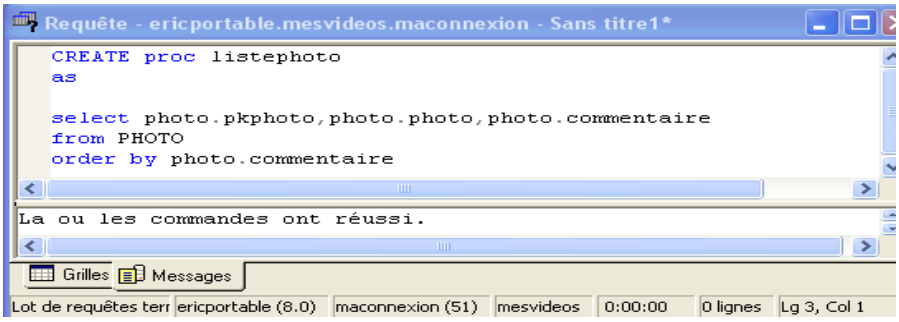
4. LA TABLE DES PHOTOS.

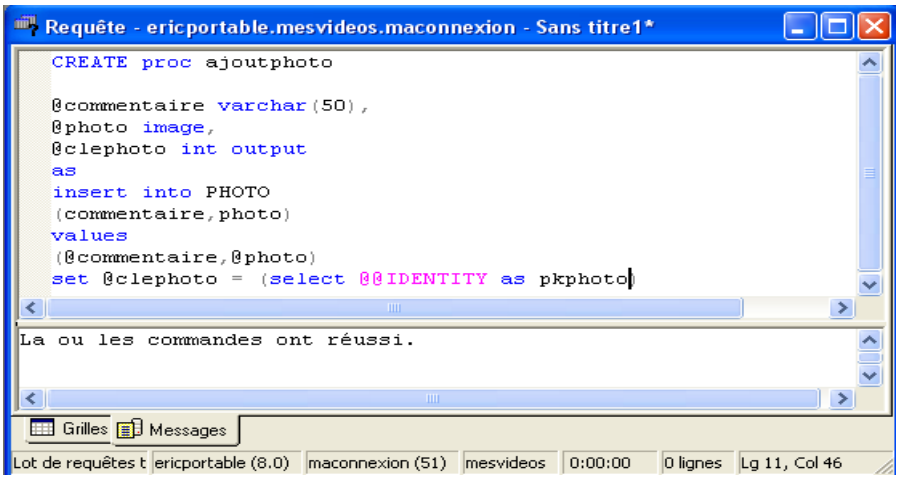
4.1. Préalable.

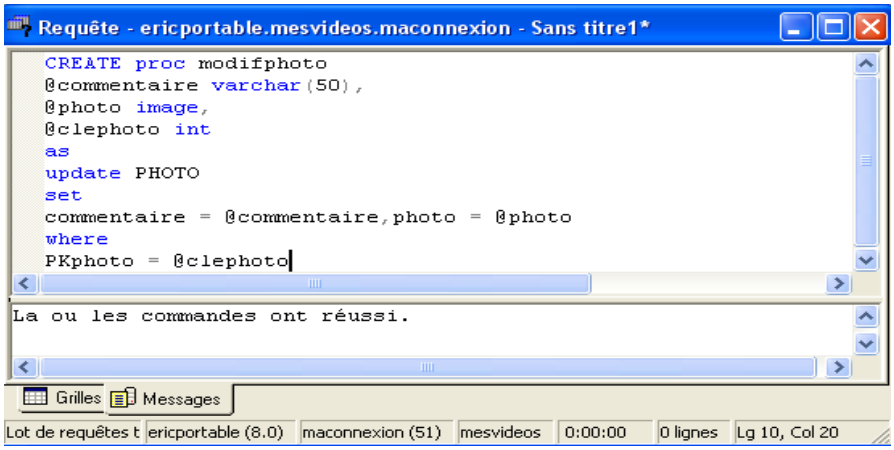
Cette table qui ne comprend presque pas de champs est la seule à contenir des images. Il va falloir gérer la visualisation, l'ajout, la suppression et l'échange de photos. Plus tard, il faudra prévoir la liaison avec les tables ACTEUR ET FILM quand elles seront disponibles. Contentons-nous du minimum (ce qui n'est déjà pas mal).

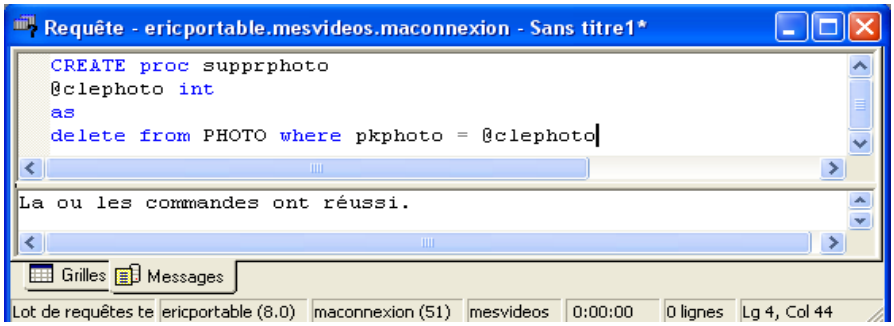
Créons la forme
name : FPHOTO
caption : Gestion des Photos.
Unité : UPHOTO

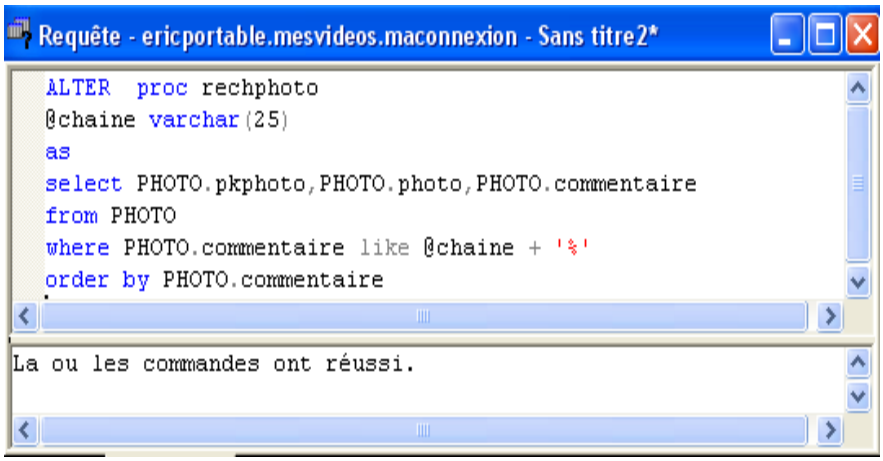
Créons les procédures stockées.



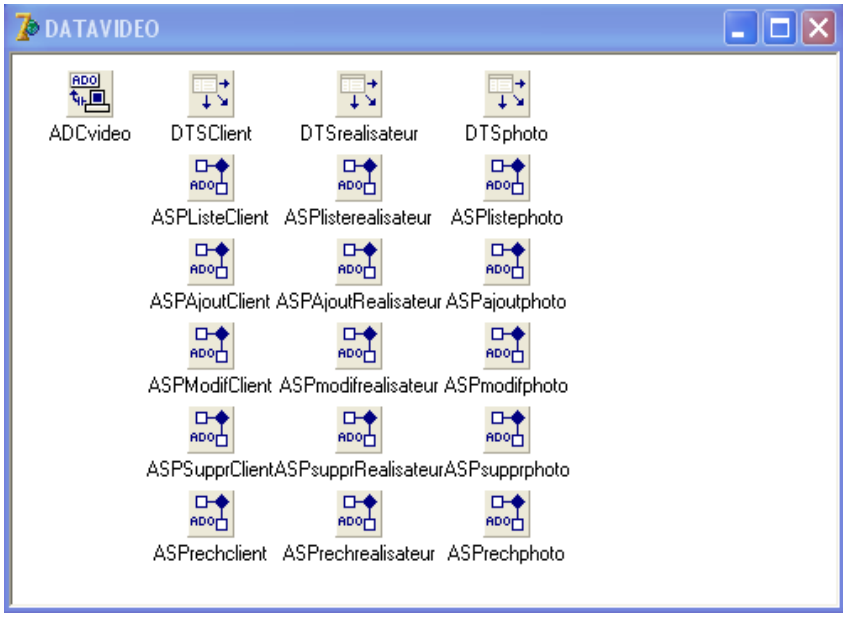








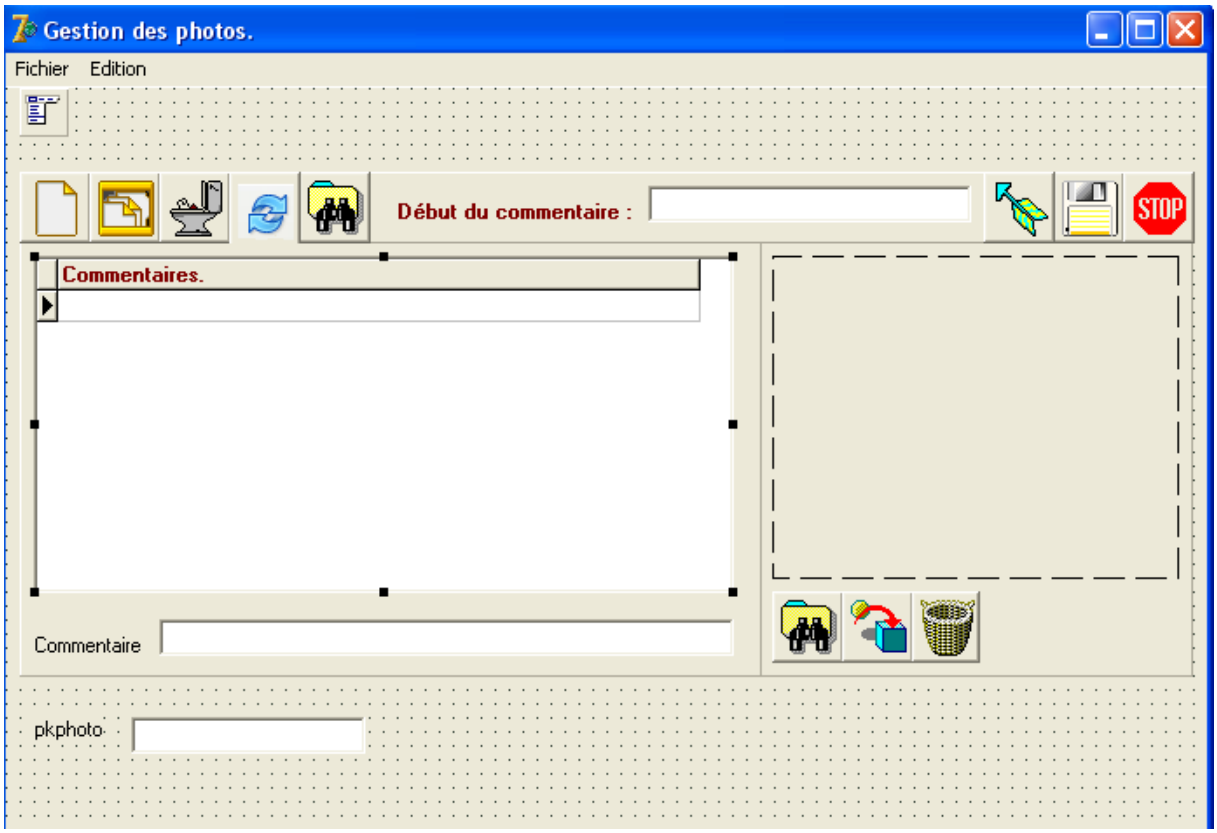
Inscrivons ces procédures stockées dans DATAVIDEO.



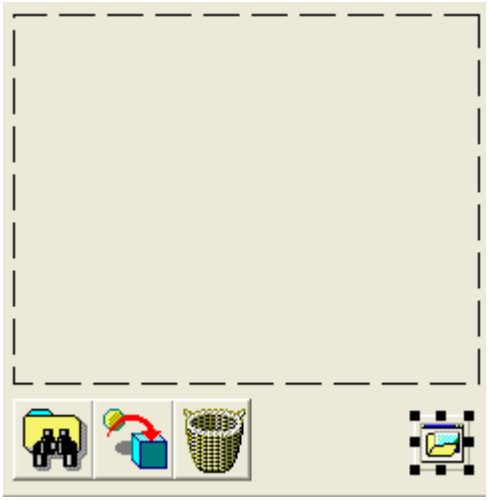
Pas de problème pour les liaisons. Vous êtes devenus des experts. La seule difficulté rencontrée vient du type assigné au paramètre photo : vous choisissez null.

4.2. Création de la forme.

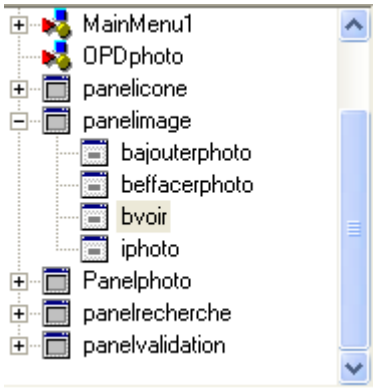
Comme pour la forme FREALISATEUR, nous allons copier les panels PANELICONE,PANELVALIDATION et PANELRECHERCHE. Nous installerons un PANELPHOTO sur lequel nous placerons aussi la grille puisqu'il y a peu de composants. La photo sera placé dans un nouveau panel : PANELIMAGE.



Vous voyez apparaître le nouveau panel avec trois boutons et un nouveau composant destiné à recevoir les photos. Il s'agit du type Image que vous trouverez dans l'onglet Supplément. Certains diront qu'il est plus simple d'utiliser un dbimage. Toutefois, ce type n'accepte pas les images jpg ce qui surcharge les disques.
Appelons ce composant iphoto. Nous ajoutons aussi un composant opendialog que nous appelons opdphoto. On le trouve dans l'ongle Dialogue. Il permet de se promener dans les répertoires pour saisir les documents (ici les photos). Vous l'utilisez tous les jours sans le savoir. Il s'agit d'une fonction de Windows. Cette icône n'apparaîtra pas lors de l'exécution du programme.



Vous trouverez ci-dessous le nom des boutons.



4.3. Stratégie et ajout.

Nous n'afficherons pas systématiquement les images qui correspondent aux lignes de la grille. En effet, afficher à chaque mouvement est facile, mais coûteux en ressources. Le bouton bvoir permettra de visionner la photo stockée dans la table.

Le bouton bnouveau sert à ajouter un nouvel enregistrement. Le bouton bajouterphoto sert à mettre la photo dans l'enregistrement ou à remplacer une photo qui s'y trouvait déjà.

Commençons par créer l'enregistrement (et annulons l'opération). N'oublions pas les procédures personnelles et formcreate. Une variable flagimage de type integer est créée. Nous verrons pourquoi plus tard. Ajoutons aussi la procédure onchange de commentaire (pour rendre utilisable le bouton d'enregistrement ou pas).

```
procedure TFPHOTO.bnouveauClick(Sender: TObject);
begin
   bajouterphoto.Visible := true;
   dbecommentaire.SetFocus;
   datavideo.ASPlistephoto.insert;
   dbgphoto.Enabled := false;
   flagimage := 0;
   fermeoption;
   ecrireoui;
end;

procedure tfphoto.ouvreoption;
begin
  panelicone.Visible := true;
  panelvalidation.Visible := false;
  nouveau.Enabled := true;
  modifier.Enabled := true;
  supprimer.Enabled := true;
  chercher.Enabled := true;
  dbgphoto.Enabled := true;
end;

procedure tfphoto.fermeoption;
begin
  panelicone.Visible := false;
  panelvalidation.Visible := true;
  nouveau.Enabled := false;
  modifier.Enabled := false;
  supprimer.Enabled := false;
  chercher.Enabled := false;
end;
```

```
procedure tfphoto.ecrireoui;
begin
  dbecommentaire.ReadOnly := false;
end;

procedure tfphoto.ecrirenon;
begin
  dbecommentaire.ReadOnly := true;
end;

procedure TFPHOTO.FormCreate(Sender: TObject);
begin
  flagimage := 0;
  panelvalidation.Visible := false;
  benregistre.Enabled := false;
  panelrecherche.Visible := false;
end;
```

```
procedure TFPHOTO.DBEcommentaireChange(Sender: TObject);
begin
   benregistre.Enabled := (trim(dbecommentaire.Text) <> '');
end;

procedure tfphoto.actualise;
begin
  datavideo.ASPlistephoto.active := false;
  datavideo.ASPlistephoto.active := true
end;

procedure TFPHOTO.bannuleClick(Sender: TObject);
begin
   datavideo.ASPlistephoto.Cancel;
   benregistre.Enabled := false;
   bajouterphoto.Visible := false;
   beffacerphoto.Visible := false;
   ecrirenon;
   ouvreoption;
   flagimage := 0;
end;

procedure TFPHOTO.bactualiseClick(Sender: TObject);
begin
   actualise;
end;
```

4.4 La gestion des photos.

Voici la partie délicate de cette unité. Nous allons devoir contourner un problème : la difficulté de gérer les images jpeg et jpg. Ces photos doivent être identifiées au moyen d'une fonction personnelle.

```
function tfphoto.departjpegblob(image : tblobfield) : integer;
var
  bs : tadoblobstream;
  buffer : word;
  hx : string;
begin
  result := -1;
  bs := tadoblobstream.create(image, bmread);
  try
    while(result = -1) and(bs.position + 1 < bs.size) do
      begin
        bs.readbuffer(buffer,1);
        hx := inttohex(buffer,2);
        if hx = 'FF' then
           begin
             bs.readbuffer(buffer,1);
             hx := inttohex(buffer,2);
             if hx = 'D8' then result := bs.position - 2
                          else if hx = 'FF' then bs.position := bs.Position - 1;
           end;
      end;
  finally
    bs.free;
  end;
end;
```

et dans la partie interface :

function departjpegblob(image : tblobfield) : integer;

Il faudra ajouter dans les uses les librairies db,adodb et jpeg. Vous le constaterez.

Explications relatives à cette fonction :

Dans un jpg ou jpeg, le début de l'image dans le fichier container se repère par deux bytes successifs qui valent FF et D8. Il faudra donc ne charger dans la base que la partie qui commence après ces deux bytes. Cette fonction détermine si ces deux bytes successifs sont présents et renvoie la position (éventuelle) au programme appelant. L'usage du try...finally...end est rendu nécessaire pour éviter des messages d'erreur.

Je vous conseille de bien garder cette fonction (que je n'ai d'ailleurs pas écrite moi-même) qui vous servira telle quelle dans tous les cas où vous utiliserez des images jpg.

```
procedure tfphoto.voirphoto;
var
   phototmp : tjpegimage;
   bs : tadoblobstream;
begin
   bs := tadoblobstream.Create(datavideo.ASPlistephotophoto,bmread);
   if bs.Size <> 0 then
      begin
        try
           bs.Seek(departjpegblob(datavideo.ASPlistephotophoto),sofrombeginning);
           phototmp := tjpegimage.Create;
           try
              phototmp.LoadFromStream(bs);
              iphoto.Visible := true;
              iphoto.Picture.Graphic := phototmp;
           finally
              phototmp.Free;
           end;
        finally
           bs.Free;
        end;
     end;
end;

procedure TFPHOTO.bvoirClick(Sender: TObject);
begin
  voirphoto;
end;
```

La procédure personnelle (ne pas oublier d'aller écrire la ligne procedure voirphoto; dans l'interface) sert à visualiser les photos. Elle est aussi construite autour d'un try...finally...end (même un double) car si la place en mémoire n'est pas suffisante, la photo n'est pas chargée et le message d'erreur est intercepté et transformé en langage compréhensible pour l'utilisateur. Cette procédure charge en mémoire l'image depuis la base de données puis la place dans le iphoto et libère l'espace temporaire de mémoire alloué au transfert.

La procédure bvoirclick correspond au bouton de visualisation. Elle appelle la procédure voirphoto. J'ai procédé de la sorte car je pense avoir besoin plus loin de la procédure personnelle appelée par une autre procédure.

```
procedure TFPHOTO.bajouterphotoClick(Sender: TObject);
begin
  if opdphoto.execute then
     begin
        iphoto.visible := true;
        iphoto.picture.loadfromfile(opdphoto.filename);
        flagimage := 1;
     end;
end;
```

Cette procédure lance le dialoque de recherche et stocke la photo que nous venons de saisir dans une zone temporaire de la mémoire. C'est cette zone que nous passerons en paramètre au moment d'enregistrer la ligne.

```
procedure TFPHOTO.benregistreClick(Sender: TObject);
var
  bs1 : tmemorystream;
  flag1 : boolean;
begin
  try
     bs1 := tmemorystream.Create;
     flag1 := true;
     try
        iphoto.picture.graphic.savetostream(bs1);
     except
        flag1 := false;
     end;
     if (datavideo.ASPlistephoto.State = dsinsert) then
        begin
           with datavideo.ASPajoutphoto do
              begin
                   parameters.parambyname('@commentaire').value :=  dbecommentaire.text;
                   if (flag1) and (flagimage = 1)
                     then parameters.parambyname('@photo').loadfromstream(bs1,ftblob)
                     else parameters.ParamByName('@photo').value := '';
                   execproc;
                   clephoto := parameters.parambyname('@clephoto').value;
              end;
        end
     else
        begin
           clephoto := strtoint(dbepkphoto.Text);
           with datavideo.ASPmodifphoto do
              begin
                   parameters.ParamByName('@clephoto').value := clephoto;
                   parameters.parambyname('@commentaire').value :=  dbecommentaire.text;
                   if (flag1) and (flagimage = 1)
                     then parameters.parambyname('@photo').loadfromstream(bs1,ftblob)
                     else parameters.ParamByName('@photo').value := '';
                   execproc;
              end;
        end;
  finally
     bs1.Free;
  end;
  benregistre.Enabled := false;
  actualise;
  bajouterphoto.Visible := false;
  datavideo.ASPlistephoto.Locate('pkphoto',clephoto,[]);
  dbgphoto.Enabled := true;
  bajouterphoto.Visible := false;
  beffacerphoto.Visible := false;
  flagimage := 0;
  ecrirenon;
  ouvreoption;
end;
```

Vous voyez ici que la zone temporaire est passée comme paramètre lors de l'enregistrement. Ici aussi, nous sommes prudents et utilisons la structure try...finally...end. Le drapeau flag (variable locale) est placé à 1 si la zone de transfert est créée. Si elle n'est pas créée, le paramètre reçoit la valeur null.
Le drapeau flagimage (variable publique) est activé par la procédure d'ajout d'image. Il est donc impossible d'insérer une image sans être dans ce mode.

Si vous créer deux enregistrements avec une image, vous constaterez que le fait de se déplacer dans la grille ne modifie pas l'image affichée dans le iphoto. De plus, je pense qu'il ne sert à rien d'afficher la photo à chaque déplacement dans la table. Il faudrait également n'afficher le bouton voir photo que si une photo ést présente pour cet enregistrement. La procédure suivante va répondre à cette demande. Dans datavideo, vous choisissez l'événement afterscroll (après défilement) de listephoto.

```
procedure TDATAVIDEO.ASPlistephotoAfterScroll(DataSet: TDataSet);
begin
     fphoto.bvoir.Enabled := ASPlistephotophoto.Value <> '';
     fphoto.iphoto.visible := false;
end;
```

On voit que le bouton est actif si une photo est présente dans l'enregistrement courant et dans ce cas, iphoto est visible et permet donc l'affichage de la photo sur demande.

Procédure de suppression d'une image : attention : uniquement possible en cas de modification ou ajout d'enregistrement.

```
procedure TFPHOTO.beffacerphotoClick(Sender: TObject);
begin
   flagimage := 0;
   iphoto.Visible := false;
   showmessage('La photo ne sera définitivement supprimée qu''après enregistrement.');
end;
```

Attention : présence d'un avertissement qui dit bien ce qui se passe car l'effacement n'est pas immédiat.(ne pas oublier de mettre deux ' dans la chaîne.

Pensons aussi à la procédure de coloriage de la grille.

```
procedure TFPHOTO.DBGphotoDrawColumnCell(Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumn;
  State: TGridDrawState);
var
   grille : tdbgrid;
begin
   grille := (sender as tdbgrid);
   if(grille.DataSource.DataSet.RecNo mod 2) =0
      then grille.Canvas.Brush.Color := clinfobk
      else grille.Canvas.Brush.Color := clmoneygreen;
   if state = [gdselected,gdfocused] then grille.Canvas.Brush.Color := clnavy;
   grille.DefaultDrawColumnCell(rect,datacol,column,state);
end;
```

4.5. Modification, suppression et recherche.

```
procedure TFPHOTO.bsupprimeClick(Sender: TObject);
begin
   if (messagedlg('Confirmez-vous la suppression de cette photo ? ',
         mtconfirmation,[mbok,mbcancel],0) = mrok) then
      begin
         with datavideo.ASPsupprphoto do
           begin
              parameters.ParamByName('@clephoto').Value := strtoint(dbepkphoto.
              execproc;
           end;
         actualise;
      end;
end;
```

```
procedure TFPHOTO.bmodifieClick(Sender: TObject);
begin
   if (trim(dbepkphoto.Text) <> '') then
      begin
        clephoto := strtoint(dbepkphoto.text);
        dbecommentaire.SetFocus;
        datavideo.ASPlistephoto.edit;
        bajouterphoto.Visible := true;
        fermeoption;
        ecrireoui;
      end;
end;
```

```
procedure TFPHOTO.bchercheClick(Sender: TObject);
begin
   fermeoption;
   panelvalidation.Visible := false;
   clephoto := strtoint(dbepkphoto.Text);
   panelrecherche.Visible := true;
   datavideo.DTSphoto.DataSet := datavideo.ASPrechphoto;
   erecherche.SetFocus;
end;

procedure TFPHOTO.erechercheChange(Sender: TObject);
begin
  with datavideo.ASPrechphoto do
    begin
      active := false;
      parameters.ParamByName('@chaine').Value := erecherche.text;
      execproc;
      active := true;
    end;
end;

procedure TFPHOTO.bretourClick(Sender: TObject);
begin
  if (trim(dbecommentaire.Text) <> '') and (trim(dbepkphoto.Text) <> '')
        then clephoto := strtoint(dbepkphoto.text);
   datavideo.ASPrechphoto.Active := false;
   datavideo.DTSphoto.DataSet := datavideo.ASPlistephoto;
   panelicone.Visible := false;
   actualise;
   datavideo.ASPlistephoto.Locate('pkphoto',clephoto,[]);
   ouvreoption;
   ecrirenon;
   panelrecherche.Visible := false;
end;
```

Toutefois, lors de cette recherche, vous constaterez que les photos ne correspondent plus aux enregistrements. Il va falloir modifier la procédure voirphoto en conséquence. Il faudra aussi dans datavideo affecter l'événement afterscroll de ASPrechphoto pour modifier la visualisation ou non.

```
procedure tfphoto.voirphoto;
var
   phototmp : tjpegimage;
   bs : tadoblobstream;
begin
   if datavideo.DTSphoto.DataSet = datavideo.ASPlistephoto
      then bs := tadoblobstream.Create(datavideo.ASPlistephotophoto,bmread)
      else bs := tadoblobstream.Create(datavideo.ASPrechphotophoto,bmread);
   if bs.Size <> 0 then
      begin
        try
           if datavideo.DTSphoto.DataSet = datavideo.ASPlistephoto
               then bs.Seek(departjpegblob(datavideo.ASPlistephotophoto),sofrombeginning)
               else bs.Seek(departjpegblob(datavideo.ASPrechphotophoto),sofrombeginning);
           phototmp := tjpegimage.Create;
           try
              phototmp.LoadFromStream(bs);
              iphoto.Visible := true;
              iphoto.Picture.Graphic := phototmp;
           finally
              phototmp.Free;
           end;
        finally
           bs.Free;
        end;
     end;
end;
```

```
procedure TDATAVIDEO.ASPrechphotoAfterScroll(DataSet: TDataSet);
begin
     fphoto.bvoir.Enabled := ASPrechphotophoto.Value <> '';
     fphoto.iphoto.visible := false
end;
```

Ne pas oublier non plus de vérifier la présence d'enregistrements dans la table.

```
procedure TDATAVIDEO.ASPlistephotoAfterOpen(DataSet: TDataSet);
begin
   fphoto.bmodifie.Enabled := ASPlistephoto.recordcount > 0;
   fphoto.bsupprime.Enabled := ASPlistephoto.recordcount > 0;
   fphoto.bcherche.Enabled := ASPlistephoto.recordcount > 0;
   fphoto.modifier.Enabled := ASPlistephoto.recordcount > 0;
   fphoto.supprimer.Enabled := ASPlistephoto.recordcount > 0;
   fphoto.chercher.Enabled := ASPlistephoto.recordcount > 0;
end;
```

Enfin, il faut documenter les boutons et revoir les événements du menu.

Il faudra enfin tester ce programme en mode programme (en lançant le programme exécutable) et non en mode exécution temporaire (depuis la flèche verte de DELPHI).

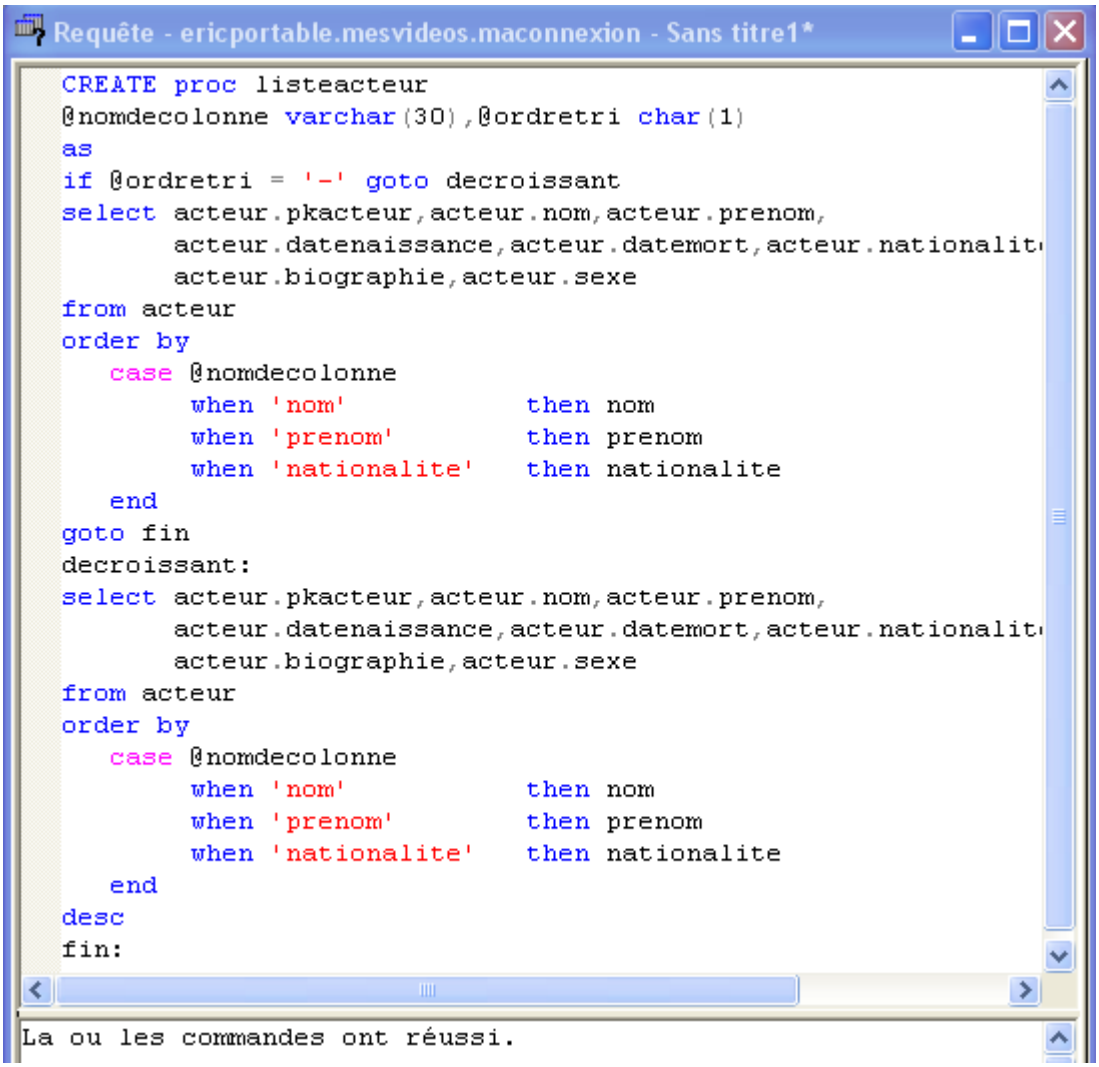
5. LA TABLE DES ACTEURS.

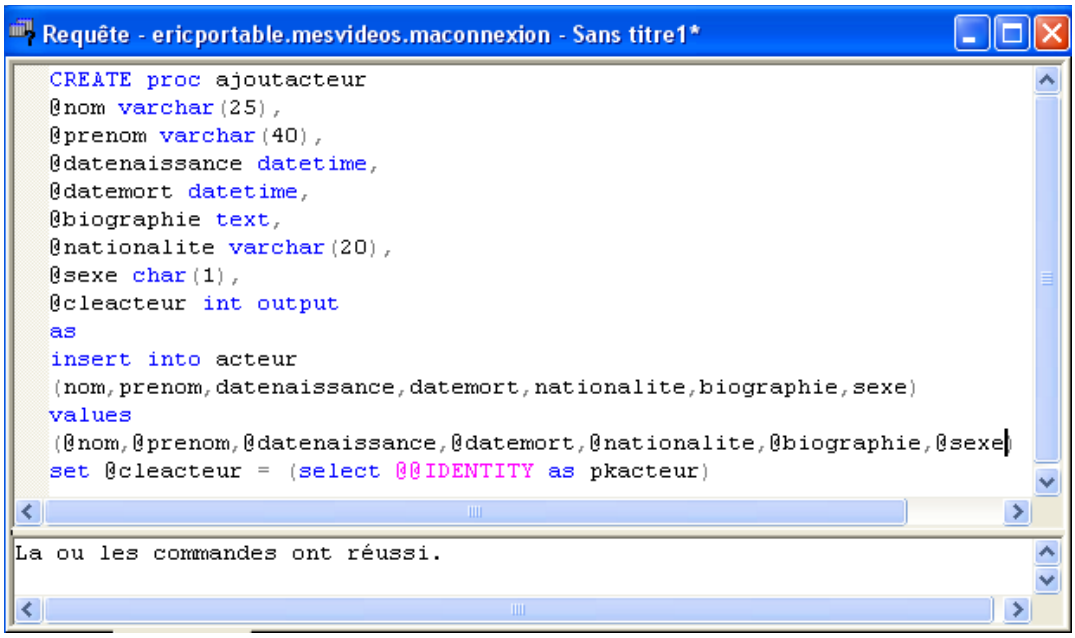
5.1. Préalable.

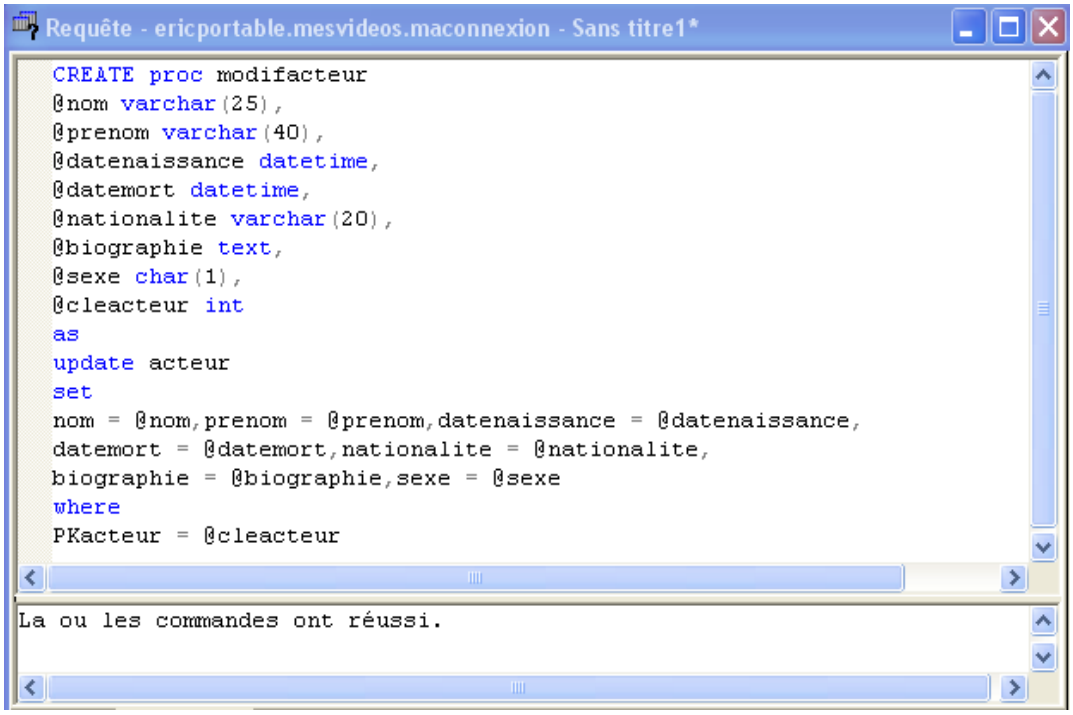
Voici une table qui va être liée à une autre : celle des photos. On se trouve devant un choix : les photos liées à un acteur seront-elles ajoutées depuis la table des acteurs ou ajouterons-nous les acteurs aux photos depuis la fenêtre des photos ? Je propose de faire les deux. Ceci nous obligera à revoir la fenêtre des photos, ce que nous ferons quand nous permettrons la liaison avec la table des films.

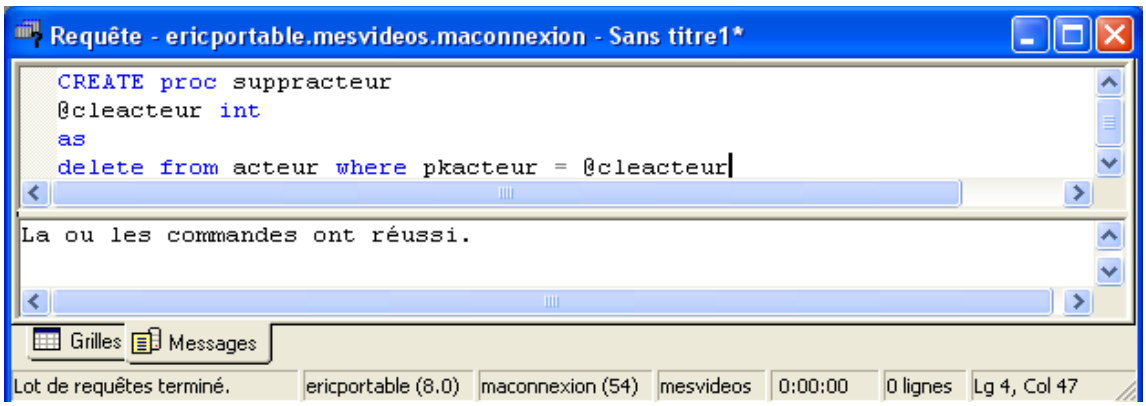
5.2. Structure de la fenêtre.

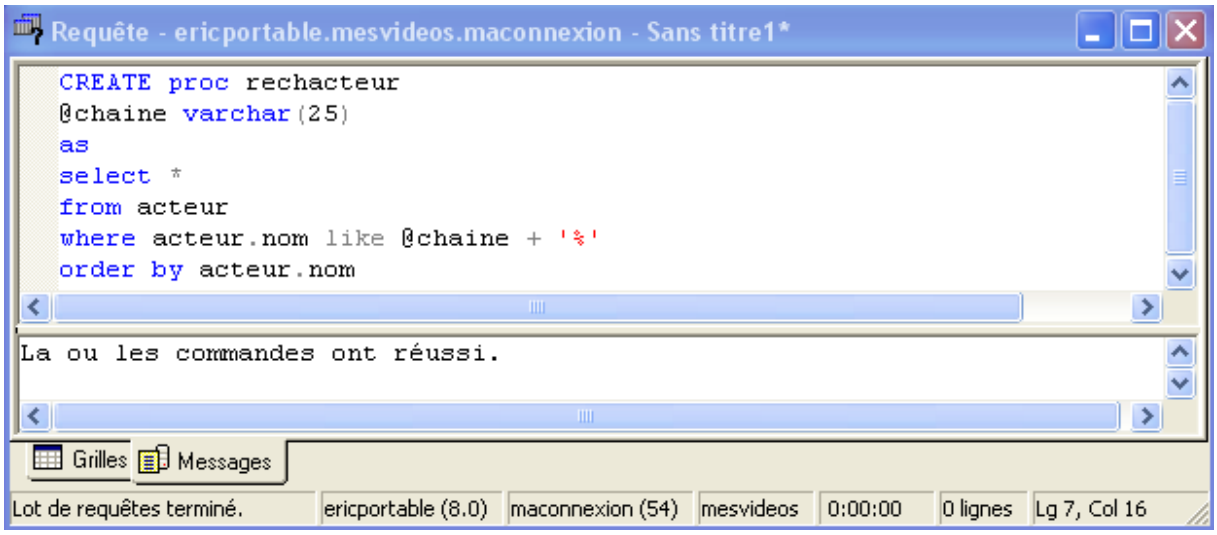
Les photos ne seront visibles que si l'utilisateur le souhaite en choisissant une icône photo. Celle-ci nous enverra sur la fenêtre des photos avec une procédure stockée restreignant les photos à celles de l'acteur. En mode création d'une liaison avec une photo, nous serons aussi envoyés dans la fenêtre des photos mais avec plus de possibilités. Construisons d'abord la fenêtre sans tenir compte des photos. Elle ressemblera à celle des réalisateurs en ajoutant le champ sexe. Nous allons d'abord créer les procédures stockées traditionnelles.

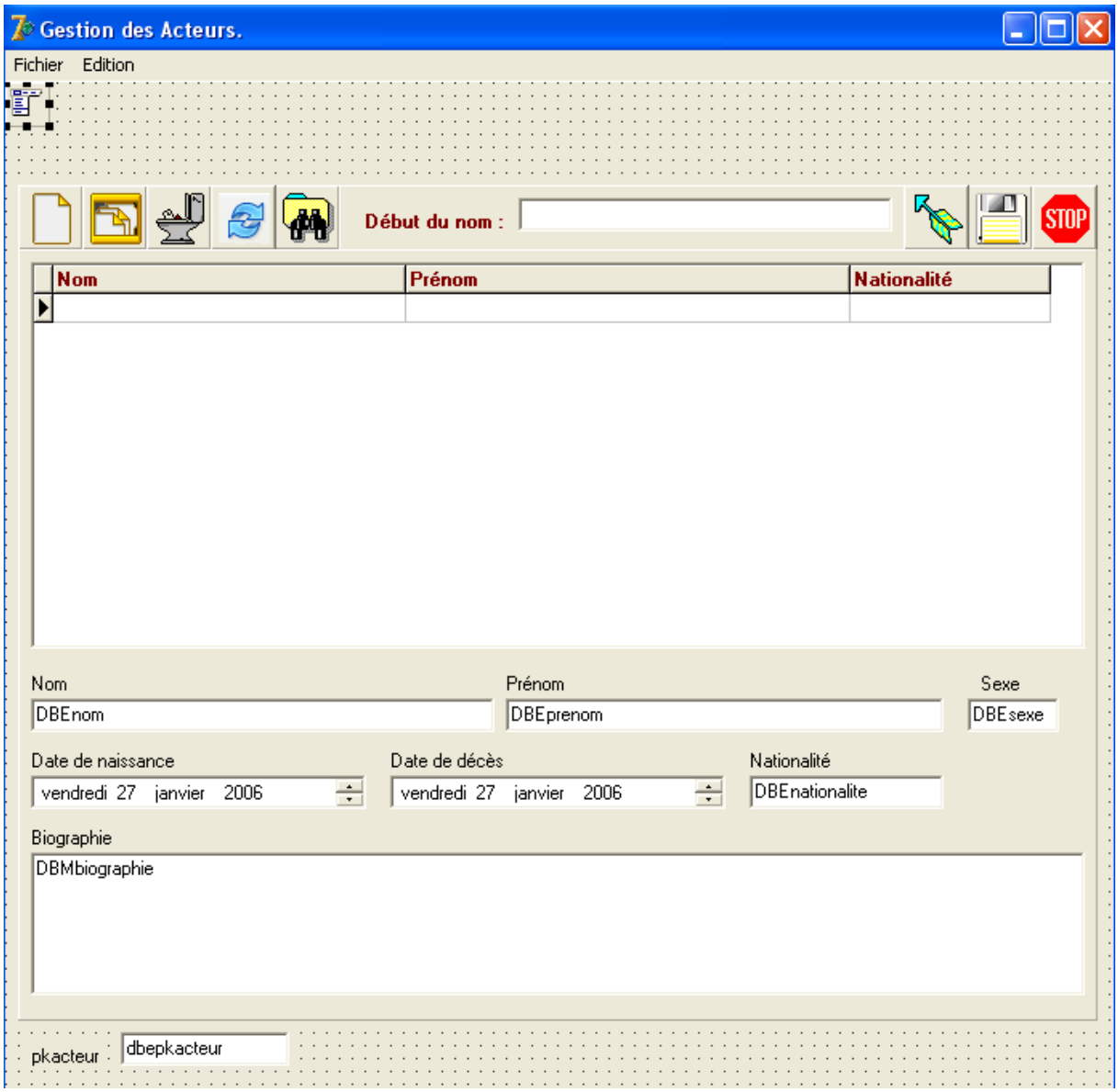



Requête - ericportable.mesvideos.maconnexion - Sans titre1*
CREATE proc ajoutacteur
@nom varchar(25),
@prenom varchar(40),
@datenaissance datetime,
@datemort datetime,
@biographie text,
@nationalite varchar(20),
@sexe char(1),
@cleacteur int output
as
insert into acteur
(nom,prenom,datenaissance,datemort,nationalite,biographie,sexe)
values
(@nom,@prenom,@datenaissance,@datemort,@nationalite,@biographie,@sexe)
set @cleacteur = (select @@IDENTITY as pkacteur)
La ou les commandes ont réussi.


Requête - ericportable.mesvideos.maconnexion - Sans titre1*
CREATE proc modifacteur
@nom varchar(25),
@prenom varchar(40),
@datenaissance datetime,
@datemort datetime,
@nationalite varchar(20),
@biographie text,
@sexe char(1),
@cleacteur int
as
update acteur
set
nom = @nom,prenom = @prenom,datenaissance = @datenaissance,
datemort = @datemort,nationalite = @nationalite,
biographie = @biographie,sexe = @sexe
where
PKacteur = @cleacteur
La ou les commandes ont réussi.

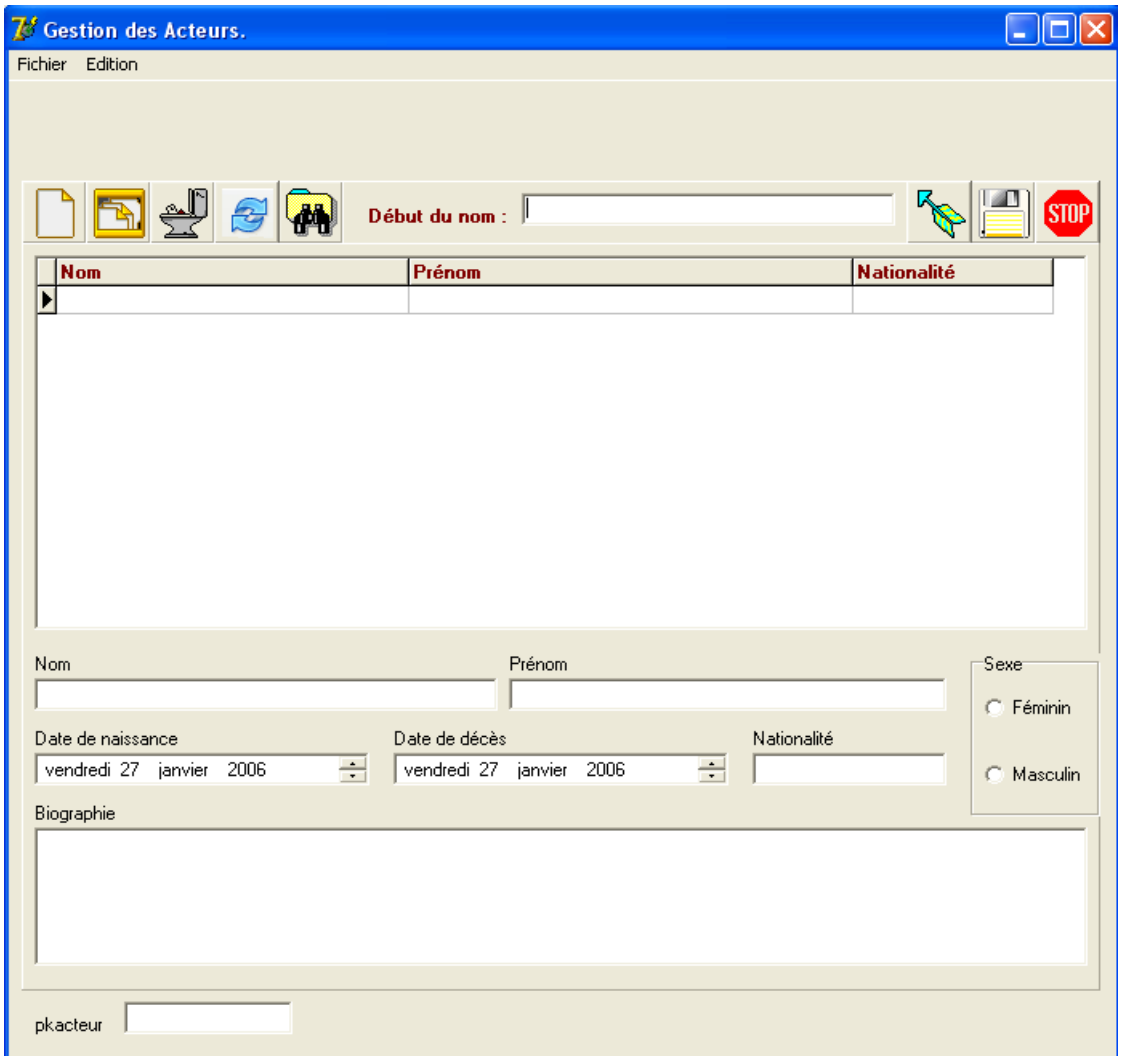

Requête - ericportable.mesvideos.maconnexion - Sans titre1*
CREATE proc suppracteur
@cleacteur int
as
delete from acteur where pkacteur = @cleacteur
La ou les commandes ont réussi.
Grilles
Messages
Lot de requêtes terminé.
ericportable (8.0)
maconnexion (54)
mesvideos
0:00:00
0 lignes
Lg 4, Col 47


Requête - ericportable.mesvideos.maconnexion - Sans titre1*
CREATE proc rechacteur
@chaine varchar(25)
as
select *
from acteur
where acteur.nom like @chaine + '%'
order by acteur.nom
La ou les commandes ont réussi.
Grilles
Messages
Lot de requêtes terminé.
ericportable (8.0)
maconnexion (54)
mesvideos
0:00:00
0 lignes
Lg 7, Col 16

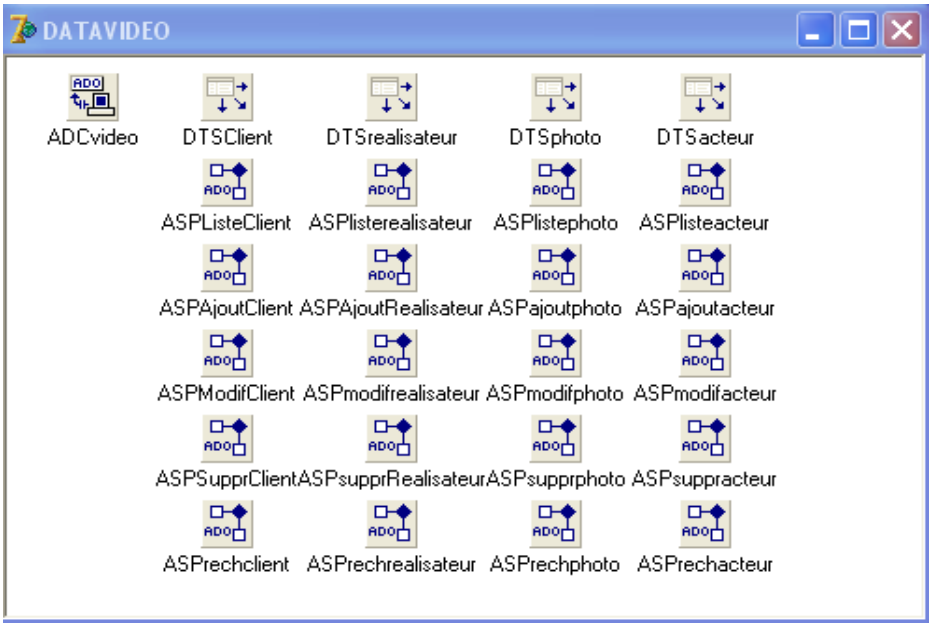


Unité : UACTEUR
fenêtre : FACTEUR
Ne pas oublier de placer un DTPnaissance et un DTPmort.

Je propose également de remplacer à la saisie le DBEsexe par un nouveau composant : un DBRadiogroup sur lequel nous aurons deux items : Féminin et Masculin. Ce qui transforme la fenêtre en :



Il faudra évidemment que le bon radio bouton soit activé au moment de la modification et initialiser un des deux en mode création.. Allons plus loin : supprimons le DBEsexe et son label pour garder tout le temps les boutons radios.



L'usage de DBRsexe exige ces préalables :

- il faut remplir la propriété items par : Féminin et à la ligne suivante Masculin.
- il faut remplir la propriété values par F et à la ligne suivante M.
- ne pas oublier de faire remplir le datasource et le datafield correctement

Voici les différentes procédures utilisées. Elles sont semblables à celles des réalisateurs à une ou l'autre ligne près.

```
    procedure actualise;
    procedure ouvreoption;
    procedure fermeoption;
    procedure ecrireoui;
    procedure ecrirenon;
    procedure FormCreate(Sender: TObject);
    procedure bnouveauClick(Sender: TObject);
    procedure bannuleClick(Sender: TObject);
    procedure benregistreClick(Sender: TObject);
    procedure DBEnomChange(Sender: TObject);
    procedure bmodifieClick(Sender: TObject);
    procedure bsupprimeClick(Sender: TObject);
    procedure bactualiseClick(Sender: TObject);
    procedure bchercheClick(Sender: TObject);
    procedure erechercheChange(Sender: TObject);
    procedure bretourClick(Sender: TObject);
    procedure DBGacteurTitleClick(Column: TColumn);
    procedure DBGacteurDrawColumnCell(Sender: TObject; const R
      DataCol: Integer; Column: TColumn; State: TGridDrawState
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  FACTEUR: TFACTEUR;
  valnom, valprenom,valnationalite : char;
  cleacteur : integer;
```

```
procedure tfacteur.ouvreoption;
begin
  panelicone.Visible := true;
  panelvalidation.Visible := false;
  nouveau.Enabled := true;
  modifier.Enabled := true;
  supprimer.Enabled := true;
  chercher.Enabled := true;
  dbgacteur.Enabled := true;
end;

procedure tfacteur.fermeoption;
begin
  panelicone.Visible := false;
  panelvalidation.Visible := true;
  nouveau.Enabled := false;
  modifier.Enabled := false;
  supprimer.Enabled := false;
  chercher.Enabled := false;
end;
```

```
procedure tfacteur.ecrireoui;
begin
  dbenom.ReadOnly := false;
  dbeprenom.ReadOnly := false;
  dbenationalite.ReadOnly := false;
  dbmbiographie.readonly := false;
  dbrsexe.ReadOnly := false;
  dbenaissance.visible := false;
  dbemort.visible := false;
  dtpnaissance.visible := true;
  dtpmort.visible := true;
end;

procedure tfacteur.ecrirenon;
begin
  dbenom.ReadOnly := true;
  dbeprenom.ReadOnly := true;
  dbenationalite.ReadOnly := true;
  dbmbiographie.readonly := true;
  dbrsexe.ReadOnly := true;
  dbenaissance.visible := true;
  dbemort.visible := true;
  dtpnaissance.visible := false;
  dtpmort.visible := false;
end;

procedure TFACTEUR.bnouveauClick(Sender: TObject);
begin
   dbeprenom.SetFocus;
   datavideo.ASPlisteacteur.insert;
   dbgacteur.Enabled := false;
   fermeoption;
   ecrireoui;
   dtpnaissance.Date := date();
   dtpmort.Date := strtodate('31/12/2099');
end;
```

```
procedure TFACTEUR.FormCreate(Sender: TObject);
begin
  panelvalidation.Visible := false;
  benregistre.Enabled := false;
  valnom := '-';
  valprenom := '-';
  valnationalite := '-';
  panelrecherche.Visible := false;
  DTPnaissance.Visible := false;
  DTPmort.Visible := false;
end;

procedure TFACTEUR.bannuleClick(Sender: TObject);
begin
   datavideo.ASPlisteacteur.Cancel;
   benregistre.Enabled := false;
   ecrirenon;
   ouvreoption;
end;
```

```
procedure tfacteur.actualise;
begin
  datavideo.ASPlisteacteur.active := false;
  datavideo.ASPlisteacteur.active := true
end;
```

```
procedure TFACTEUR.benregistreClick(Sender: TObject);
begin
   if (datavideo.ASPlisteacteur.State = dsinsert) then
      begin
        with datavideo.ASPajoutacteur do
          begin
            parameters.parambyname('@nom').value :=  dbenom.text;
            parameters.ParamByName('@prenom').Value := dbeprenom.Text;
            parameters.ParamByName('@datenaissance').Value := dateof(dtpnaissance.Date);
            if (datetostr(dtpmort.Date) <> '31/12/2099')
               then parameters.ParamByName('@datemort').Value := dateof(dtpmort.date)
               else parameters.ParamByName('@datemort').Value := null;
            parameters.ParamByName('@nationalite').Value := dbenationalite.text;
            parameters.ParamByName('@biographie').Value := dbmbiographie.Text;
            parameters.ParamByName('@sexe').Value := dbrsexe.Value;
            execproc;
            cleacteur := parameters.parambyname('@cleacteur').value;
          end;
      end
   else
```

```
      begin
        cleacteur := strtoint(dbepkacteur.Text);
        with datavideo.ASPmodifacteur do
          begin
            parameters.ParamByName('@cleacteur').value := cleacteur;
            parameters.parambyname('@nom').value :=  dbenom.text;
            parameters.ParamByName('@prenom').Value := dbeprenom.Text;
            parameters.ParamByName('@datenaissance').Value := dateof(dtpnaissance.Date);
            if (datetostr(dtpmort.Date) <> '31/12/2099')
               then parameters.ParamByName('@datemort').Value := dateof(dtpmort.date)
               else parameters.ParamByName('@datemort').Value := null;
            parameters.ParamByName('@nationalite').Value := dbenationalite.text;
            parameters.ParamByName('@biographie').Value := dbmbiographie.Text;
            parameters.ParamByName('@sexe').Value := dbrsexe.Value;
            execproc;
          end;
      end;
  benregistre.Enabled := false;
  actualise;
  datavideo.ASPlisteacteur.Locate('pkacteur',cleacteur,[]);
  ecrirenon;
  ouvreoption;
end;
```

```
procedure TFACTEUR.DBEnomChange(Sender: TObject);
begin
   benregistre.Enabled := (trim(dbenom.Text) <> '');
end;

procedure TFACTEUR.bmodifieClick(Sender: TObject);
begin
   if (trim(dbepkacteur.Text) <> '') then
      begin
        cleacteur := strtoint(dbepkacteur.text);
        dbeprenom.SetFocus;
        datavideo.ASPlisteacteur.edit;
        fermeoption;
        ecrireoui;
        dtpnaissance.Date := datavideo.ASPlisteacteurdatenaissance.Value;
        if datavideo.ASPlisteacteurdatemort.value <> strtodate('30/12/1899')
            then dtpmort.Date := datavideo.ASPlisteacteurdatemort.value
            else dtpmort.Date := strtodate('31/12/2099');
      end;
end;
```

```
procedure TFACTEUR.bsupprimeClick(Sender: TObject);
begin
   if (messagedlg('Confirmez-vous la suppression de cet acteur ? ',
         mtconfirmation,[mbok,mbcancel],0) = mrok) then
      begin
         with datavideo.ASPsuppracteur do
          begin
             parameters.ParamByName('@cleacteur').Value := strtoint(dbepkacteur.Text);
             execproc;
          end;
         actualise;
      end;
end;

procedure TFACTEUR.bactualiseClick(Sender: TObject);
begin
   actualise;
end;

procedure TFACTEUR.bchercheClick(Sender: TObject);
begin
   fermeoption;
   panelvalidation.Visible := false;
   cleacteur := strtoint(dbepkacteur.Text);
   panelrecherche.Visible := true;
   datavideo.DTSacteur.DataSet := datavideo.ASPrechacteur;
   erecherche.SetFocus;
end;

procedure TFACTEUR.erechercheChange(Sender: TObject);
begin
  with datavideo.ASPrechacteur do
    begin
      active := false;
      parameters.ParamByName('@chaine').Value := erecherche.text;
      execproc;
      active := true;
    end;
end;
```

```
procedure TFACTEUR.DBGacteurTitleClick(Column: TColumn);
var
   cle : integer;
begin
   with DATAVIDEO.ASPlisteacteur do
      begin
        if (recordcount <> 0) then
          begin
            cle := strtoint(dbepkacteur.text);
            active := false;   // retour du curseur
            parameters.parambyname('@nomdecolonne').value :=  column.FieldName;
            if column.FieldName = 'nationalite' then
              begin
                 if valnationalite = '+' then valnationalite := '-'
                                         else valnationalite := '+';
                 parameters.ParamByName('@ordretri').value := valnationalite;
              end;
            if column.FieldName = 'nom' then
              begin
                 if valnom = '+' then valnom := '-'
                                 else valnom := '+';
                 parameters.ParamByName('@ordretri').value := valnom;
              end;
            if column.FieldName = 'prenom' then
              begin
                 if valprenom = '+' then valprenom := '-'
                                    else valprenom := '+';
                 parameters.ParamByName('@ordretri').value := valprenom;
              end;
            active := true;
            locate('pkacteur',cle,[]);
          end;
      end;
end;
```

```
procedure TFACTEUR.DBGacteurDrawColumnCell(Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumn;
  State: TGridDrawState);
var
   grille : tdbgrid;
begin
   grille := (sender as tdbgrid);
   if(grille.DataSource.DataSet.RecNo mod 2) =0
      then grille.Canvas.Brush.Color := clinfobk
      else grille.Canvas.Brush.Color := clmoneygreen;
   if state = [gdselected,gdfocused] then grille.Canvas.Brush.Color := clnavy;
   grille.DefaultDrawColumnCell(rect,datacol,column,state);
end;
```

Et dans UVIDEO :

```
procedure TDATAVIDEO.ASPlisteacteurAfterOpen(DataSet: TDataSet);
begin
   facteur.bmodifie.Enabled := ASPlisteacteur.recordcount > 0;
   facteur.bsupprime.Enabled := ASPlisteacteur.recordcount > 0;
   facteur.bcherche.Enabled := ASPlisteacteur.recordcount > 0;
   facteur.modifier.Enabled := ASPlisteacteur.recordcount > 0;
   facteur.supprimer.Enabled := ASPlisteacteur.recordcount > 0;
   facteur.chercher.Enabled := ASPlisteacteur.recordcount > 0;
end;
```

5.3. Lier une photo à un acteur.

Avant de nous lancer dans la mise en page et la programmation, il me semble opportun de rappeler quelques points importants :

- Quand je suis en mode suppression d'acteur, il ne faudra pas oublier que cet acteur ne pourra être vraiment enlevé que s'il n'y a plus aucune photo liée (intégrité référentielle). Il faut bannir l'effacement en cascade. On peut laisser le système agir, nous verrons les messages affichés.
- Quand je suis en mode modification des renseignements de l'acteur, plusieurs options coexistent :
  + ajout d'une photo liée
  + changement de photo liée
  + suppression d'une photo liée.

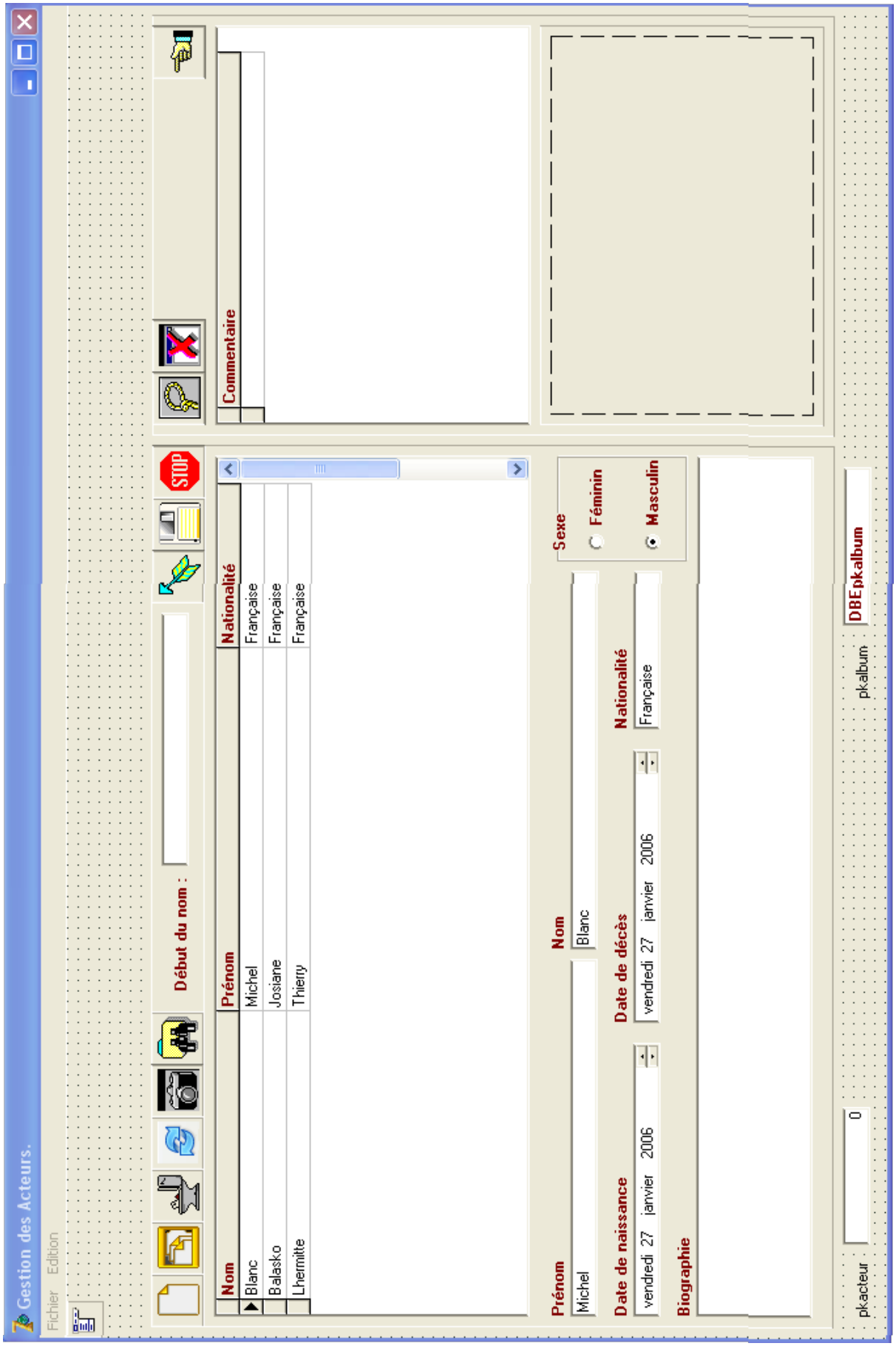
  On pourrait imaginer plusieurs solutions pour pratiquer ces opérations. J'ai retenu la technique suivante : ne permettre la liaison qu'à partir d'un nouveau bouton placé dans le PANELICONE (j'ai choisi un appareil photographique) et agrandir la fenêtre facteur. Ceci nous obligera à fixer la position et la taille de la fenêtre pour pouvoir procéder à la modification lors de l'agrandissement et du retour à la normale.

Voici une idée du nouveau menu :

Ce nouveau bouton ne sera accessible qu'avec PANELICONE visible, c'est à dire quand nous sommes positionnés sur un acteur. Si nous créons un nouvel acteur, la liaison avec des photos se fera en passant par cette nouvelle icône

Toutes ces options nécessitent des procédures stockées qui concernent la table de liaison entre PHOTO et ACTEUR, c'est à dire la table ALBUM. On nommera ces trois procédures stockées : AjoutAlbum, ListeAlbum et SupprAlbum.

Voici la fenêtre imaginée.


Gestion des Acteurs.
Fichier
Edition
Début du nom :
STOP
Nom
Prénom
Nationalité
Blanc
Michel
Française
Balasko
Josiane
Française
Lhermitte
Thierry
Française
Commentaire
Sexe
Féminin
Masculin
Nom
Blanc
Prénom
Michel
Date de décès
vendredi 27 janvier 2006
Nationalité
Française
Date de naissance
vendredi 27 janvier 2006
Biographie
pkacteur
0
pkalbum
DBEpkalbum

Dans la nouvelle partie de la fenêtre, on découvre une grille qui affichera les commentaires joints aux photos liées à l'acteur en cours. Cette grille devra être remplie immédiatement par la procédure ListeAlbum. Sous cette grille, un composant Timage montrera la photo (si elle existe) qui détient le focus dans la grille du dessus. Comme le nombre de photos par acteur est illimité mais je suppose raisonnable, on peut se permettre de monter les photos automatiquement à chaque mouvement.
Au-dessus de la grille, on trouve trois boutons :

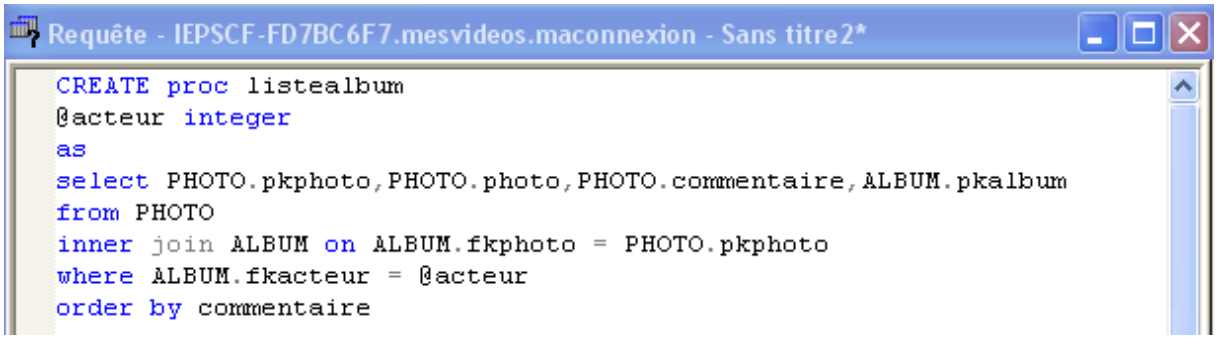

Il nous enverra sur la fenêtre des photos pour pouvoir les traiter. (bnouvellephoto)

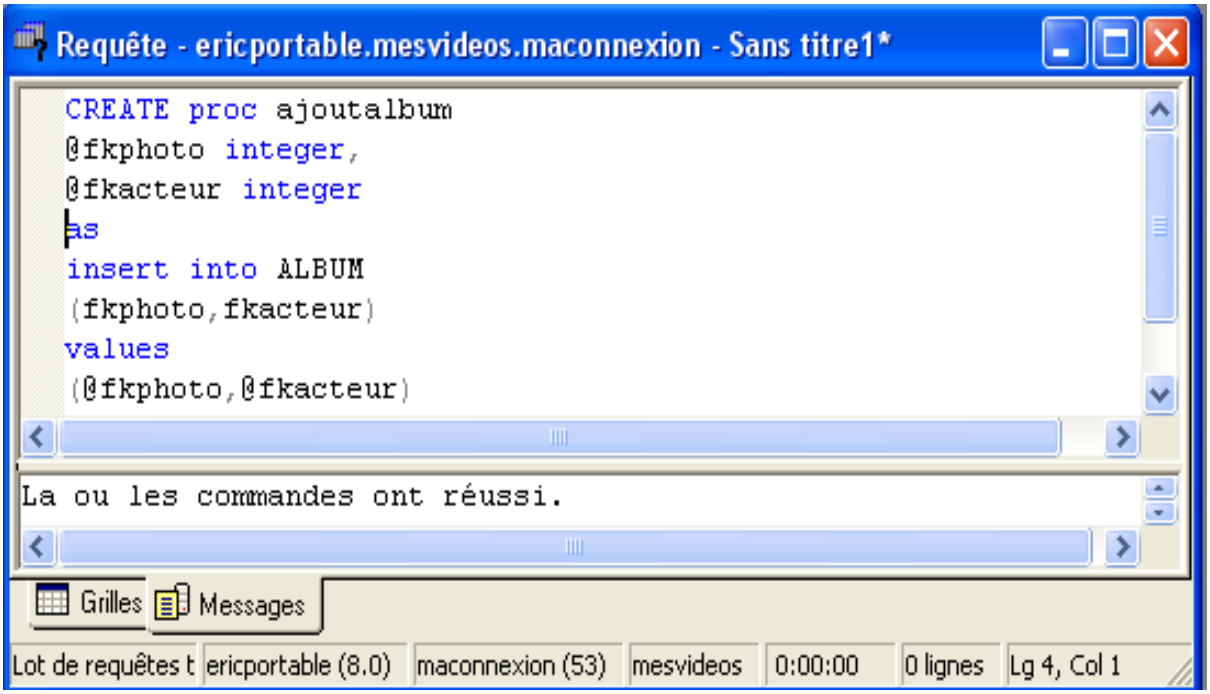

Il supprime (après confirmation) le lien entre la photo et l'acteur. (bdelier)

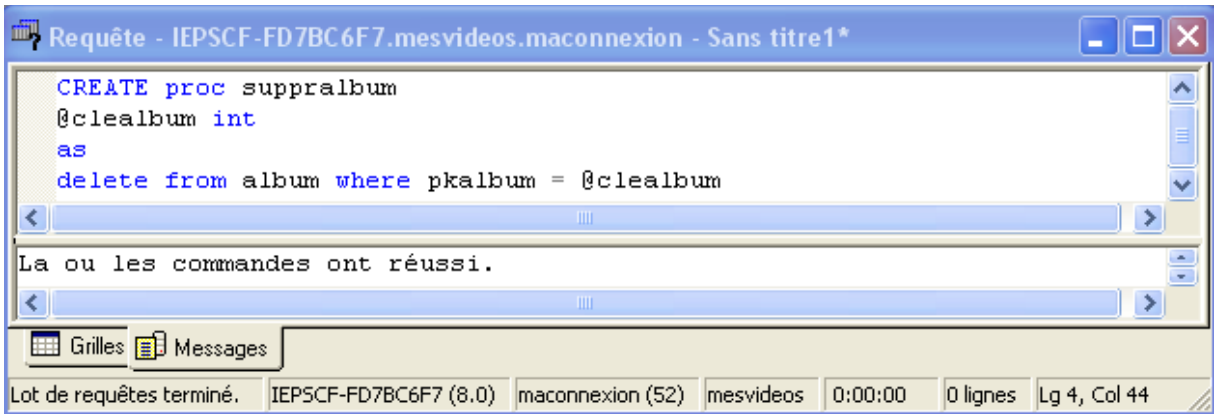

Il dimensionne la fenêtre à sa taille de départ. (breferme)

5.4. Remplir la grille de l'album automatiquement.

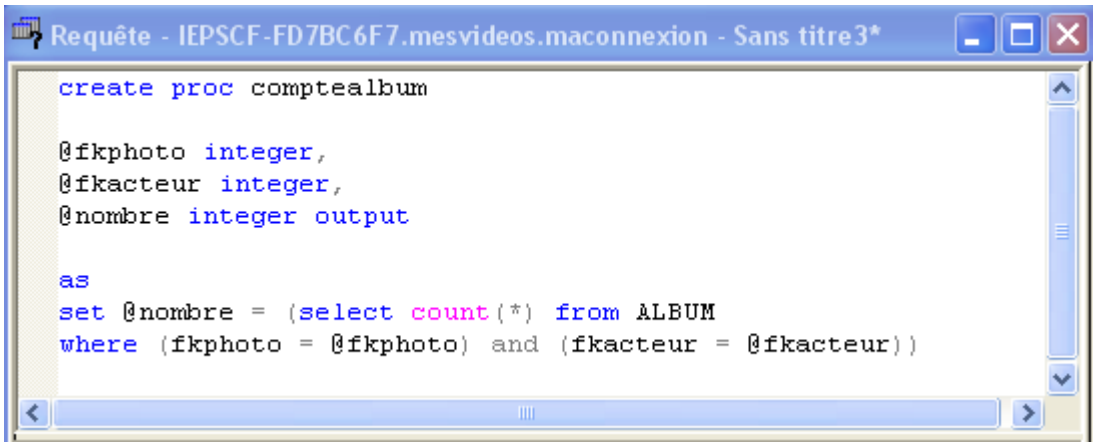
Voici la procédure stockée de listage. Il s'agit d'une jointure entre ma table photo et ma table album. Elle reçoit un paramètre qui est la clé primaire de l'acteur. Cette clé primaire acteur est une clé étrangère dans la table album. Vous trouverez également les procédures d'ajout et de suppression.



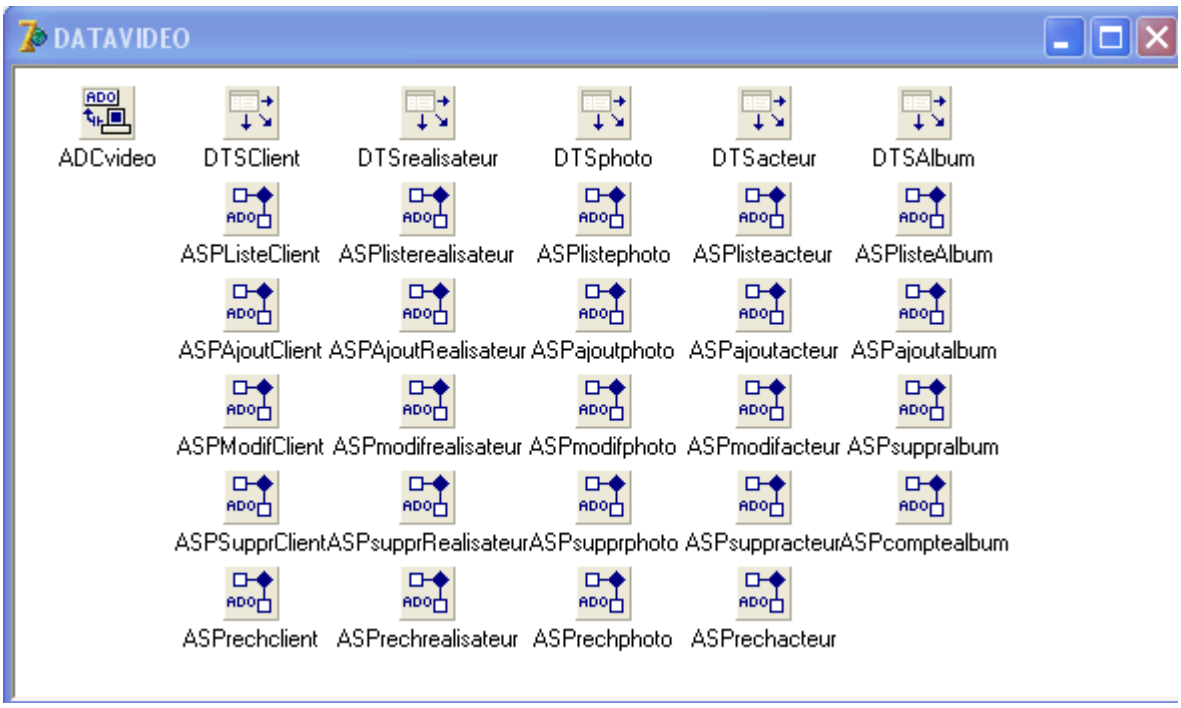




Je vais y ajouter une procédure qui nous renvoie le nombre de photos liées à un acteur. Cette procédure nous sera très utile pour éviter les doublons (deux fois la même photo pour un acteur).



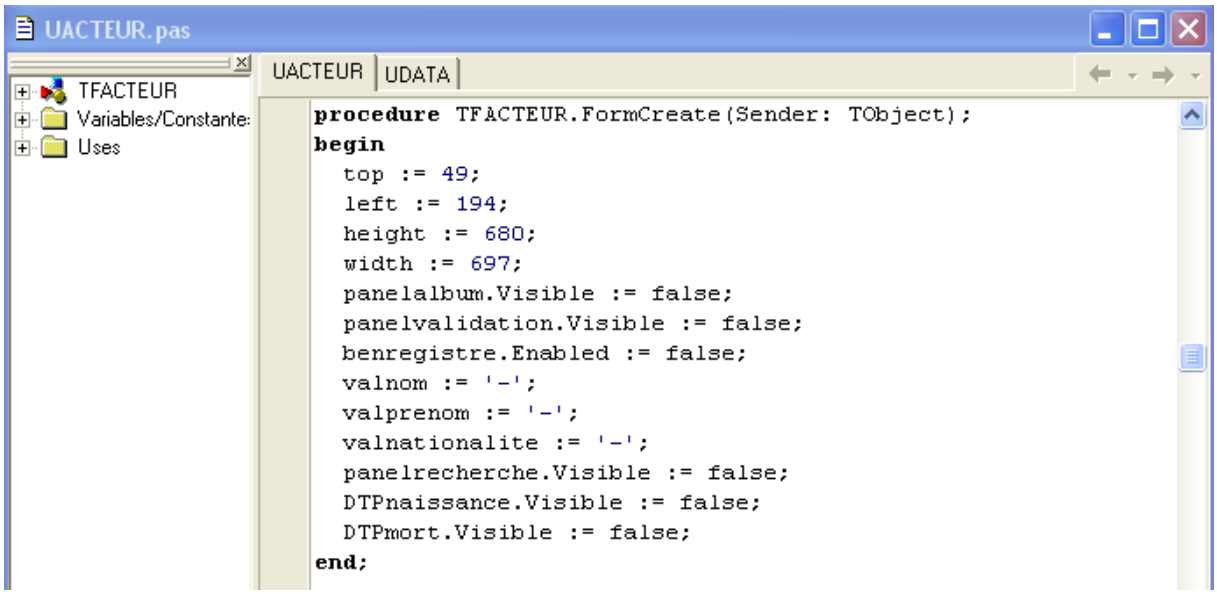
Nous pouvons ajouter les composants dans datavideo.

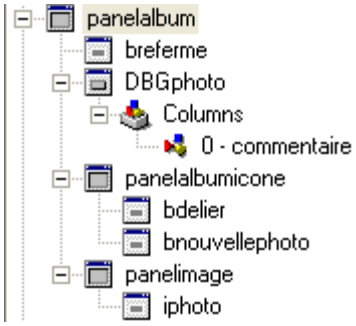


Dans l'éditeur de champs de ASPListeAlbum, ne pas oublier d'ajouter tous les champs.

Il faut laisser la propriété active de ASPListeAlbum à false, car c'est le bouton bphoto de la fenêtre facteur qui activera la procédure à la demande. Quand nous rendrons à la fenêtre sa taille normale, nous en profiterons pour désactiver la procédure.

Commençons par dimensionner (taille normale) la fenêtre lors de sa création (formcreate) :





Voici les composants à ajouter dans la partie supplémentaire de la fenêtre :

Un panel 'PANELALBUM) supportera les autres composants. Le reste est connu.

Voici la procédure d'agrandissement de la fenêtre :

```
procedure TFACTEUR.bphotoClick(Sender: TObject);
begin
  top := 30;
  left := 1;
  height := 680;
  width := 1024;
  panelalbum.Visible := true;
  with DATAVIDEO.ASPlistealbum do
      begin
        active := false;   // retour du curseur
        parameters.parambyname('@acteur').value :=  strtoint(dbepkacteur.text);
        active := true;
      end;
  breferme.Visible := true;
  dbgacteur.Enabled := false;
  fichier.Enabled := false;
  edition.Enabled := false;
  panelicone.Visible := false;
  dbrsexe.Enabled := false;
  dbeprenom.enabled := false;
  dbenom.Enabled := false;
  dbenationalite.Enabled := false;
  dbmbiographie.Enabled := false;
  dbenaissance.Enabled := false;
  dbemort.Enabled := false;
end;
```

Les quatre premiers paramètres fournissent respectivement la position supérieure gauche et la taille.
Pour éviter l'apparition d'ascenseurs en bas et à droite de la fenêtre en taille normale, il faut rendre invisibles les composants placés dans la partie droite de la fenêtre. Vous constaterez que les éléments de la partie gauche sont rendus inutilisables pendant que la partie droite est visible.

Il est évident que tous ces paramètres seront inversés lors du retour. Voici la procédure de fermeture.

```
procedure TFACTEUR.brefermeClick(Sender: TObject);
begin
  top := 49;
  left := 194;
  height := 680;
  width := 697;
  panelalbum.Visible := false;
  DATAVIDEO.ASPlistealbum.Active := false;
  breferme.Visible := false;
  dbgacteur.Enabled := true;
  fichier.Enabled := true;
  edition.Enabled := true;
  iphoto.Visible := false;
  panelicone.Visible := true;
  dbrsexe.Enabled := true;
  dbeprenom.enabled := true;
  dbenom.Enabled := true;
  dbenationalite.Enabled := true;
  dbmbiographie.Enabled := true;
  dbenaissance.Enabled := true;
  dbemort.Enabled := true;
end;
```

Voici la procédure de suppression de liaison et celle d'actualisation.

```
procedure TFACTEUR.bdelierClick(Sender: TObject);
begin
   if (messagedlg('Confirmez-vous la suppression de cette liaison ? ',
         mtconfirmation,[mbok,mbcancel],0) = mrok) then
      begin
         with datavideo.ASPsuppralbum do
           begin
              parameters.ParamByName('@clealbum').Value := strtoint(dbepkalbum.Text);
              execproc;
           end;
         actualisealbum;
      end;
end;

procedure tfacteur.actualisealbum;
begin
  datavideo.ASPlistealbum.active := false;
  datavideo.ASPlistealbum.active := true
end;
```

Et enfin : la procédure d'appel de la fenêtre fphoto.

```
procedure TFACTEUR.bnouvellephotoClick(Sender: TObject);
begin
   fphoto.brentree.Visible := true;
   uphoto.envoyeur := 1;
   uphoto.cleacteur := strtoint(dbepkacteur.Text);
   fphoto.BorderIcons := [];
   fphoto.ShowModal();
   actualisealbum;
end;
```

Cette dernière procédure demande quelques points d'éclaircissement :

- rendre visible le bouton de retour dans la fenêtre fphoto
- je passe un paramètre (inutile pour l'instant) pour savoir depuis quelle fenêtre on a ouvert la fenêtre fhoto
- je transmets la valeur de l'acteur courant
- Pour imposer un retour propre, je supprime momentanément les icônes du coin supérieur droit de la fenêtre appelée
- la fonction showmodal() lance la fenêtre dont le nom est précisé juste devant. Attention : cette façon de faire impose le travail dans la fenêtre fphoto.
- attention : quand cette ligne est exécutée par le programme, vous vous trouvez dans la fenêtre appelée. Ce qui signifie que les lignes qui suivent ne seront exécutées qu'au retour de la fenêtre appelée
- j'actualise ma grille de visualisation.

5.5. Montrer les photos automatiquement.

Chaque fois que je me déplace dans la grille des liaisons, la photo liée doit être montrée (si elle existe). Je dois le faire dans UDATA. On y retrouve la fonction personnelle (ne pas oublier de la déclarer dans l'interface) departjpegblob. Ne pas oublier non plus de modifier les uses.

```
function TDATAVIDEO.departjpegblob(image : tblobfield) : integer;
var
  bs : tadoblobstream;
  buffer : word;
  hx : string;
begin
  result := -1;
  bs := tadoblobstream.create(image, bmread);
  try
    while(result = -1) and(bs.position + 1 < bs.size) do
      begin
        bs.readbuffer(buffer,1);
        hx := inttohex(buffer,2);
        if hx = 'FF' then
           begin
             bs.readbuffer(buffer,1);
             hx := inttohex(buffer,2);
             if hx = 'D8' then result := bs.position - 2
                          else if hx = 'FF' then bs.position := bs.Position - 1;
           end;
      end;
  finally
    bs.free;
  end;
end;
```

Pour montrer la photo concernée lors de chaque déplacement dans la grille, il faut utiliser l'événement AfterScroll de ASPListeAlbum.

Pensons aussi à la procédure d'ouverture d'ASPLIsteAlbum qui cache qui montre le bouton d'effacement des liaisons.

```
procedure TDATAVIDEO.ASPlisteAlbumAfterOpen(DataSet: TDataSet);
begin
   facteur.bdelier.Enabled := ASPlistealbum.recordcount > 0;
end;
```

```
procedure TDATAVIDEO.ASPlisteAlbumAfterScroll(DataSet: TDataSet);
var
   phototmp : tjpegimage;
   bs : tadoblobstream;
begin
   if (ASPlistealbumphoto.Value <> '') then
      begin
         facteur.iphoto.visible := true;
         bs := tadoblobstream.Create(ASPlistealbumphoto,bmread);
         if bs.Size <> 0 then
            begin
              try
                 bs.Seek(departjpegblob(ASPlistealbumphoto),sofrombeginning);
                 phototmp := tjpegimage.Create;
                 try
                    phototmp.LoadFromStream(bs);
                    facteur.iphoto.Picture.Graphic := phototmp;
                 finally
                    phototmp.Free;
                 end;
              finally
                 bs.Free;
              end;
            end;
      end;
end;
```

Cette procédure n'est pas inconnue : on l'a déjà utilisée en partie pour la table des photos.

Avant de modifier la fenêtre des photos en ajoutant le bouton de retour, pensons à colorier notre grille en plaçant comme événement DrawColumnCell la procédure DBGacteurDrawColumnCell qui peut être utilisée car elle ne reçoit aucun paramètre.

5.6. Le bouton de retour de la fenêtre des photos.

Je commence par obtenir le nombre de photos dont le numéro d'acteur et le numéro de photo sont identiques. Ceci m'évitera les doublons dans la table de liaison.

Au moment où on clique sur le bouton de retour, un message demande si on souhaite lier la photo courante à l'acteur courant. Si oui, la liaison est faite dans la table album (sauf si elle existait déjà).

Il faut encore rendre invisible le bouton de retour, replacer les icônes du coin supérieur droit visibles pour la prochaine utilisation de la fenêtre des photos et enfin fermer la fen^tre des photos et rendre la main à la fenêtre appelante. L'instruction modalresult agit dans ce sens et peut même transmettre un paramètre entier à la fonction ou procédure appelante.

```
procedure TFPHOTO.brentreeClick(Sender: TObject);
var
  nombre : integer;
begin
  with datavideo.ASPcomptealbum do
     begin
        parameters.ParamByName('@fkphoto').Value := strtoint(dbepkphoto.Text);
        parameters.ParamByName('@fkacteur').Value := cleacteur;
        execproc;
        nombre := parameters.parambyname('@nombre').value;
     end;
  if (trim(dbepkphoto.text) <> '') and
     (messagedlg('Confirmez-vous la liaison de cette photo à cet acteur ? ',
      mtconfirmation,[mbok,mbcancel],0) = mrok) and
      (nombre = 0) then
      begin
        with datavideo.ASPajoutalbum do
          begin
             parameters.ParamByName('@fkphoto').Value := strtoint(dbepkphoto.Text);
             parameters.ParamByName('@fkacteur').Value := cleacteur;
             execproc;
          end;
      end;
  bretour.Visible := false;
  fphoto.BorderIcons := [biSystemMenu,biMinimize,biMaximize];
  modalresult := 1;
end;
```

Posons-nous maintenant la question de l'utilité d'une table réalisateur et d'une table acteur.

Si nous fusionnons ces deux tables, nous allons y trouver des avantages mais aussi des inconvénients.

Avantage :

- Les réalisateurs pourront aussi avoir leur(s) photo(s)
- Si un réalisateur est aussi un acteur, nous ne l'encoderons qu'une seule fois, ce qui nous fera gagner du temps mais aussi de l'espace disque.

Inconvénients :

- Il va falloir créer deux relations entre les films et les acteurs – réalisateurs au lieu d'une
- Une recherche sur un réalisateur effectuera aussi cette recherche sur les acteurs.

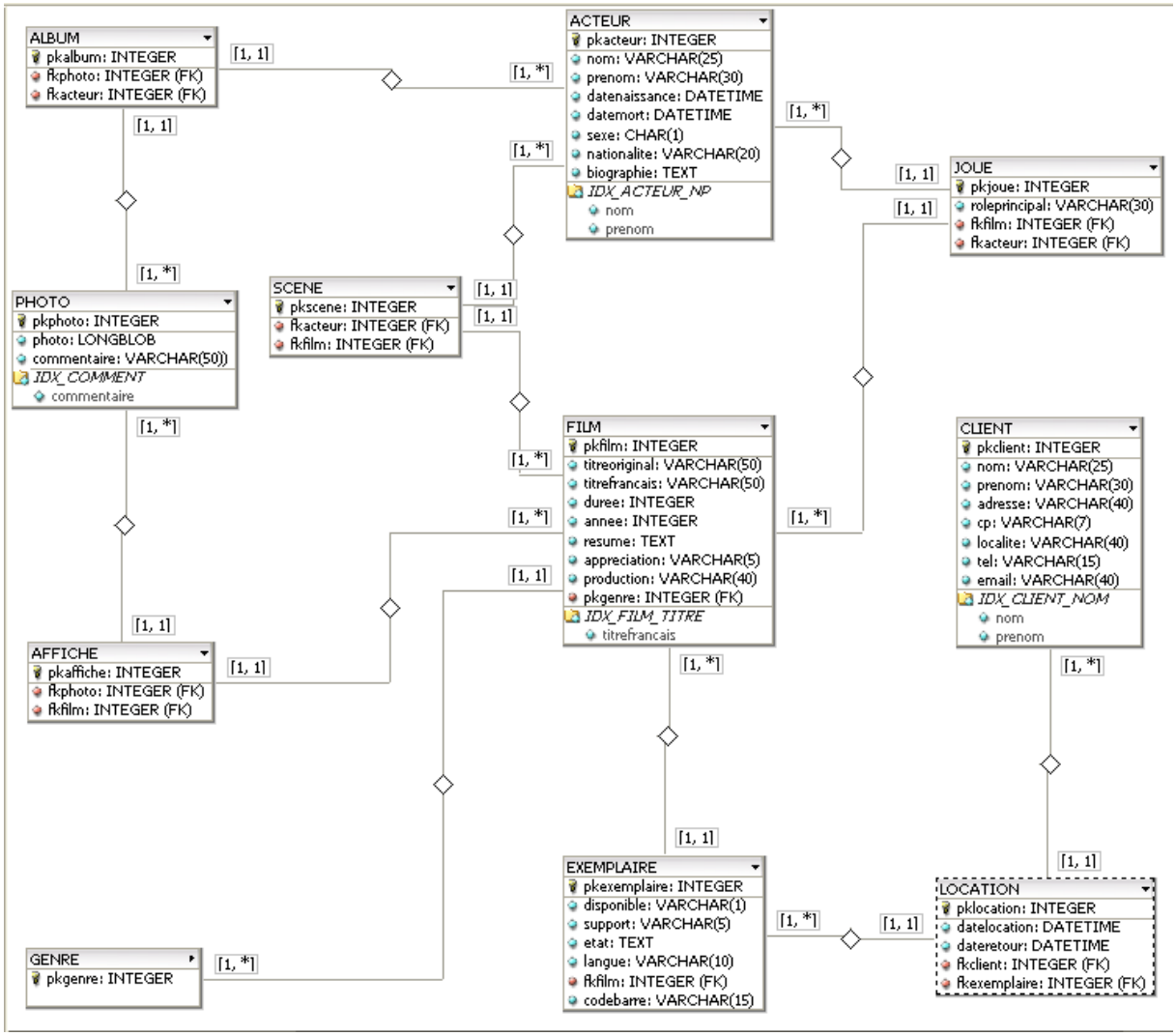
Je pense toutefois que le jeu en vaut la chandelle.

6. FUSION DES TABLES ACTEURS ET REALISATEURS.

Le travail va consister essentiellement à supprimer des procédures et événements et à modifier un peu l'aide et les titres des fenêtres. Il vaut toutefois mieux le faire avant d'aborder les films qui seront liés doublement à cette table fusionnée.

6.1. Nouvelle structure de la base de données.

La structure de ma base de données est ainsi modifiée :



Ceci va nous imposer de supprimer la table REALISATEUR et de modifier la clé étrangère de la table SCENE. Je décide de la renommer fkacteur pour coller à la clé primaire de ACTEUR.

En pratique : comment procéder ?

1° Faire un backup de ma base de données
2° Faire un backup de mon travail de programmation
3° Dans delphi, supprimer l'unité UREALISATEUR et les procédures annexes
4° Dans SQL SERVER, supprimer les procédures stockées liées à la table REALISATEUR, supprimer la clé étrangère fkrealisateur dans la table SCENE et recréer la clé étrangère fkacteur puis supprimer REALISATEUR.

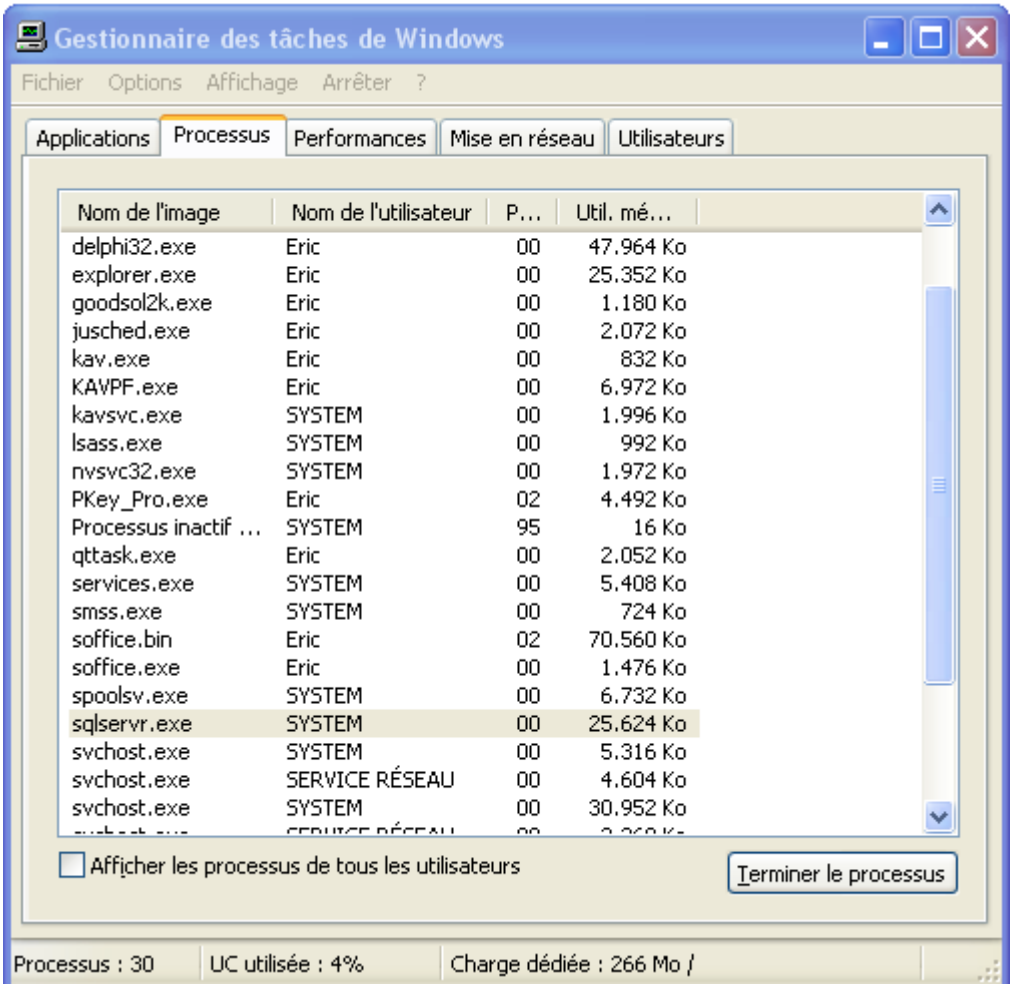
Il est indispensable de respecter cet ordre pour éviter des problèmes de connexion. Voyons le détail.

6.2. Backup de la base de données.

Vous trouverez dans SGBDR.PDF la marche à suivre pour des backups. Ici, je choisirai la copie manuelle.

1° Fermer le SERVICE MANAGER de SQL SERVER s'il est en route.
- clic droit sur l'icône du SERVICE MANAGER
- quitter

2° Vérifier dans le gestionnaire des tâches de Windows (ALT + CTRL + DEL) s'il ne reste pas un processus comme sqlservr.exe en route. Si oui, terminer ce processus.


Gestionnaire des tâches de Windows
Fichier Options Affichage Arrêter ?
Applications Processus Performances Mise en réseau Utilisateurs
Nom de l'image Nom de l'utilisateur P... Util. mé...
delphi32.exe Eric 00 47.964 Ko
explorer.exe Eric 00 25.352 Ko
goodsol2k.exe Eric 00 1.180 Ko
jusched.exe Eric 00 2.072 Ko
kav.exe Eric 00 832 Ko
KAVPF.exe Eric 00 6.972 Ko
kavsvc.exe SYSTEM 00 1.996 Ko
lsass.exe SYSTEM 00 992 Ko
nvsvc32.exe SYSTEM 00 1.972 Ko
PKey_Pro.exe Eric 02 4.492 Ko
Processus inactif ... SYSTEM 95 16 Ko
qttask.exe Eric 00 2.052 Ko
services.exe SYSTEM 00 5.408 Ko
smss.exe SYSTEM 00 724 Ko
soffice.bin Eric 02 70.560 Ko
soffice.exe Eric 00 1.476 Ko
spoolsv.exe SYSTEM 00 6.732 Ko
sqlservr.exe SYSTEM 00 25.624 Ko
svchost.exe SYSTEM 00 5.316 Ko
svchost.exe SERVICE RÉSEAU 00 4.604 Ko
svchost.exe SYSTEM 00 30.952 Ko
Afficher les processus de tous les utilisateurs
Terminer le processus
Processus : 30 UC utilisée : 4% Charge dédiée : 266 Mo /

Confirmer la fin du processus (demandé par un message d'alerte).

3° Copier les fichiers mesvideos.mdf et mesvideos_log.ldf et les coller dans un répertoire de sauvegarde où nous sauvegarderons aussi les fichiers DELPHI par exemple sauvemesvideos.

6.3. Backup des informations DELPHI.

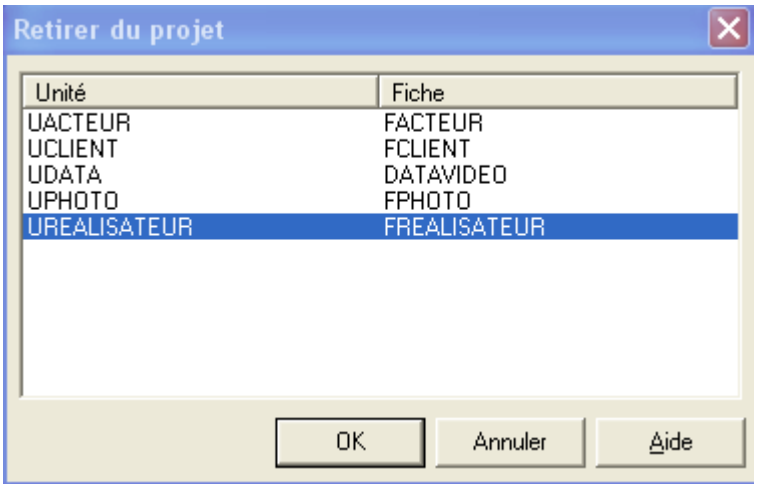
Copier tous les fichiers du répertoire choisi pour placer les fichiers DELPHI et les copier dans le répertoire du point 6.2 (pas la peine de copier DBVIDEO.EXE).

6.4. Suppression dans DELPHI.

Commençons par relancer le SERVICE MANAGER de SQL SERVER pour pouvoir établir la connexion.
Lançons DELPHI et chargeons mesvideos.

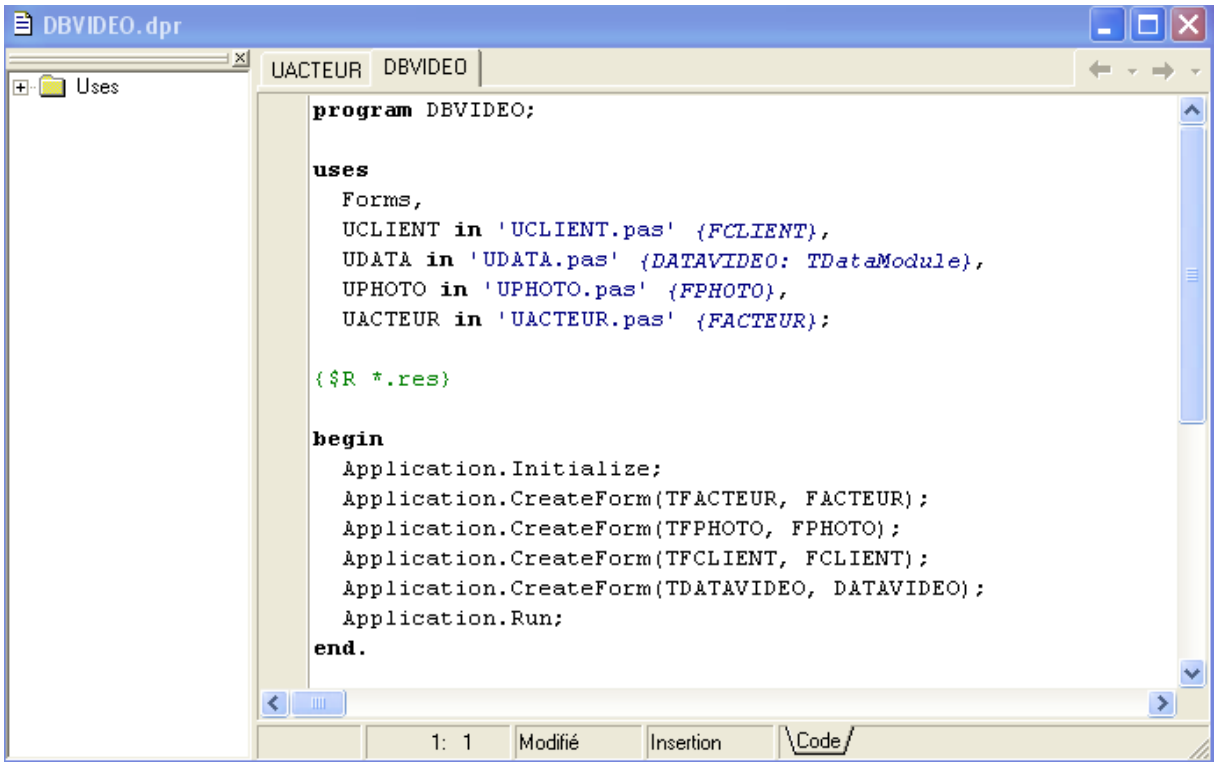
1° Retirer UREALISATEUR du projet.

Projet > retirer du projet >



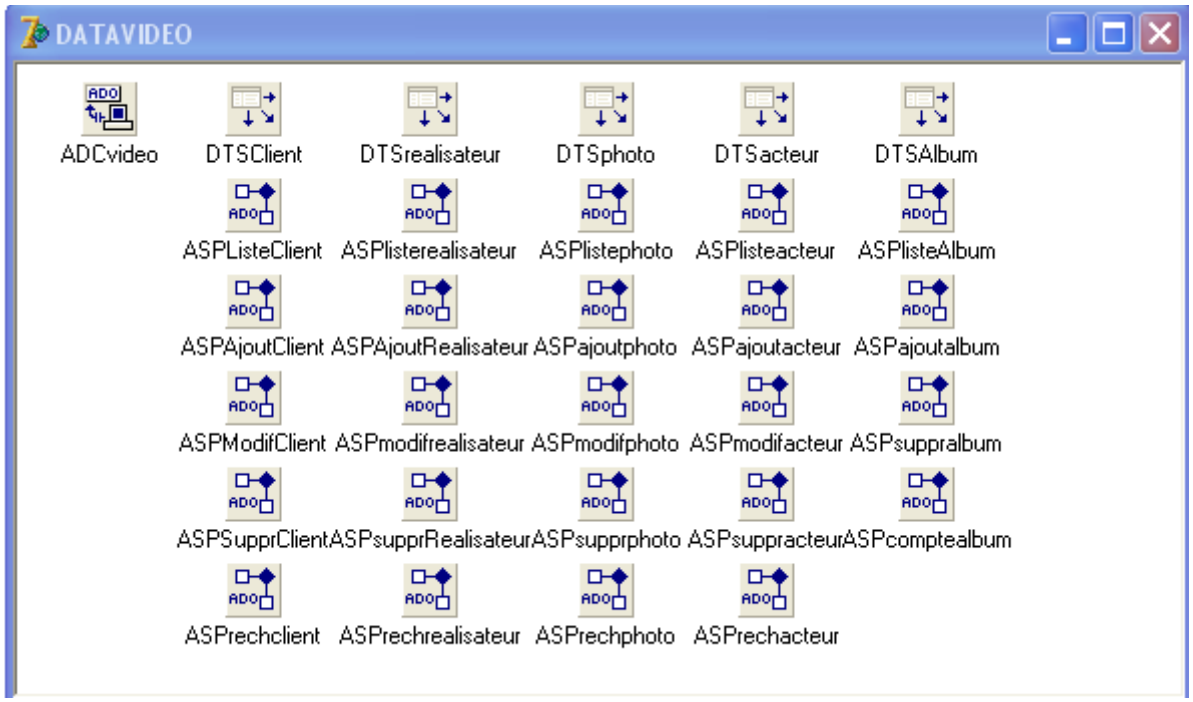
Cliquer sur OK puis confirmer.

2° Vérifier que c'est bien enlevé du projet. (Voir une unité ou CTRL F12 et DBVIDEO).

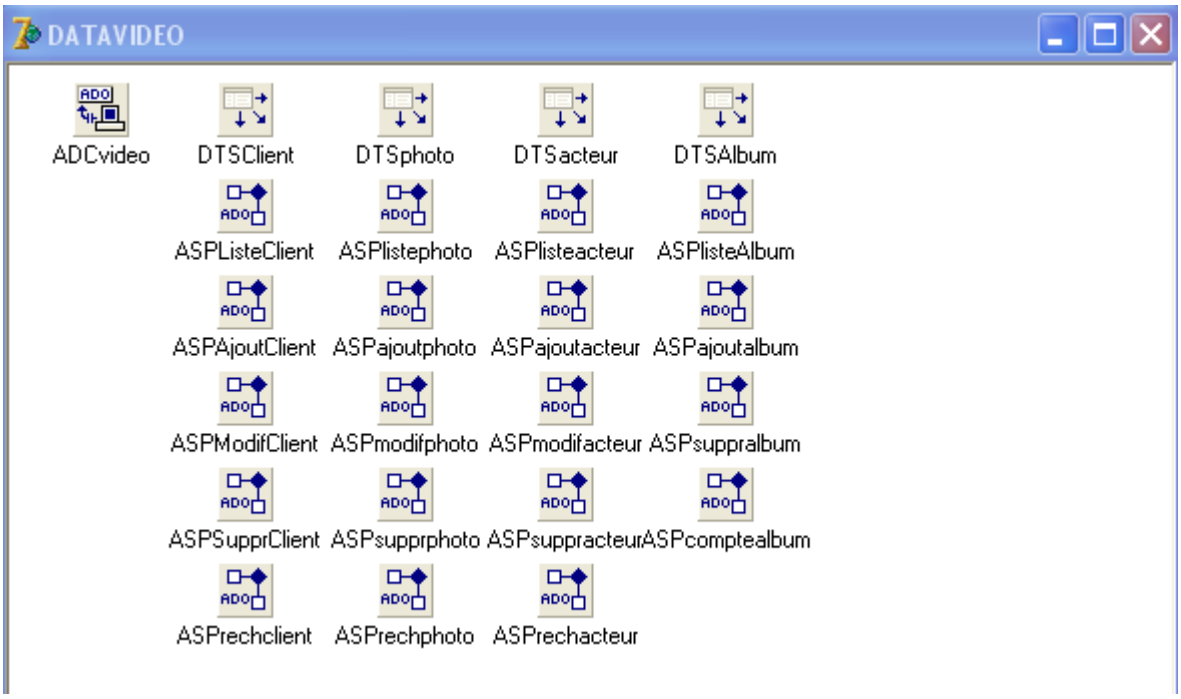


L'unité ne fait plus partie du projet.

3° Dans la fenêtre DATAVIDEO, retirer les six icônes relatives aux réalisateurs.



Ce qui nous donne :



4° Dans UDATA, retirer Urealisateur des uses et supprimer la procédure ASPlisterealisateurAfterOpen (deux fois).

5° Désactiver la connexion (propriété connected devient false) puis sauvegarder et fermer DELPHI.

6° Avec l'explorateur Windows, se rendre dans le répertoire des fichiers DELPHI. Y supprimer tous les fichiers UREALISATEUR.* (ils sont sept normalement).

7° Relancer DELPHI. Charger mesvideos, remettre le propriété connected de la connexion à true et ne pas oublier de remettre à true la propriété active de toutes les ASPliste... sauf ASPListeAlbum. En effet, déconnecter de la base de données désactive les procédures stockées liées dans DELPHI.

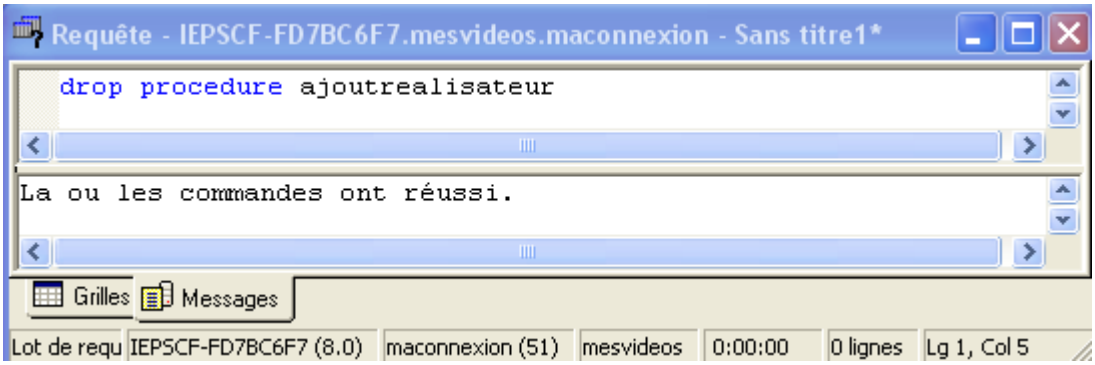
8° Lancer une compilation. Ceci doit fonctionner.

6.5. Modifications dans SQL SERVER.

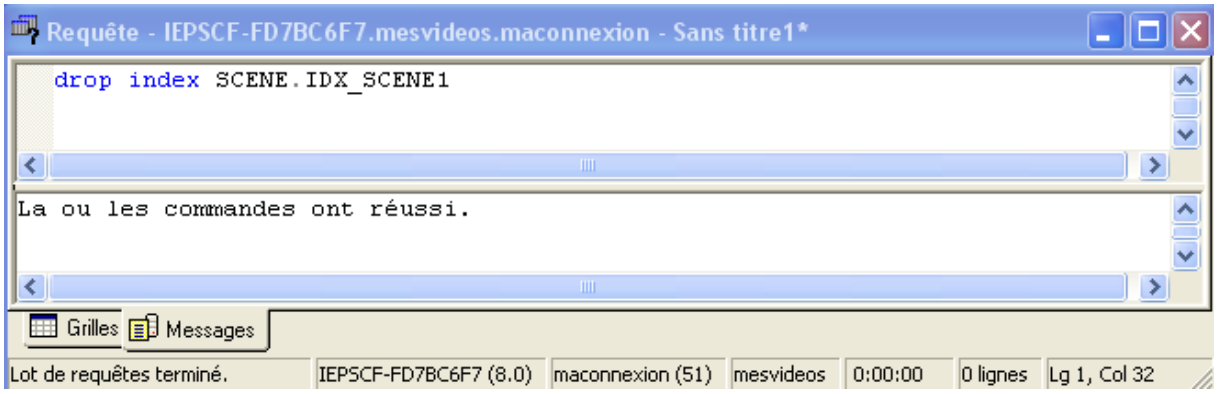
1° Déconnecter la connexion dans le programme DELPHI et fermer DELPHI.

2° Lancer l'analyseur de requêtes. Bien se positionner dans l'onglet MESVIDEOS. Supprimer toutes (elles sont cinq) les procédures stockées relatives aux réalisateurs : clic droit sur la procédure > supprimer et confirmer

ou

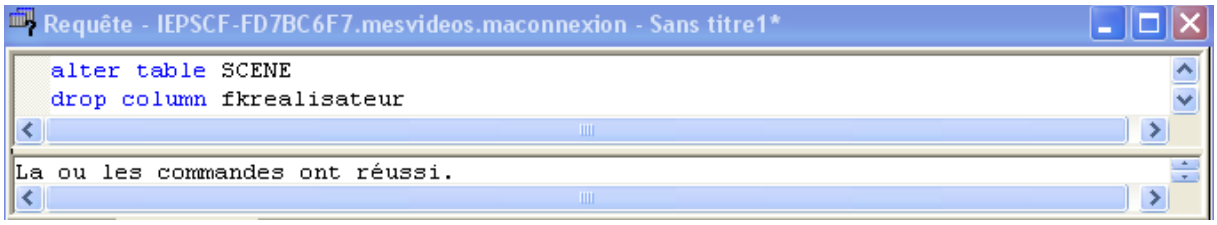


3° Supprimer la clé étrangère fkrealisateur dans SCENE. Pour pouvoir le faire, il faut d'abord supprimer l'index qui y est joint. Il se nomme IDX_SCENE1 (voir séquence de création et requête CREATE INDEX IDX_SCENE1 ON SCENE (fkrealisateur)). Son nom complet est SCENE.IDX_SCENE1.

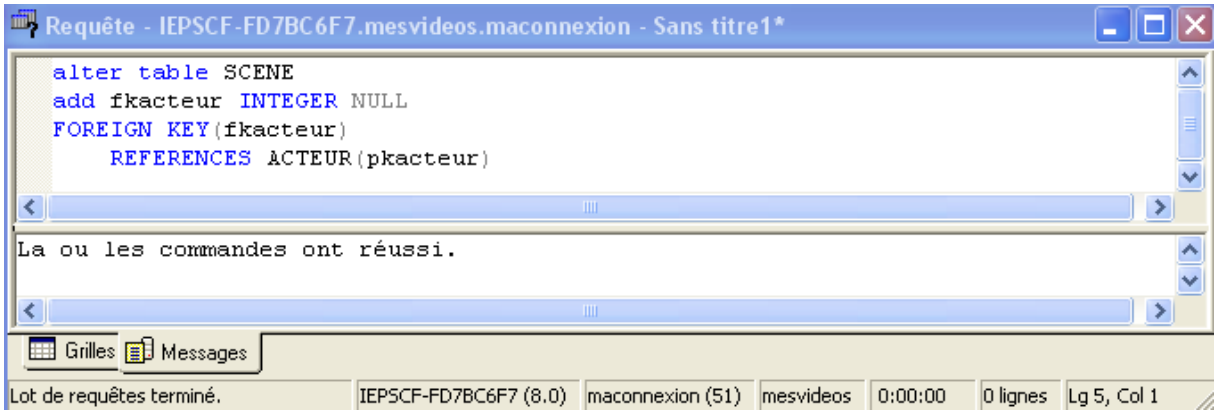


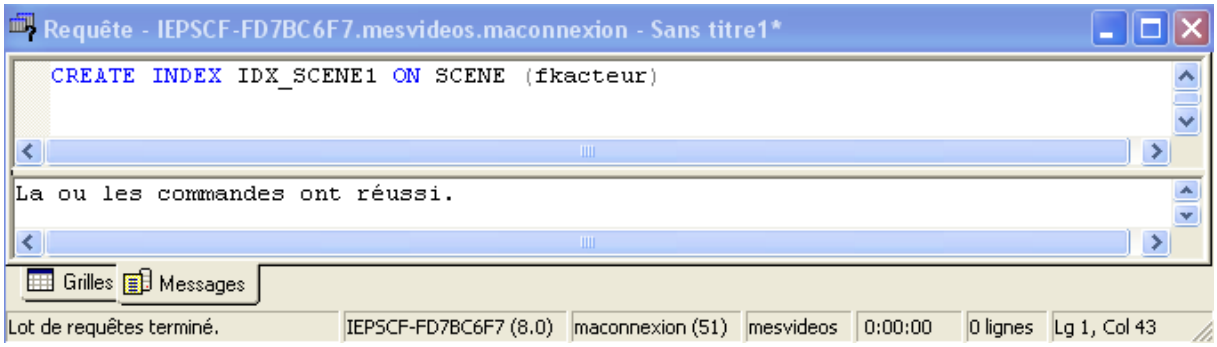
On va ensuite supprimer la contrainte de clé étrangère liée. Un clic droit sur la contrainte concernée puis supprimer. (chez moi, c'était FK__SCENE__fkrealisa__0BC6C43E).

Nous pouvons maintenant supprimer la clé elle-même.

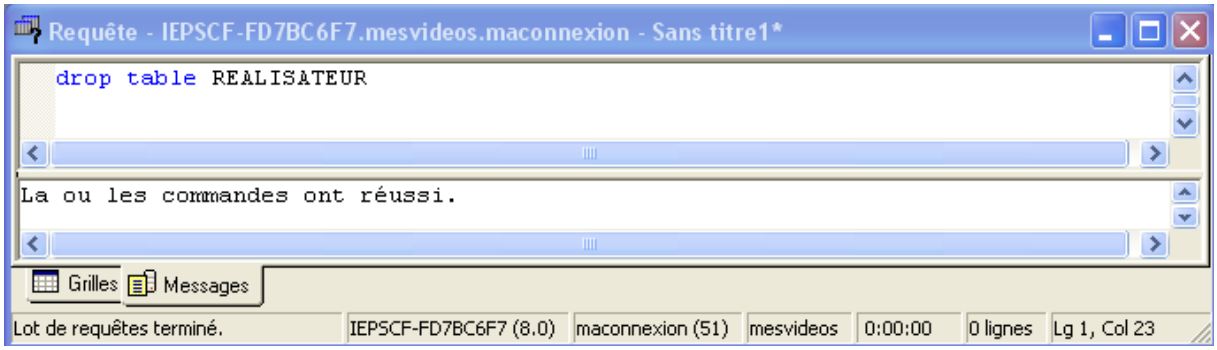


4° Recréer la colonne fkacteur, sa contrainte de clé étrangère et son index.





5° Supprimer la table REALISATEUR.



6.6. La finition.

Il nous reste à faire l'une ou l'autre modification dans le programme DELPHI.
Avant, n'oublions pas de nous reconnecter à la base de données avec notre programme (connected à true et trois procédures stockées avec active à true).

Il faut changer le titre et les aides des icônes, quand vous remarquerez la nécessité de faire une modification.

Table des matières