Les macros complémentaires avec Excel

Macros, présentation

Excel intègre dorénavant un langage de programmation, VBA, qui permet de réaliser des applications puissantes et conviviales, allant bien au-delà ce que qu'évoque le terme de macro (cf. l'historique ci-dessous). Malheureusement, parmi les utilisateurs d'Excel, presque personne n'en est conscient.

Historique

Une macro est un terme générique pour désigner un moyen de mémoriser un enchaînement de tâches au sein d'un logiciel.

Il est ensuite possible d'utiliser la macro – l'expression standard est "lancer la macro" – pour répéter le même enchaînement. Plusieurs méthodes peuvent être employées pour lancer une macro :

- On peut associer un raccourci clavier à la macro.
- L'application peut comporter une commande provoquant l'affichage de la liste des macros disponibles. Il suffit alors de choisir celle que l'on veut lancer et de valider.
- Il est parfois possible également d'afficher un bouton, éventuellement situé dans une barre d'outils. Un simple clic sur le bouton provoque l'exécution de la macro.

Dans la majorité des cas, les macros ne permettent de réaliser que des choses très simples, par exemple la mise en page d'un document (orientation et marges) dans un traitement de texte. Les macros sont alors des programmes automatiques. Dans ce contexte, il est parfois possible d'accéder au code de la macro, c'est-à-dire au programme, pour modifier tel ou tel paramètre. Cette option améliore les choses, mais les possibilités restent cependant limitées.

En tant qu'outil de programmation, ces macros manquaient à la fois de souplesse et de puissance. C'est pourquoi les informaticiens les ont toujours considérées avec condescendance.

Ce type de macros a tout de même rendu service a un certain nombre d'utilisateurs, grâce au temps gagné.

Macros et tableurs

Lotus 1,2,3 a probablement été le premier tableur comportant des macros. Celles-ci étaient relativement puissantes, mais leur lecture difficile, car chaque instruction était un code de 3 ou 4 lettres, éventuellement complété de paramètres.

J'ai eu l'occasion de voir des macros réalisées avec Lotus, mais je n'en ai jamais utilisée ni *a fortiori* écrite. Toutefois, je pense que leur syntaxe et l'emploi de code abscons devaient limiter leur usage et leur maintenance.

Excel est sorti en 1985. La première version tournait sur Mac exclusivement, nécessitait 512 Ko et tenait sur une disquette. Mais, merveille, il y avait un langage macro relativement puissant et lisible.

On était encore loin d'un langage de programmation classique : il manquait en particulier la possibilité de créer des menus et des zones de dialogue, et les structures classiques de boucles.

Malgré ces limitations, l'outil était suffisamment puissant pour que j'ai pu créer et commercialiser en 1987, toujours avec la version 1.0 d'Excel, une application commerciale, Mac Bilan, qui complétait les logiciels de compta (à l'époque Maestria et Gestion Simil).

Mac Bilan permettait à partir d'une balance (la liste des comptes d'une entreprise avec leurs soldes) d'établir le bilan, le compte de résultat et le détail des postes.

En outre, un certain nombre de contrôles comptables étaient effectués, et il était possible de saisir des écritures complémentaires à prendre en compte pour le bilan.

Le tout avec Excel 1.0 !

Macros XL4 et VBA

Ce langage macro, a évolué au fil des versions d'Excel jusqu'à la version 4.0 (d'où son nom actuel de macros XL4) et est devenu de plus en plus puissant, en conservant malgré tout un positionnement un peu bâtard :

Trop intimement lié à Excel, et trop différent des langages classiques, il n'a jamais réussi à surmonter la condescendance de l'immense majorité des informaticiens. D'autre part, acquérir une maîtrise de ce langage macro, suffisante pour réaliser des applications utilisables par des tiers, demandait à un non informaticien de formation, une motivation et un investissement en temps considérable. La plupart de ceux qui ont essayé n'avaient pas les capacités nécessaires ou se décourageaient trop tôt.

Le résultat est que très peu de gens étaient capables d'employer ce langage pour autre chose que des tâches très simples.

À partir d'Excel 5, le langage macro initial a cessé d'évoluer et a été concurrencé par un autre langage, également intégré à Excel, VBA.

VBA est un langage informatique relativement classique, orienté objet. Depuis Excel 97/98, il comporte un environnement spécifique, avec des outils de déboguage.

Ces différents éléments l'ont rendu beaucoup plus populaires que l'ancien langage. Certains utilisateurs de VBA considèrent que le terme de macro est réducteur et souhaitent même qu'on l'évite.

Qu'est-ce qu'une macro ? À quoi cela peut-il servir ?

Je pourrais m'attarder longuement sur la définition exacte des macros. Pour cela, nous avons divers sites comme www.gaboly.com.
Je me contenterai, ici, de répondre aux seules questions qui sont posées ci-dessus.

- La macro complémentaire, dans Excel, est un code écrit en VBA.
- Elle sert à exécuter automatiquement une tâche dans Excel.

Les macros sont souvent utilisées pour des actions répétitives et longues à faire.

Pour que vous ayez une bonne idée des possibilités offertes, je vais vous faire faire un petit

TP.
Je vous fournis donc le fichier de base et nous travaillerons dessus.

Récupérez ce fichier : **comptes.xls**

Afin de colorer un peu plus le code, j'ai utilisé les **balises-codes** du **VB.NET** et non celles du **VBA**, qui sont inexistantes.
Je précise bien que le VB.NET **n'a rien à voir avec** le VBA.

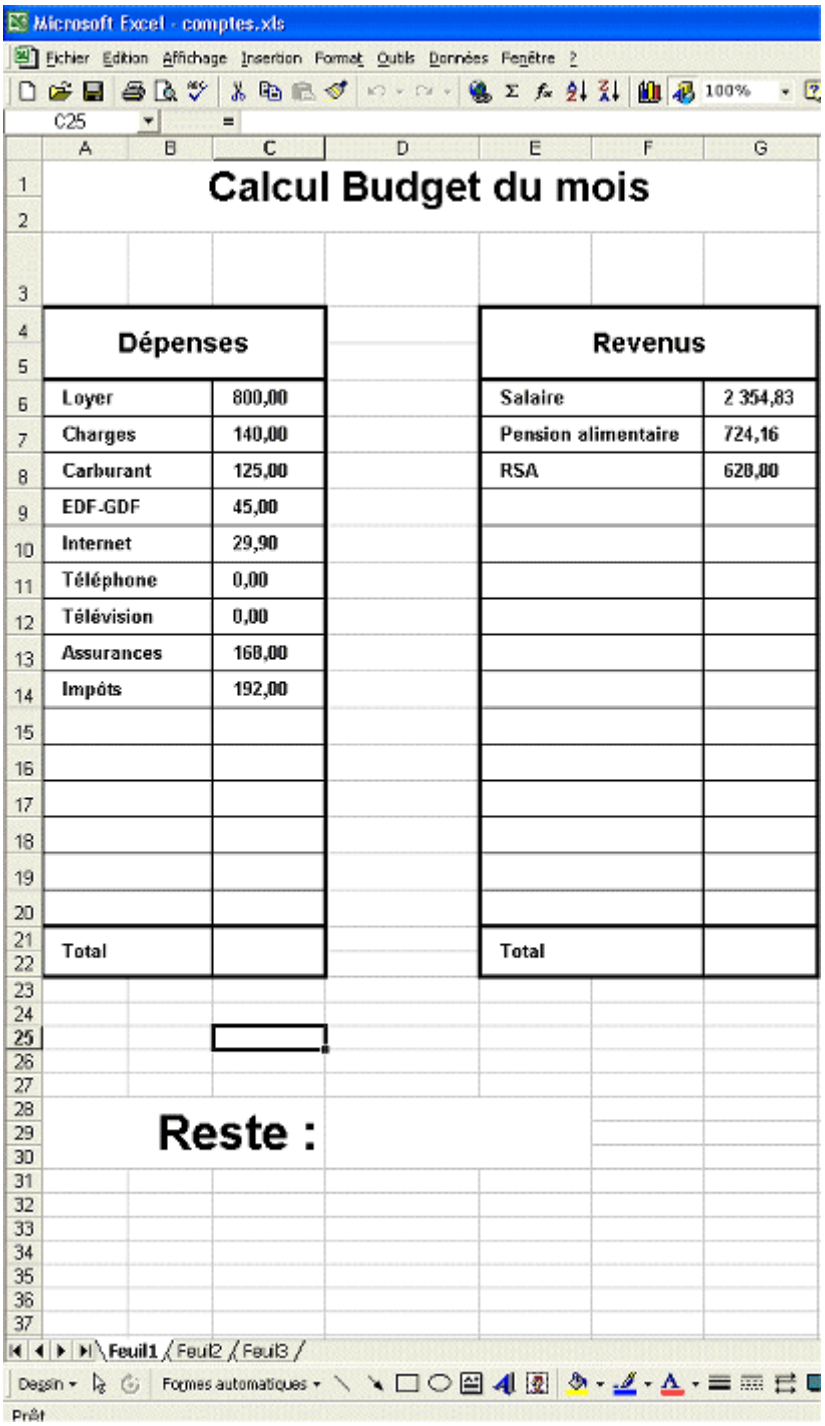
Sommaire du tutoriel :

Retour en haut

L'écriture d'une macro

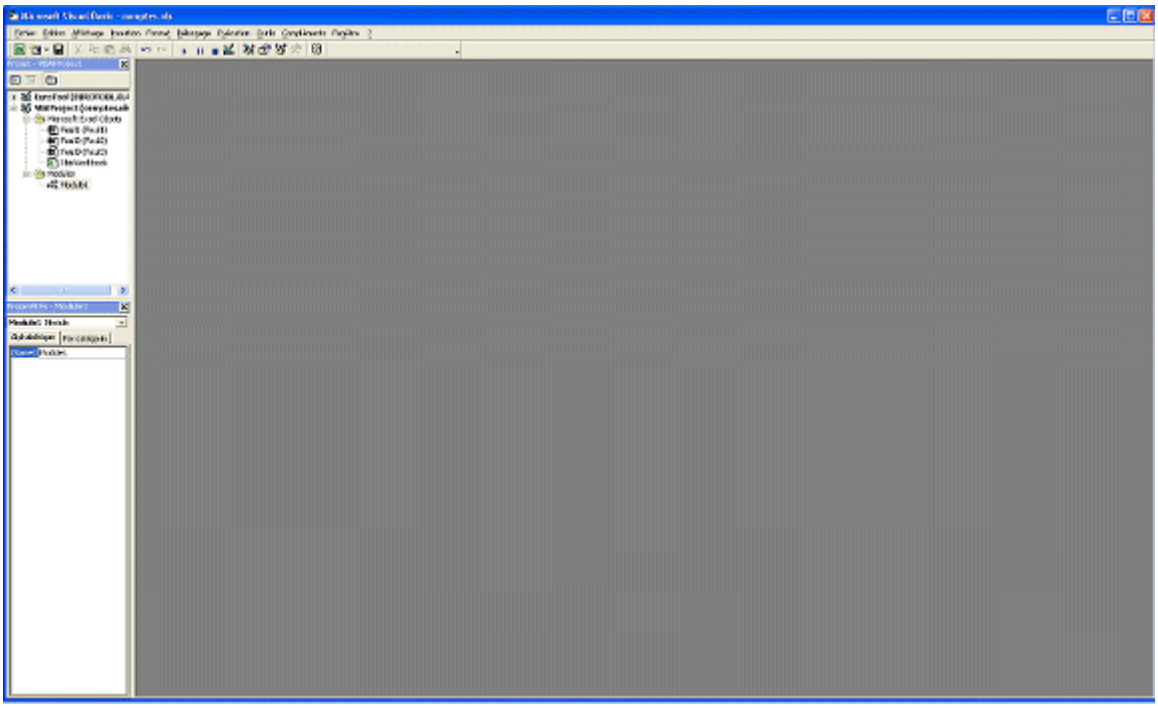
Je vais en premier lieu vous montrer la façon la plus compliquée de créer une macro.
Pour cela, il faut avoir certaines bases en VBA.
Le but de ce tutoriel n'est pas de vous apprendre le Visual Basic. Il s'agit simplement de vous en montrer quelques caractéristiques avant de poursuivre.
Cela vous sera utile au cas où vous seriez amenés à modifier une macro manuellement.

On va commencer par créer une macro qui nous permet de calculer la somme des dépenses.
Pour cela, ouvrez le fichier que vous avez téléchargé.



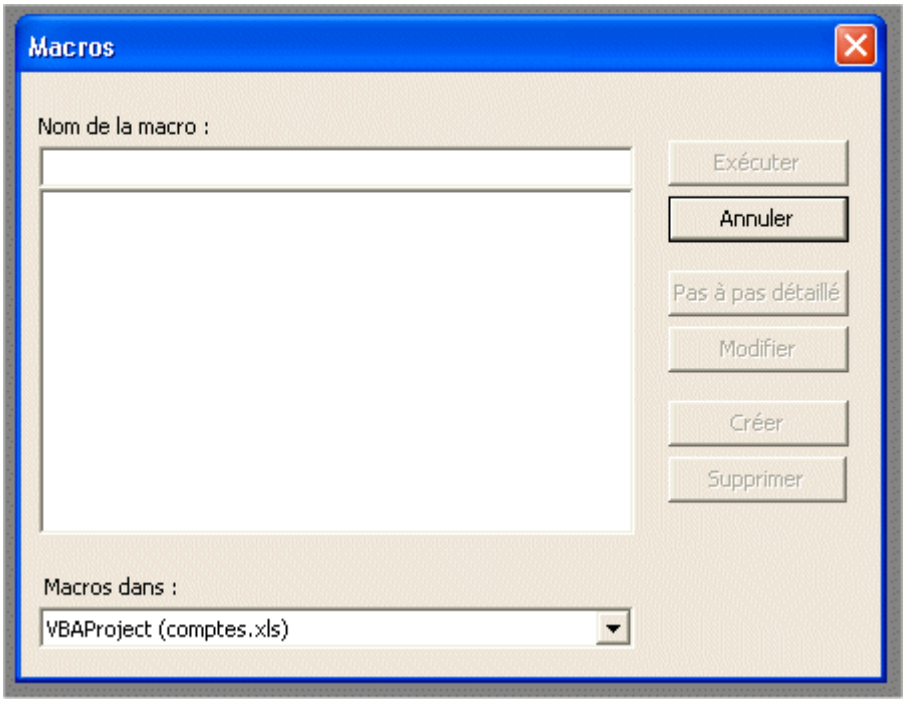
Écriture d'une macro

Maintenant, cliquez sur le menu **Outils** puis sur **Macros** et enfin sur **Visual Basic Editor**. Nous pouvons maintenant créer notre première macro avec cet éditeur.



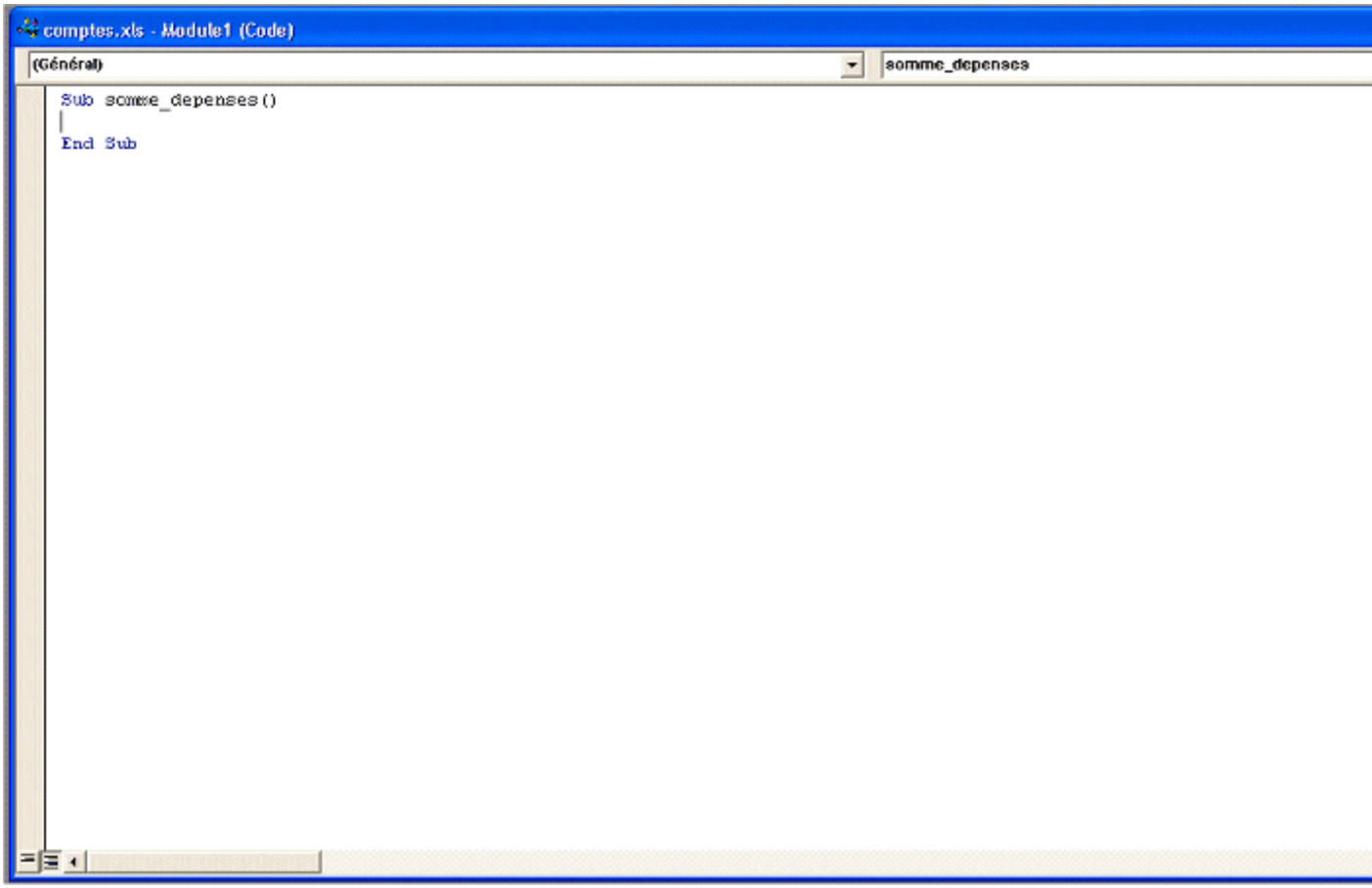
Pour créer notre première macro, vous devez cliquer sur le menu **Outils** puis **Macros**.
Une petite fenêtre apparaît avec toutes les macros du classeur. Bien sûr, celui-ci étant vierge, vous n'en voyez pas.

Dans le champ « **nom de la macro** », mettez « **somme_depenses** ».
Il faut éviter tous les caractères spéciaux, ils sont cause de problèmes.
Donc, pas d'espaces, pas d'accents, pas de « + - / * », etc.



Cliquez ensuite sur **Créer**.

La nouvelle fenêtre qui apparaît nous permet donc de taper notre code en Visual Basic.



La macro commence toujours par **Sub nom de la macro()** et se termine toujours par **End Sub**.
Dans notre cas, nous avons :

Code : VB.NET - Sélectionner

```
1 Sub somme_depenses()
2
3 End Sub
```

Nous allons pouvoir commencer à taper notre code.
Nous devons donc définir la cellule où la somme des dépenses s'affichera.
Il s'agit de la case **C21**.
Celle-ci est fusionnée avec la C22.

Voici comment lui dire que nous sélectionnons la cellule C21 (qui contient aussi la cellule C22) :

Code : VB.NET - Sélectionner

```
1 Range("C21:C22").Select
```

Il nous faut ensuite lui faire calculer la somme des cellules C6 à C20 dans la cellule active.
Là où ça va faire réfléchir, c'est le positionnement dans les formules.

Code : VB.NET - Sélectionner
```
1 ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
```

Humm, hummmm ?
C'est là que ça commence à chauffer derrière les yeux et entre les oreilles.

=Sum() représente la formule **=Somme()** qu'on peut facilement taper dans une cellule du tableau.
R[-15]C:R[-1]C représente la zone de sélection par rapport à la cellule actuelle.

Il faut remonter de 15 pour sélectionner la cellule 6 et remonter de 1 pour sélectionner la cellule 20 par rapport à la cellule active (C21:C22).
C'est en suivant cette logique que nous avons écrit cette ligne.
Les deux points dans la formule définissent la plage comprise entre les deux cellules.

Je vous affiche le code complet de la macro.

Code : VB.NET - Sélectionner
```
1 Sub somme_depenses()
2     Range("C21:C22").Select
3     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
4 End Sub
```

Qu'avons-nous fait exactement ?

Vous allez rire !
Cette macro permet d'écrire **=somme(C6:C20)** dans la cellule C21:C22.

Vous n'êtes pas obligés de fermer l'éditeur pour pouvoir revenir à Excel, « ALT + TAB » est amplement suffisant.

Pour l'exécuter, vous retournez dans Excel et vous cliquez sur **Outils → Macro → Macros**, choisissez la macro puis cliquez sur **Exécuter**.

Là, vous allez me dire que c'est se casser la tête pour pas grand-chose et je ne vous donne pas tort.
Nous allons donc passer ensuite à la méthode la plus simple pour créer une macro.
Il fallait juste que vous sachiez ce qui se passe à l'écriture d'une macro avant de poursuivre ce tutoriel.

Je vous donne quand même un petit supplément à cette macro.

Code : VB.NET - Sélectionner

```
1 Range("G21:G22").Select
2     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
```

Ce qui nous donne donc :

Code : VB.NET - Sélectionner

```
1 Sub somme_depenses()
2
3     Range("C21:C22").Select
4     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
5     Range("G21:G22").Select
6     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
7
8 End Sub
```

Allez sur la cellule **C21:C22** et supprimez son contenu.
Exécutez ensuite la macro et regardez ce qui a changé.

J'espère que ce premier changement vous incite à poursuivre ce tutoriel…
Retour en haut

L'assistant création de macros

Après avoir aperçu l'écriture d'une macro, nous allons passer à l'outil qui nous sera le plus utile.

L'assistant enregistrera chaque chose que l'on fera. Le moindre clic sera inscrit en VBA.

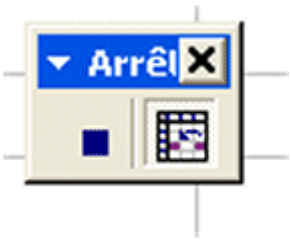
Nous risquons d'augmenter considérablement la taille de la macro en effectuant des tâches inutiles.
Il nous faut faire uniquement le strict minimum.

Enregistrer une macro complémentaire

Cliquez donc sur **Outils → Macro → Nouvelle_macro**.
Vous voyez qu'on peut attribuer un raccourci pour son exécution ou même l'enregistrer ailleurs que dans le classeur actuel.
Nous n'allons pas nous attarder sur ces options et rester sur ce classeur.

Mettez le nom que vous voulez à votre macro (pas de caractères spéciaux) et cliquez sur « OK ».
J'ai mis **test** comme nom de macro et « **essai pour le SDZ** » en description.

Vous voyez la barre d'outils s'afficher.



Elle nous servira à la fin.

C'est à partir de là qu'Excel enregistre tout ce qu'on fait, il nous faut donc éviter les clics superflux.

- Sélectionnez la cellule **D28** ;
- appuyez sur la touche **+=}** pour faire le signe égal ;
- sélectionnez la cellule **G21** ;
- appuyez sur **6-|** pour faire le signe moins ;
- sélectionnez la cellule **C21** ;
- validez avec la touche **Entrée** ;
- sélectionnez, à nouveau, la cellule **D28**.

Lors de l'enregistrement, vous devez avoir en G28 :
Citation : Excel
=G21-C21

Si tout s'est bien passé, nous avons nos deux totaux de renseignés et la somme restante.

Appuyer sur le bouton « Arrêter l'enregistrement » de la barre d'outils macros.

Allez voir l'éditeur, Microsoft Visual Basic, qui contient nos deux macros.

Vous pouvez remarquer une grande différence entre celles-ci.

Code : VB.NET - Sélectionner

```
 1 Sub somme_depenses()
 2
 3     Range("C21:C22").Select
 4     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 5     Range("G21:G22").Select
 6     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 7
 8 End Sub
 9
10 Sub test()
11 '
12 ' test Macro
13 ' essai pour le SDZ
```

```
14 '
15
16 '
17     ActiveCell.Offset(-2, -6).Range("A1:B3").Select
18     ActiveCell.FormulaR1C1 = "=R[-7]C[3]-R[-7]C[-1]"
19     ActiveCell.Range("A1:B3").Select
20 End Sub
```

L'arrangement manuel d'une macro enregistrée

C'est dans des moments comme celui-ci qu'il est possible de simplifier la macro, et c'est ce que nous allons faire en fusionnant les deux précédentes dans une seule et même macro.

Nous gardons donc la première macro :

Code : VB.NET - Sélectionner

```
1 Sub somme_depenses()
2
3     Range("C21:C22").Select
4     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
5     Range("G21:G22").Select
6     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
7
8 End Sub
```

Nous allons l'arranger avec ce qu'on a dans la seconde.

Nous devons garder la formule utilisée.
Code : VB.NET - Sélectionner

```
1 ActiveCell.FormulaR1C1 = "=R[-7]C[3]-R[-7]C[-1]"
```

Mais juste avant celle-ci, nous allons donner la cellule cible.
Code : VB.NET - Sélectionner

```
1 Range("D28:E30").Select
```

Ce qui donne :

Code : VB.NET - Sélectionner

```
1 Range("D28:E30").Select
2     ActiveCell.FormulaR1C1 = "=R[-7]C[3]-R[-7]C[-1]"
```

Notre macro complète sera donc :

Code : VB.NET - Sélectionner

```
 1 Sub somme_depenses()
```

```
 2
 3     Range("C21:C22").Select
 4     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 5     Range("G21:G22").Select
 6     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 7     Range("D28:E30").Select
 8     ActiveCell.FormulaR1C1 = "=R[-7]C[3]-R[-7]C[-1]"
 9
10 End Sub
```

Supprimez tout le reste (l'autre macro).
Il ne doit vous rester que celle-ci.

Cette seule macro permet donc d'inscrire les 3 formules dans les trois cellules.
1 macro pour 3 résultats en un seul coup.
Je vous accorde que de poser une macro de ce genre n'est guère très intéressant, mais passer par là n'est certainement pas une mauvaise chose. Nous allons désormais pouvoir poursuivre avec d'autres macros complémentaires et d'autres façons de les utiliser.

Le nettoyage des cellules

Il n'y a pas de grosses questions à se poser là-dessus, on va juste faire le vide dans toutes les cellules où se trouvent nos chiffres.

Cliquez sur **Outils → Macro → Nouvelle_macro** et nommez celle-ci « **effacer** ».
Cliquez sur **OK**.

- Sélectionnez les cellules de **C6** à **C22** ;
- appuyez sur CTRL ou MAJ (selon la version d'Excel) pour ajouter une sélection ;
- restez appuyés et sélectionnez les cellules de **G6** à **G22** ;
- et la cellule **D28** ;
- appuyer sur la touche **Suppr** de votre clavier.

Arrêtez l'enregistrement de la macro.

Retournez voir l'éditeur.
Ce dernier doit afficher :

Code : VB.NET - Sélectionner

```
 1 Sub somme_depenses()
 2
 3     Range("C21:C22").Select
 4     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 5     Range("G21:G22").Select
 6     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 7     Range("D28:E30").Select
 8     ActiveCell.FormulaR1C1 = "=R[-7]C[3]-R[-7]C[-1]"
 9
10 End Sub
```

```
11 Sub effacer()
12 '
13 ' effacer Macro
14 ' essai pour le SDZ
15 '
16
17 '
18     ActiveCell.Offset(-22, -1).Range("A1:A17,E1:E17,B23:C25").Select
19     ActiveCell.Activate
20     Selection.ClearContents
21 End Sub
```

Oups !

Nous avons commis une erreur.
Nous voulons juste effacer les résultats et non la totalité, ce qui implique d'effacer juste C21, G21 et D28.
Appuyez sur « CTRL + Z » pour annuler cette dernière opération.

Modifions notre macro.

Code : VB.NET - Sélectionner

```
 1 Sub effacer()
 2 '
 3 ' effacer Macro
 4 ' essai pour le SDZ
 5 '
 6
 7 '
 8     ActiveCell.Offset(-22, -1).Range("A1:A17,E1:E17,B23:C25").Select
 9     ActiveCell.Activate
10     Selection.ClearContents
11 End Sub
```

Nous gardons donc l'activation de cellule et l'effacement.
Nous ne devons modifier que la première ligne qui ne cible pas les bonnes cellules.
Mettons donc :

Code : VB.NET - Sélectionner

```
 1 Sub effacer()
 2 '
 3 ' effacer Macro
 4 ' essai pour le SDZ
 5 '
 6
 7 '
 8     Range("C21,G21,D28").Select
 9     ActiveCell.Activate
10     Selection.ClearContents
11 End Sub
```

Et, au passage, on va enlever l'inutile.
Ce qui nous donnera :

Code : VB.NET - Sélectionner

```
1 Sub effacer()
2     Range("C21,G21,D28").Select
3     ActiveCell.Activate
4     Selection.ClearContents
5 End Sub
```

Voilà qui est bien mieux.

Nous pouvons effacer simplement le contenu des trois cellules de résultats à l'aide de cette macro.

Peut-on faire plus simple que d'aller cliquer sur « Outils → Macro → Macros » et de sélectionner la macro « supprimer » puis de cliquer sur « OK » ?

Vous vous dites : « je vais devoir supprimer les chiffres au moins 200 fois dans ma journée car je veux calculer le budget de chacun de mes clients et je dois avouer qu'un simple clic ça m'aiderait bien ».

Je vais alors vous répondre qu'on va arranger ça.
Ce n'est pas très compliqué.
Retour en haut

Lier une macro à un objet

Pour aider à la tâche, il serait bien de pouvoir lier cette macro à un objet comme un bouton ou une image.
Excel nous offre quelques possibilités sur ce point-là, que nous allons aborder dans cette partie du tutoriel.

Comment exécuter la macro ?

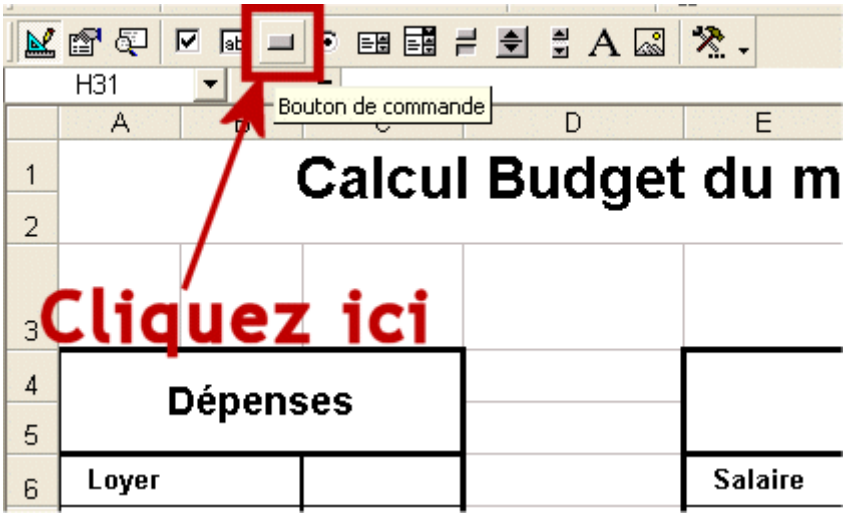
Il existe plusieurs façons d'exécuter celle-ci.
Je vais en énumérer 3 :

- Un bouton Excel ;
- une zone de texte ;
- une image.

Un bouton Excel

Pour créer un bouton Excel, il nous faut la barre d'outils **Boîte à outils Contrôles**.

Cliquez donc sur **Affichage** puis **Barre d'outils** et cochez **Boîte à outils Contrôles**.
Cliquez sur l'icône **Bouton de commande**.



Cliquez sur votre classeur, restez appuyés sur le bouton droit de la souris et tracez un rectangle.
Il sera votre bouton. Relâchez quand la taille vous semble correcte.

Vous pouvez modifier, par la suite, sa taille, la police, etc.

Faites un clic droit sur le bouton et choisissez **Objet Bouton de commande → Édition**.
Vous pouvez changer le nom du bouton. J'ai mis « **EFFACER** ».

Un petit rappel de nos deux macros :

Code : VB.NET - Sélectionner

```
 1 Sub somme_depenses()
 2
 3     Range("C21:C22").Select
 4     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 5     Range("G21:G22").Select
 6     ActiveCell.FormulaR1C1 = "=SUM(R[-15]C:R[-1]C)"
 7     Range("D28:E30").Select
 8     ActiveCell.FormulaR1C1 = "=R[-7]C[3]-R[-7]C[-1]"
 9
10 End Sub
11
12 Sub effacer()
13
14     Range("C21,G21,D28").Select
15     ActiveCell.Activate
16     Selection.ClearContents
17
```

```
18 End Sub
```

Faites un clic droit sur le bouton et choisissez **Visualiser le code**.
Mettez-y le code de la macro « effacer » mais ne touchez pas aux lignes **Sub**.

Ce qui donnera :

Code : VB.NET - Sélectionner

```
1 Private Sub CommandButton1_Click()
2
3     Range("C21,G21,D28").Select
4     ActiveCell.Activate
5     Selection.ClearContents
6
7 End Sub
```

Un simple clic sur le bouton suffira à effacer le contenu des 3 cellules.
Les boutons Excel sont assez pénibles à embellir et à gérer. Il faut faire de multiples manipulations pour arriver à un résultat peu attrayant.

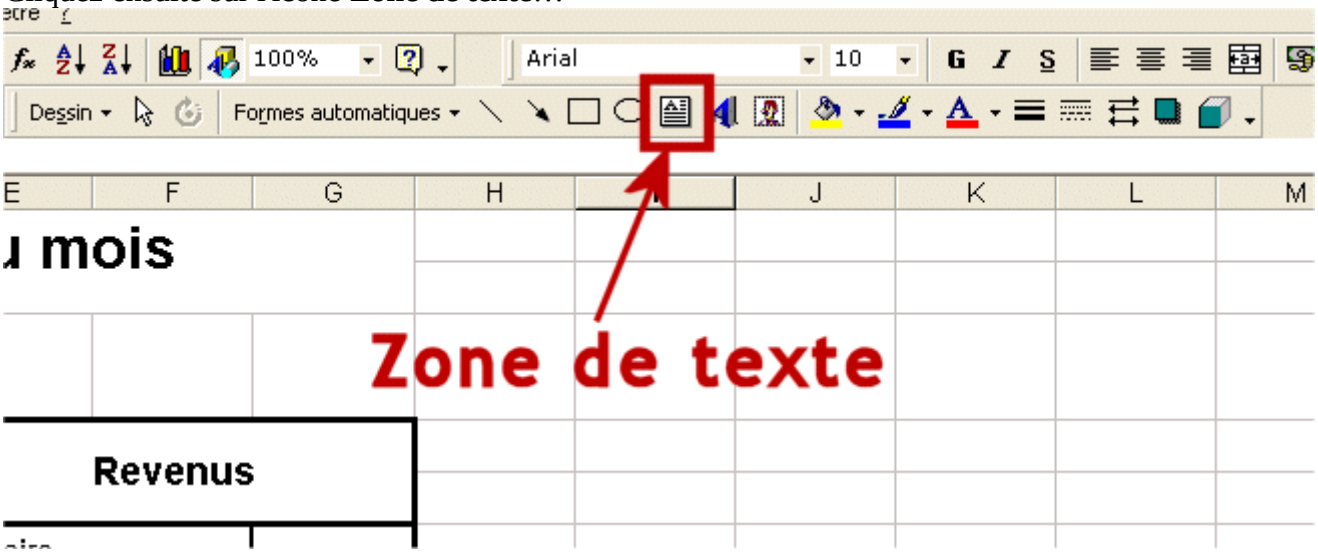
Je vous conseille de n'utiliser cette méthode qu'à titre d'exercice. Il vaut mieux l'oublier au plus vite.

La zone de texte

On peut utiliser une zone de texte aussi.
Là, c'est du simple et rapide.

La barre d'outils « dessin » doit être affichée.
Si ce n'est pas le cas, cliquez sur **Affichage → Barre d'outils → Dessin**.

Cliquez ensuite sur l'icône **Zone de texte**…



… et créez un rectangle comme pour le bouton Excel.

Je vous laisse gérer votre zone de texte comme vous le voulez (clic droit sur le bord puis **Format de la zone de texte**).
Vous pouvez modifier la couleur du fond, la taille du bord,…

Une fois que vous avez fait cela, vous faites un clic droit sur le bord et vous cliquez sur **Affecter une macro**.
Choisissez la macro « **effacer** » et validez ce choix.
Cliquez en dehors de votre zone de texte pour que la macro ne soit plus sélectionnée.
Maintenant, si vous cliquez dessus, la macro sera exécutée.

Une image

J'ai créé une petite image pour l'occasion.

Récupérez celle-ci à l'aide d'un clic droit et choisissez « Enregistrer l'image sous… ».

Cliquez ensuite sur le menu **Insertion → Image → À partir du fichier** et sélectionnez celle-ci.
Ajustez-la si besoin est.

Faites ensuite un clic droit dessus et choisissez **Affecter une macro**. Sélectionnez la macro « effacer » comme précédemment.

Vous pouvez essayer de vous-mêmes d'affecter la macro « **somme_depenses** » à ce bouton.

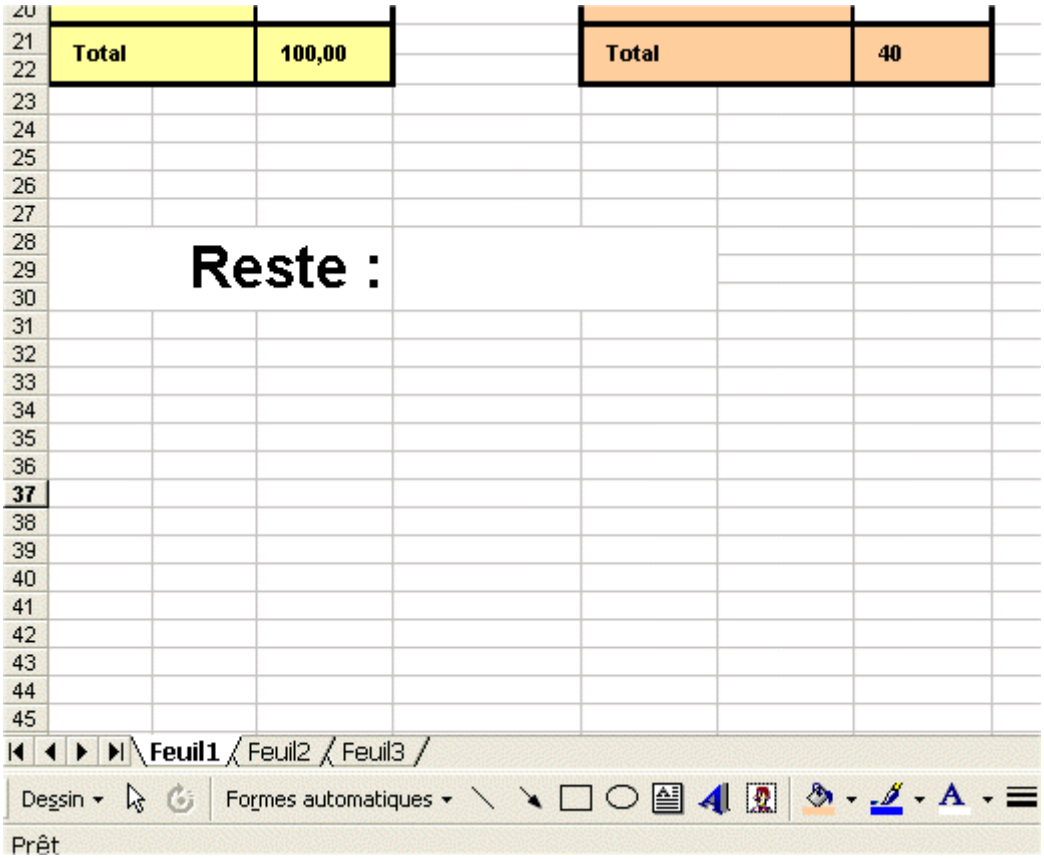
Si vous vous débrouillez bien avec un logiciel de dessin ou de retouche photos, vous pourrez vous faire de beaux boutons.
Affecter une macro à une image, un bouton ou une zone de texte peut permettre de naviguer entre plusieurs feuilles par de simples clics.
On peut, par exemple, faire une page de garde pour un fichier assez volumineux et naviguer à partir de là entre les onglets du classeur.

Et si on le faisait ?

On peut par exemple aller voir les graphiques liés aux tableaux.

Pour cela, cliquez sur le menu **Outils → Macro → Nouvelle Macro…**.
Nommez celle-ci **Graphs**.

Cliquez ensuite sur l'onglet **Feuil2** en bas d'Excel, à gauche.



Arrêtez l'enregistrement.

Cliquez maintenant sur le menu **Outils → Macro → Nouvelle Macro…** et nommez cette dernière **Accueil**.

Cliquez sur l'onglet **Feuil1** et arrêtez l'enregistrement.

Récupérez cette image et placez-la sur la feuille 1 de votre classeur.

Affectez-lui la macro **Graphs**.

Cliquez ensuite dessus et sur le bouton que j'ai déjà placé, et affectez-lui la macro **Accueil**.

Sur la première page, vous devez avoir 2 boutons (« Effacer » et « Graphs ») et sur la feuille 2, vous devez avoir 1 bouton « retour » permettant de revenir très vite au premier onglet.
Retour en haut

Quelques exemples de macros

Nous pouvons avoir un fichier à imprimer dans plusieurs formats différents.

Admettons que nous devions imprimer une affiche en format A4 et en format A6 (Bristol).
Nous pouvons créer 2 macros, chacune affectée à un bouton.

Ainsi, d'un simple clic, nous imprimons soit en A4 soit en A6.
Au lieu de modifier à chaque fois le format, nous cliquons sur 1 seul bouton que nous avons rattaché à une macro.

Voici une mise en page A6 :

Code : VB.NET - Sélectionner

```
Sub format_A6()

    With ActiveSheet.PageSetup
        .PrintTitleRows = ""
        .PrintTitleColumns = ""
    End With
    ActiveSheet.PageSetup.PrintArea = ""
    With ActiveSheet.PageSetup
        .LeftHeader = ""
        .CenterHeader = ""
        .RightHeader = ""
        .LeftFooter = ""
        .CenterFooter = ""
        .RightFooter = ""
        .LeftMargin = Application.InchesToPoints(0)
        .RightMargin = Application.InchesToPoints(0)
        .TopMargin = Application.InchesToPoints(0.78740157480315)
        .BottomMargin = Application.InchesToPoints(0)
        .HeaderMargin = Application.InchesToPoints(0)
        .FooterMargin = Application.InchesToPoints(0)
        .PrintHeadings = False
        .PrintGridlines = False
        .PrintComments = xlPrintNoComments
        .PrintQuality = 600
        .CenterHorizontally = True
        .CenterVertically = False
        .Orientation = xlPortrait
        .Draft = False
        .PaperSize = 266
        .FirstPageNumber = xlAutomatic
        .Order = xlDownThenOver
        .BlackAndWhite = False
        .Zoom = False
        .FitToPagesWide = 1
        .FitToPagesTall = 1
    End With
End Sub
```

Et voilà un format A4 :

Code : VB.NET - Sélectionner

```
Sub format_A4()

    With ActiveSheet.PageSetup
        .PrintTitleRows = ""
        .PrintTitleColumns = ""
    End With
    ActiveSheet.PageSetup.PrintArea = ""
    With ActiveSheet.PageSetup
        .LeftHeader = ""
        .CenterHeader = ""
        .RightHeader = ""
        .LeftFooter = ""
        .CenterFooter = ""
        .RightFooter = ""
        .LeftMargin = Application.InchesToPoints(0)
        .RightMargin = Application.InchesToPoints(0)
        .TopMargin = Application.InchesToPoints(0.78740157480315)
        .BottomMargin = Application.InchesToPoints(0)
        .HeaderMargin = Application.InchesToPoints(0)
        .FooterMargin = Application.InchesToPoints(0)
        .PrintHeadings = False
        .PrintGridlines = False
        .PrintComments = xlPrintNoComments
        .PrintQuality = 600
        .CenterHorizontally = True
        .CenterVertically = False
        .Orientation = xlPortrait
        .Draft = False
        .PaperSize = xlPaperA4
        .FirstPageNumber = xlAutomatic
        .Order = xlDownThenOver
        .BlackAndWhite = False
        .Zoom = False
        .FitToPagesWide = 1
        .FitToPagesTall = 1
    End With
End Sub
```

Retour en haut

Ce ne sont pas les possibilités qui manquent.
Il faut juste arriver à comprendre le VBA et arriver à ajuster les macros le plus simplement possible.
Croyez-moi, quand on a une centaine de macros, on cherche à les réduire au maximum.

- Les macros ne doivent être créées que pour exécuter des actions répétitives et pénibles.
- Ne chargez pas trop vos fichiers avec des tonnes d'images, ils ont tendance à devenir très gros après.

J'espère que ce tuto aura aidé les plus novices d'entre vous à comprendre comment se présente une macro complémentaire, à quoi elle peut servir.
Ce tutoriel n'est qu'une approche des macros. Si vous voulez poursuivre après ça, je ne peux que vous suggérer de vous orienter sur des sites spécialisés en Visual Basic et dans Excel.

Insérer des données d'un fichier Excel ou Access dans MySQL

- Web
- Livre
- eBook
- PDF
- Ajouter à mes favoris

Par Desolation

Mise à jour : 17/02/2012

Difficulté : Facile Durée d'étude : 30 minutes

Bonjour à tous !

Vous possédez un grand nombre de données contenues sur des supports tels que des fichiers issus d'Access ou d'Excel et souhaitez les insérer dans une base de données MySQL sans avoir à tout recopier ? Vous êtes sur la bonne page !

Ce tutoriel a pour vocation d'aider un maximum de personnes qui rencontrent cette difficulté. Ayant moi-même été confronté à cette situation, j'ai pour vous plusieurs solutions :

1. avec un pilote ODBC, la plus complexe ;
2. avec le format CSV, la plus rapide.

Afin de mener à bien nos manipulations, vous devez disposer d'un certain nombre d'éléments :

- un fichier source *.xls de test (télécharger), que je vous fournis (il a été fait sous Excel 2010 mais enregistré pour une compatibilité Excel 2003 — notez que vous avez tout de même la possibilité, si vous êtes utilisateur de la version 2003, de lire des fichiers issus des versions 2007 et supérieures en téléchargeant un petit utilitaire fourni par Microsoft) ;
- la suite Microsoft Office (2003 ou supérieure), principalement Access et Excel : j'utiliserai la version 2010 ;
- la plateforme WAMP (ou similaire), que nous emploierons seulement pour le moteur MySQL : j'utiliserai WAMP Server 2.2 avec MySQL 5.5.16.

Maintenant que vous avez tous les outils nécessaires, concentrons-nous sur la mise en œuvre.
Retour en haut

Sommaire du tutoriel :

Retour en haut

Solution n° 1 : avec un pilote ODBC

Préparation

Pour cette solution, il est nécessaire de passer par ces quelques étapes de préparation :

1. analyser la structure du fichier ;
2. mettre en place la base de données et la table qui recevra les données du côté de MySQL ;
3. installer le pilote ODBC pour faire communiquer Access et MySQL ;
4. créer la source de données.

Analyse du fichier source

Nous allons ici ouvrir le fichier qui contient les données, et ainsi prendre connaissance du nombre de champs (colonnes) et des types de valeurs que nous souhaiterions obtenir en sortie. Regardons notre exemple, fraîchement téléchargé et ouvert :

	A	B
1	1	Jean
2	2	Paul
3	3	Bernard
4	4	Richard
5	5	Hervé
6	6	Joseph
7	7	André
8	8	Maxime
9	9	Olivier
10	10	Bertrand
11	11	Guillaume
12	12	Henry
13	13	Marc-Antoine
14	14	Claude
15	15	François
16	16	Didier
17	17	Michel
18	18	Patrick

Dans la colonne **A**, nous avons des nombres (certes, pas stockés sous un type numérique dans notre champ, mais ce n'est pas un problème) et, en colonne **B**, des prénoms.
Cette situation est plus que classique : on peut déduire qu'à un identifiant unique correspond un prénom (certes, lui, pas forcément unique).

Mise en place de la base de données de destination

Nous avons donc deux champs. Le premier se nommera **id** et sera de type INT, le second se nommera **prenom** et sera de type VARCHAR(50).

Voici le code SQL permettant de créer la base de données tuto et d'ajouter la table personnes contenant les deux champs cités précédemment.
Code : SQL - Sélectionner

```
1 CREATE DATABASE tuto;
2 USE tuto;
3
4 CREATE TABLE personnes (
5 id int(11) NOT NULL AUTO_INCREMENT,
6 prenom varchar(50) NOT NULL,
7 PRIMARY KEY (id)
8 );
```

Notre objectif est maintenant d'insérer nos données dans cette table !

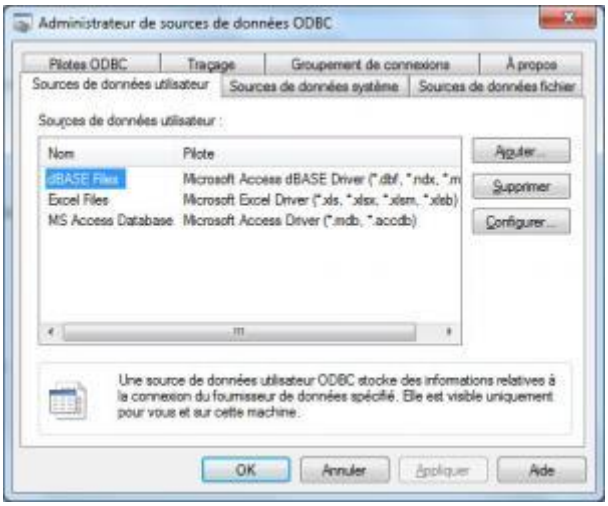
Installation du pilote

Pour cette solution, vous devez installer un pilote ODBC pour MySQL. Téléchargez la version 32 bits ou 64 bits (cliquez sur « No thanks, just take me to the downloads! » si vous ne souhaitez pas créer de compte) selon votre configuration Windows et installez. J'utiliserai cette version qui est la 5.1.10 (64 bits).

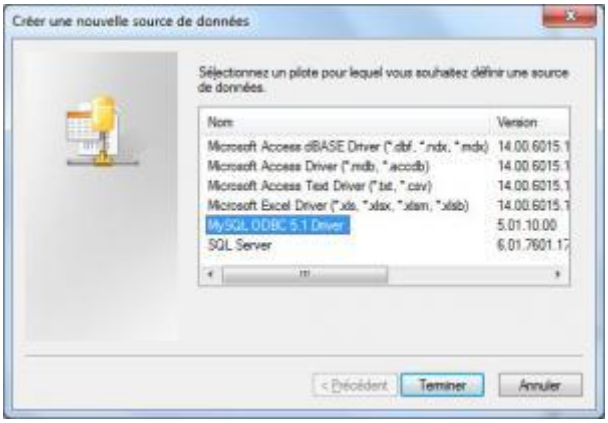
Création de la source de données

Pour qu'Access puisse communiquer avec MySQL, il a besoin d'un pilote mais aussi d'une source de données, habituellement appelée **DSN** (Data Source Name) ; c'est elle qui va indiquer sur quel serveur et quelle base se connecter.

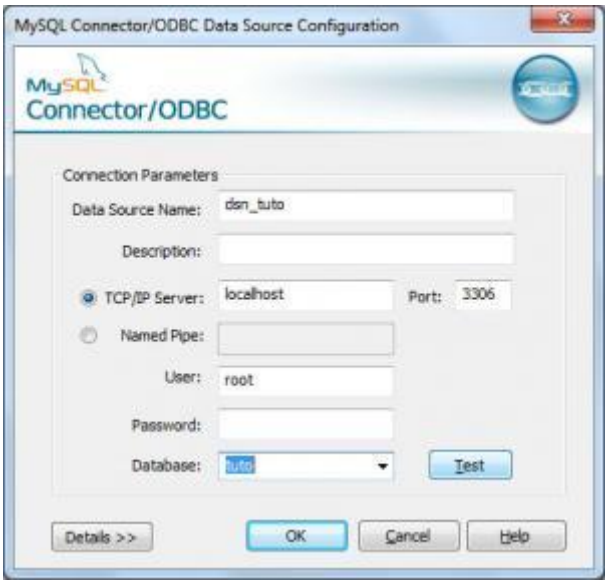
Lancez votre serveur MySQL et vérifiez qu'il est bien démarré. Rendez-vous dans **Panneau de configuration > Système et sécurité > Outils d'administration > Sources de données (ODBC)**. L'écran suivant s'affiche alors :



Dans « Source de données utilisateur », sélectionnez « Ajouter », après quoi vous obtenez ceci :



Vous avez la liste de tous les pilotes utiles pour les bases de données installés sur votre machine, sélectionnez donc « MySQL ODBC 5.1 Driver », le pilote que vous venez d'installer, et faites « Terminer ».



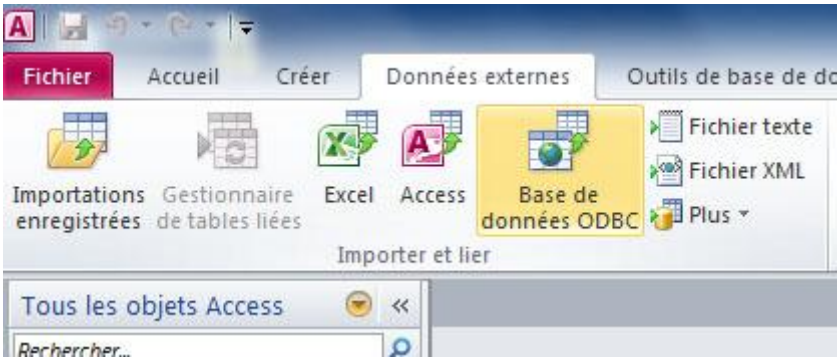
Configurons à présent notre connexion à MySQL. Il nous faut tout d'abord donner un nom à notre DSN, choisissez donc un nom qui vous permettra de la retrouver facilement par la suite. Indiquez ensuite le nom du serveur (localhost), le nom d'utilisateur (par défaut, root sans mot de passe) puis sélectionnez dans la liste la base de données concernée (soit tuto). Cliquez ensuite sur « Test » pour tester la connexion. Vous devriez obtenir « Connection successful ». Cliquez sur « OK ».

Notre DSN est désormais créée : nous pouvons ouvrir Access !

Connexion à MySQL depuis Access

Créez un nouveau fichier Access. Étant sous 2010, j'obtiens un fichier *.accdb — sous 2003, vous devriez obtenir un *.mdb. Ouvrez le fichier si ce n'est déjà fait.

Si vous utilisez Access 2007, cherchez Données externes > Plus > Base de données ODBC.
Sous Access 2010, cherchez Données externes > Base de données ODBC.



Mais qu'est-ce que ce « ODBC » ? Tu nous as fait installer un truc dont on ne connaît même pas l'utilité ?!

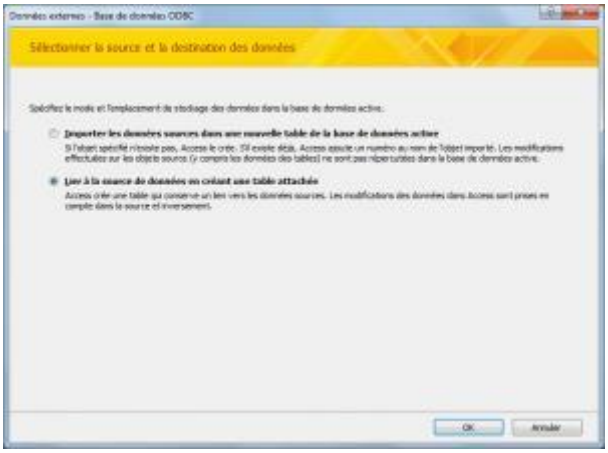
Citation : Comment Ça Marche

ODBC signifie Open DataBase Connectivity. Il s'agit d'un format défini par Microsoft permettant la communication entre des clients bases de données fonctionnant sous Windows et les SGBD du marché.

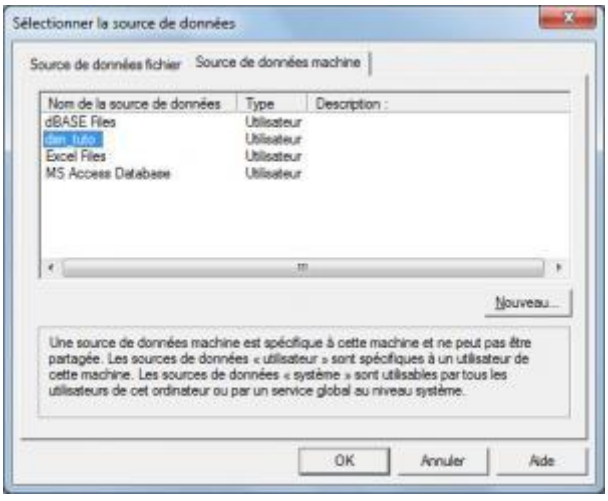
Le gestionnaire ODBC est présent sur les systèmes Windows. Il existe toutefois des implémentations sur d'autres plates-formes, notamment des plates-formes UNIX/Linux.

Voilà pour la définition. Seulement, le pilote pour MySQL n'est pas natif à Windows, c'est pourquoi je vous l'ai fait télécharger.

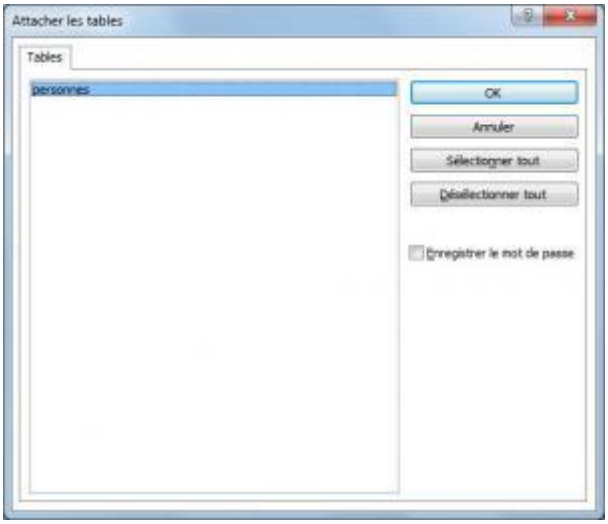
Vous devriez obtenir cette fenêtre. Sélectionnez le second choix car nous souhaitons nous synchroniser avec notre table personnes (pour ceux qui suivent).



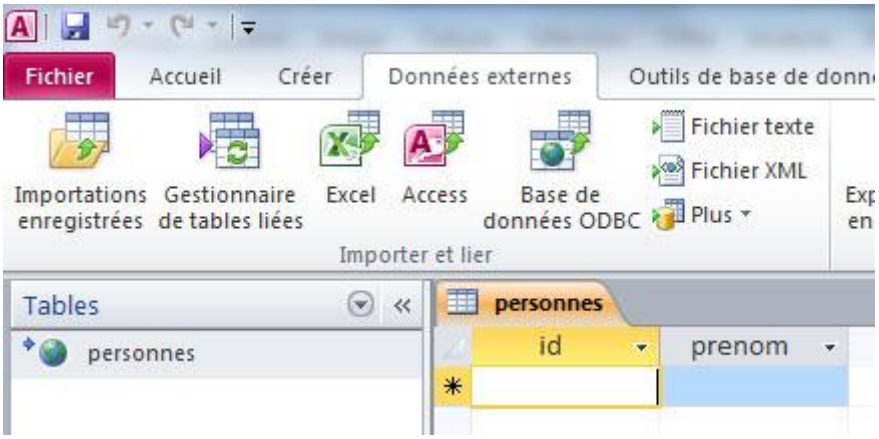
Ensuite, nous allons retrouver notre DSN dans « Source de données machine ».



Sont répertoriées les tables appartenant à la base de données sélectionnée (d'où l'intérêt de construire sa table auparavant), sélectionnez donc la table personnes et cliquez sur « OK ».



Sur la liste, à gauche, apparaît l'entrée personnes : double-cliquez dessus pour l'ouvrir ; nous retrouvons bien nos champs **id** et **prenom**.

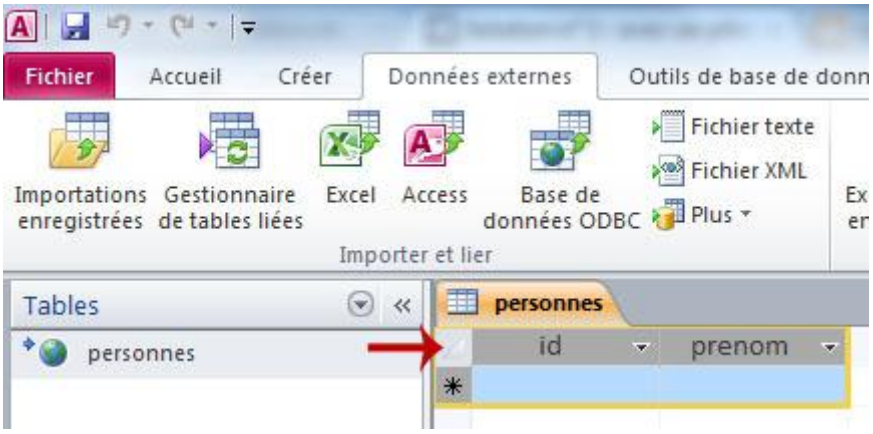


Retournons à présent sur notre fichier Excel (ne fermez pas le fichier Access) — rouvrez-le si vous l'aviez fermé. Il va maintenant s'agir de sélectionner toutes nos données par une combinaison que vous connaissez bien.

Petite astuce : placez-vous en cellule **A1** et appuyez sur les touches suivantes : Ctrl + Shift + Fin. Ainsi, toutes les données placées entre la cellule de départ en haut à gauche (ici **A1**) et la cellule en bas à droite seront sélectionnées jusqu'à ce qu'Excel ne trouve plus de données (dans notre cas, en **B18**). Vous pouvez maintenant copier (Ctrl + C).

	A	B
1	1	Jean
2	2	Paul
3	3	Bernard
4	4	Richard
5	5	Hervé
6	6	Joseph
7	7	André
8	8	Maxime
9	9	Olivier
10	10	Bertrand
11	11	Guillaume
12	12	Henry
13	13	Marc-Antoine
14	14	Claude
15	15	François
16	16	Didier
17	17	Michel
18	18	Patrick
19		

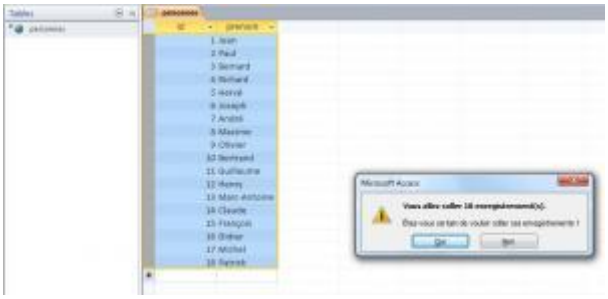
Retournez sur votre fichier Access puis sélectionnez l'ensemble des données en cliquant sur la petite flèche qui se situe à gauche de **id** et au-dessus de l'étoile ***** (c'est très important, sinon la copie ne fonctionnera pas) :



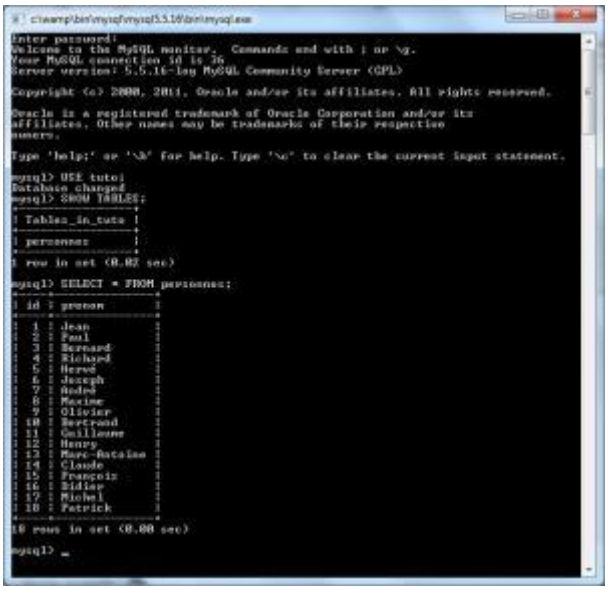
Il ne vous reste plus qu'à coller les données avec Ctrl + V.

Si vous êtes sous Windows XP, vous aurez peut-être cette erreur. Si tel est le cas, reportez-vous au chapitre « **Résolution du problème** ».

Reprenons. Vous devriez maintenant obtenir ceci :



Bien sûr, répondez « Oui ». C'est terminé, vos données sont insérées ! Allez quand même vérifier via phpMyAdmin si vous en doutez. Ou via la console MySQL (pour le fun !).



Nous retrouvons donc nos enregistrements : mission accomplie !

Retour en haut

Résolution du problème

J'ai rencontré ce problème sous Windows XP et avec Office 2007.

Première solution

Allez dans Panneau de configuration > Ajout/Suppression de programmes > cochez « Afficher les mises à jour » > sélectionnez la mise à jour pour Excel 2007 **KB958437** et supprimez-la. Copiez à nouveau la sélection dans Excel et collez dans Access. Si le problème persiste ou si vous ne trouvez pas cette mise à jour dans la liste, passez à la seconde solution.

Seconde solution

Ouvrez un document Word vierge, collez avec Ctrl + V, sélectionnez à nouveau avec Ctrl + A, copiez avec Ctrl + C, collez dans Access avec Ctrl + V et ça devrait fonctionner.

Retour en haut

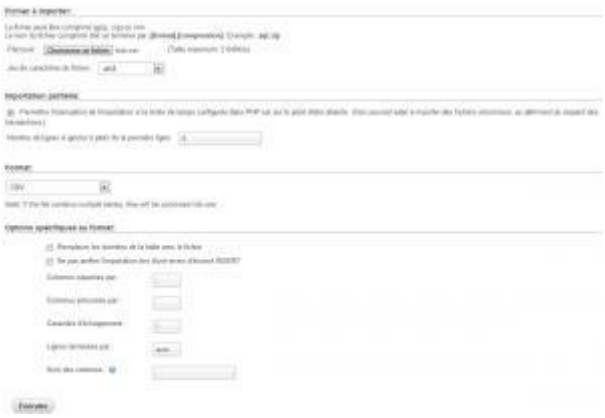
Solution n° 2 : avec le format CSV

Cette solution est la plus simple. Il faut au préalable convertir le fichier tuto**.xls** en tuto**.csv**.

Il faut pour cela, dans Excel, faire un « Enregistrer sous… » et choisir le type approprié :

En mode graphique

phpMyAdmin, pour ceux qui en disposent, propose un import de données par fichier dont le poids n'excède pas 2 Mo.
Choisissez votre base de données puis sélectionnez l'option « Importer » et remplissez le formulaire comme suit :



Pour que les accents soient correctement enregistrés dans la table de destination, il faut convertir le fichier ***.csv** en UTF-8. Pour cela, vous pouvez l'ouvrir avec Notepad++ et utiliser le menu **Encodage > Convertir en UTF-8**.

phpMyAdmin vous a créé une table et des colonnes avec des noms génériques.

En mode console

Par ce biais, vous n'aurez (normalement) pas de contrainte concernant le poids du fichier.
Il faut d'abord créer la base de données et la table, tout comme dans la solution n° 1. Voici le code pour créer la base et la table :

Code : SQL - Sélectionner

```
1 CREATE DATABASE tuto;
2 USE tuto;
3
4 CREATE TABLE personnes (
5 id int(11) NOT NULL AUTO_INCREMENT,
6 prenom varchar(50) NOT NULL,
7 PRIMARY KEY (id)
8 );
```

Ensuite, il faut utiliser cette commande pour intégrer le ***.csv**.

Que ce soit sous UNIX/Linux ou Windows, le séparateur de répertoire reste le slash (/).

Code : SQL - Sélectionner

```
1 LOAD DATA LOCAL INFILE 'C:/User/Syl/Desktop/tuto.csv'
2 INTO TABLE tuto.personnes
3 FIELDS TERMINATED BY ';'
4 ENCLOSED BY ''
```

```
5 LINES TERMINATED BY '\r\n';
```

Voilà, les données sont insérées, vous pouvez vérifier comme je l'ai indiqué en fin de solution n° 1.