Tutoriel ACCESS :

Bien commencer une base de données ACCESS

* Commencer proprement une nouvelle base ACCESS.
* Faciliter ultérieurement développements et maintenance.

Table des matières

[1 Introduction 2](#_Toc138660930)

[1.1 Qu’est-ce qu’ACCESS ? 2](#_Toc138660931)

[1.2 Quel est le public cible ? 2](#_Toc138660932)

[1.3 SGBDR et modèle relationnel 2](#_Toc138660933)

[1.4 Exemple traité dans le tutoriel ACCESS 2](#_Toc138660934)

[2 Comment créer une table ? 2](#_Toc138660935)

[2.1 Réfléchir très précisément aux caractéristiques des champs à créer 2](#_Toc138660936)

[2.1.1 Liste des tables à créer 2](#_Toc138660937)

[2.1.2 Liste des champs à créer 2](#_Toc138660938)

[2.2 Créer les champs dans la base 3](#_Toc138660939)

[2.3 Créer les liens entre les tables 3](#_Toc138660940)

[3 Utilisation des Tables et Champs créés 4](#_Toc138660941)

[3.1 Utilisation des champs créés dans une requête 4](#_Toc138660942)

[3.2 Utilisation des champs créés dans un formulaire ou un état 4](#_Toc138660943)

[4 Conventions 5](#_Toc138660944)

[4.1 Conventions de noms des tables 5](#_Toc138660945)

[4.2 Conventions de noms des champs 5](#_Toc138660946)

[4.3 Conventions de noms des requêtes 5](#_Toc138660947)

[4.4 Conventions de noms des formulaires et états 6](#_Toc138660948)

[4.5 Conventions de développement des modules 6](#_Toc138660949)

# Introduction

## Qu’est-ce qu’ACCESS ?

*« MS-Access est un système de gestion de base de données relationnelles développé par Microsoft. Il fournit une interface conviviale pour créer et gérer des bases de données, permettant aux utilisateurs de stocker, organiser et manipuler efficacement les données de manière graphique. »* - ChatGPT.

## Quel est le public cible ?

* Les utilisateurs d’EXCEL qui souhaiteraient par exemple faire des calculs complexes sur plusieurs feuilles en même temps, ou visualiser les données de plusieurs sources dans un même formulaire.
* Les utilisateurs d’ACCESS qui aimeraient améliorer la rigueur avec laquelle ils créent leurs bases de données, faciliter les développements ultérieurs, et même les rendre auto-documentées.

## SGBDR et modèle relationnel

ACCESS est un SGBDR : Système de Gestion de [Base de Données Relationnelles](https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_relationnelle).

Il semble donc intéressant de créer des [Relations](https://fr.wikipedia.org/wiki/Mod%C3%A8le_relationnel) entre les tables envisagées dans une base de données.

A défaut de maîtriser les [formes normales](https://fr.wikipedia.org/wiki/Forme_normale_(bases_de_donn%C3%A9es_relationnelles)) en modélisation, quelques principes de base :

* Eviter (sauf exceptions mûrement réfléchies) les informations redondantes.
* Créer une clé primaire pour chaque table, ensuite utilisée comme clé étrangère dans les tables liées.

## Exemple traité dans le tutoriel ACCESS



# Comment créer une table ?

## Réfléchir très précisément aux caractéristiques des champs à créer

Une bonne approche est d’établir au début de tout Projet, un Dictionnaire des Données communs avec les Métiers. Le temps passé sur ce travail permettra de gagner 20 fois plus de temps par la suite.

### Liste des tables à créer

Indiquer un descriptif fonctionnel court et précis de chaque table :

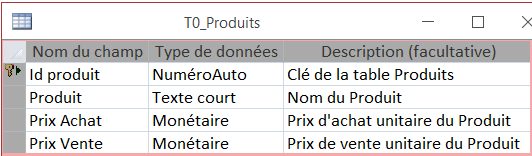
|  |  |
| --- | --- |
| Table | Description |
| T0\_Produits | Table des Produits vendus |
| T0\_Vendeurs | Table des Vendeurs |
| T0\_Factures | Tables des Factures |
| T0\_Ventes | Tables des Produits vendus |

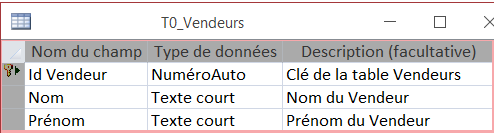
### Liste des champs à créer

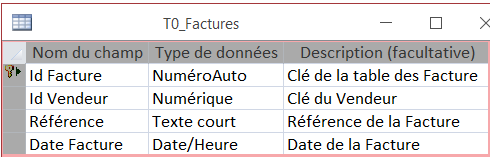
Pour chaque champ, indiquer un descriptif fonctionnel court et précis, ainsi que les caractéristiques techniques de chaque champ. Il est intéressant de maîtriser l’utilité de chaque colonne du tableau suivant :

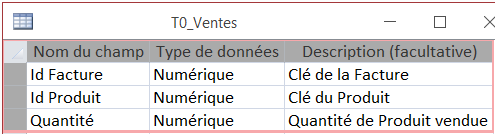
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nom Table** | **Nom Champ** | **Ordre** | **Obligatoire** | **Type** | **Longueur** | **Valeur par défaut** | **Valeurs possibles** | **Description** | **Format** | **Majuscules** |
| T0\_Facture | Id Facture | 10 | Oui | Entier |  |  |  | Clé de la table des Facture |  |  |
| T0\_Facture | Id Vendeur | 20 | Oui | Entier |  |  |  | Clé du Vendeur |  |  |
| T0\_Facture | Référence | 30 | Oui | Texte | 50 |  |  | Référence de la Facture | Lettres uniquement | Oui |
| T0\_Facture | Date Facture | 40 | Oui | Date |  | Aujourd'hui |  | Date de la Facture |  |  |
| T0\_Produits | Id produit | 10 | Oui | Entier |  |  |  | Clé de la table Produits |  |  |
| T0\_Produits | Produit | 20 | Oui | Texte | 50 |  |  | Nom du Produit |  |  |
| T0\_Produits | Prix Achat | 30 | Oui | Monétaire |  |  |  | Prix d'achat unitaire du Produit |  |  |
| T0\_Produits | Prix Vente | 40 | Oui | Monétaire |  |  |  | Prix de vente unitaire du Produit |  |  |
| T0\_Vendeurs | Id Vendeur | 10 | Oui | Entier |  |  |  | Clé de la table Vendeurs |  |  |
| T0\_Vendeurs | Nom | 20 | Oui | Texte | 50 |  |  | Nom du Vendeur |  |  |
| T0\_Vendeurs | Prénom | 30 | Oui | Texte | 50 |  |  | Prénom du Vendeur |  |  |
| T0\_Ventes | Id Facture | 10 | Oui | Entier |  |  |  | Clé de la Facture |  |  |
| T0\_Ventes | Id Produit | 20 | Oui | Entier |  |  |  | Clé du Produit |  |  |
| T0\_Ventes | Quantité | 30 | Oui | Réel |  |  |  | Quantité de Produit vendue |  |  |

## Créer les champs dans la base



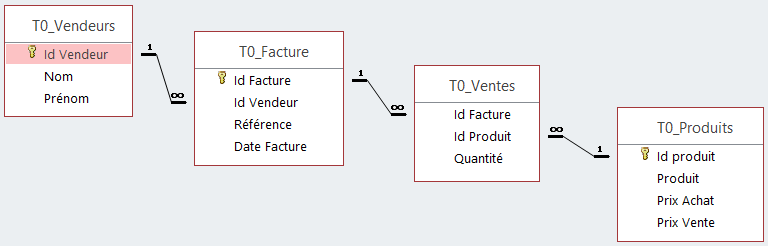






## Créer les liens entre les tables

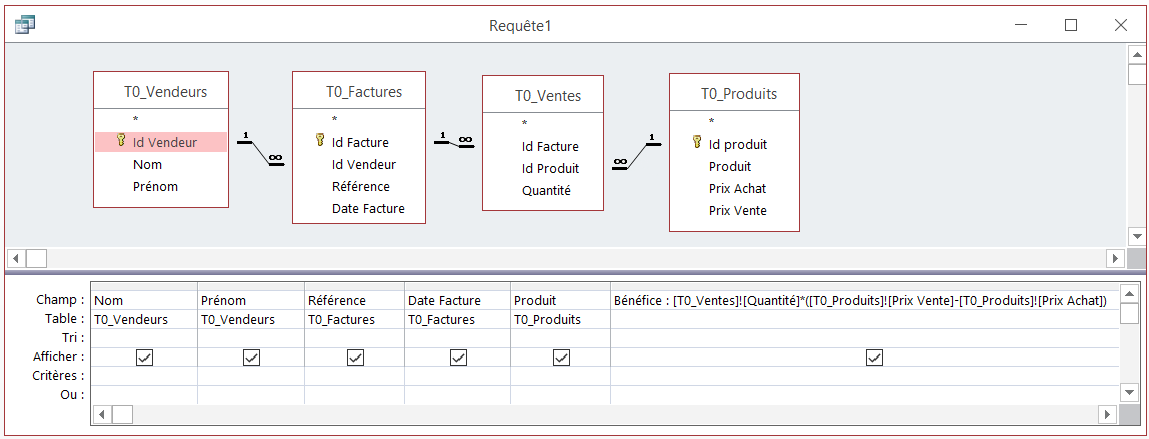




# Utilisation des Tables et Champs créés

## Utilisation des champs créés dans une requête

Grace au modèle relationnel employé, les liens entre les tables sont repris lors de la construction des requêtes :

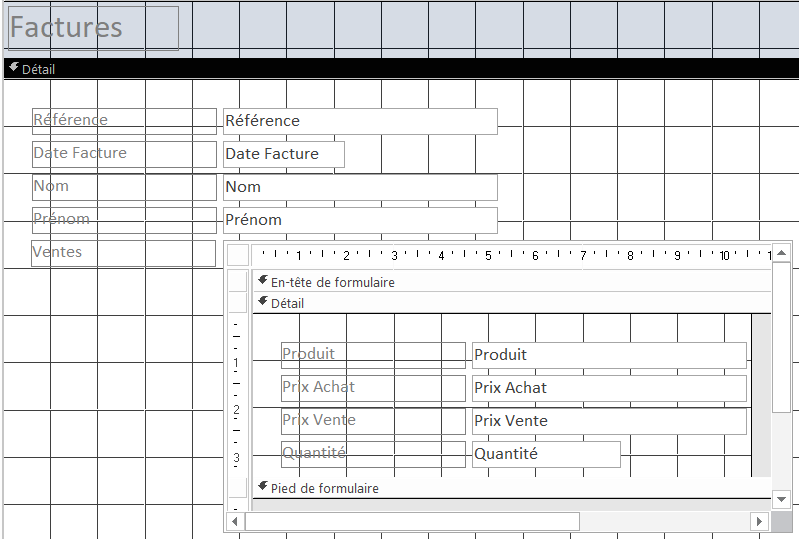


## Utilisation des champs créés dans un formulaire ou un état

Lors de la création d’un formulaire basé sur une ou plusieurs tables, des avantages multiples découlent du travail et du temps passé à la création des tables :

* Les caractéristiques des champs sont automatiquement reprises : taille maximale, type des champs, format des données saisissables, valeurs possibles dans des listes déroulantes…
* Libellés et descriptions des champs s’affichent sans devoir être retapés.
* Grace au modèle relationnel employé, les sous formulaires sont d’emblée liés au formulaire principal.

En utilisant l’assistant formulaire, on arrive ainsi au résultat suivant en quelques clics seulement :



# Conventions

Réfléchir à utiliser une même convention dans une application.

## Conventions de noms des tables

Utiliser une norme pour les noms des tables qui identifie clairement les différentes utilisations possibles pour les tables et l'ordre dans lequel elles interviennent. Par exemple :

* Sources liées ou importées : T0\_Xxxx
* Paramètres utilisateurs : TP\_Xxxx
* Tables de mapping : Proposition TP\_Mapping\_Xxxx
* Résultats (intermédiaires jusqu'à final) : T1\_Xxxx à T9\_Xxxx
* Paramétrage de l'application (Settings) : TS\_Xxxx

Ne pas oublier de décrire l’objet de chaque table et reporter cette description dans ses propriétés. Cela facilitera la documentation à posteriori. Faire de même pour chaque objet créé : Tables, Requêtes, Formulaires, Etats, Modules.

## Conventions de noms des champs

Utiliser des noms courts et fonctionnellement parlants, avec une bonne orthographe et une convention unique dans une base. Ex : « Nom Champ » ou « NomChamp » ou « Nom\_Champ ».

Il est recommandé d’utiliser une norme unique pour nommer tous les Champs de la base de données ACCESS, et on préférera la norme « Nom Champ » avec une Majuscule au début de chaque mot, par exemple « Id Facture ».

Les normes sans espace « IdFacture » ou « Id\_Facture » sont également rigoureuses, mais peuvent conduire à une perte de temps dans la suite des développements ACCESS.

## Conventions de noms des requêtes

Il sera bénéfique pour la maintenance ultérieure, que les requêtes soient préfixées et numérotées, avec un nom et une description parlantes, surtout si elles sont nombreuses :

Q010\_\*\*\*\*\*\_Desc\_Bloc\_\*\*\*\*\*\*\*\*

Q010\_010\_Action1

Q010\_020\_Action2

Il est préférable d’éviter les requêtes complexes ou trop imbriquées. Mieux vaut parfois créer des tables intermédiaires pour augmenter la rapidité des traitements.

## Conventions de noms des formulaires et états

Les formulaires peuvent être préfixés et numérotés s’ils sont nombreux ou si des sous blocs fonctionnels sont identifiés :

F01\_Menu

F02\_010\_Parameters

F02\_020\_Calculations

F02\_030\_Output

F09\_Logs

Tous les objets du formulaire seront préfixés de trigrammes standards MS :

lbl\_..., btn\_... ou cmd\_..., txt\_..., cmb\_... ou cbo\_..., chk\_..., fra\_..., lst\_..., opt\_..., sub\_...

## Conventions de développement des modules

Qualités recherchées : programmation procédurale, commentée, simple, maintenable.

Créer des modules de fonctions par thèmes fonctionnels, par exemple : Export XL, Calculs de TEG…

Ajouter un bloc d'en-tête au moins aux procédures et fonctions importantes, avec une description précise, qui évitera de décrypter parfois des milliers de lignes de codes, avant d’en comprendre l’objet :

=================================================

' Author :

' Name : Func or Sub name

' Purpose : Description fonctionnelle

' Parameters : Impact détaillé des paramètres

' Creation date :

' Modifications : Date + Author + Desc

'=================================================

Déclaration obligatoire de toutes les variables :

Option Explicit

Ne pas utiliser de variables globales ou a minima.

Produire un code lisible, indenté, commenté.

Utiliser des noms de variables parlants, avec si possible le type de la variable :

iDeb, sName, bTest, dDateEnd...

De façon générale, réfléchir à une convention de nommage des objets et s’y tenir au cours d’un développement. Par exemple : <https://www.databasezone.com/techdocs/naming.html>.

Ajouter si besoin une gestion des erreurs, pas forcément nécessaire pour du développement rapide.