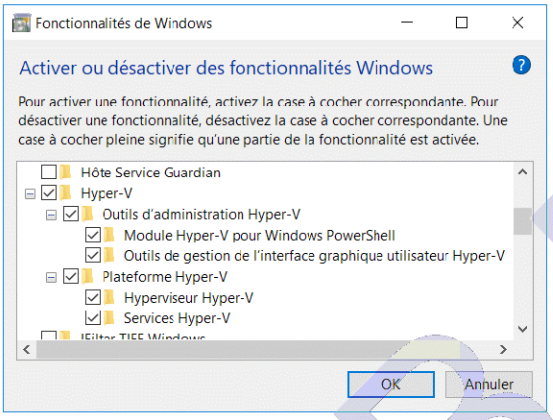
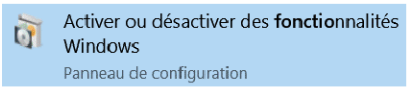
SQL Server

Déployer Hyper-V

- Activer la virtualisation dans le bios.
- Activer la fonctionnalité Hyper-V.

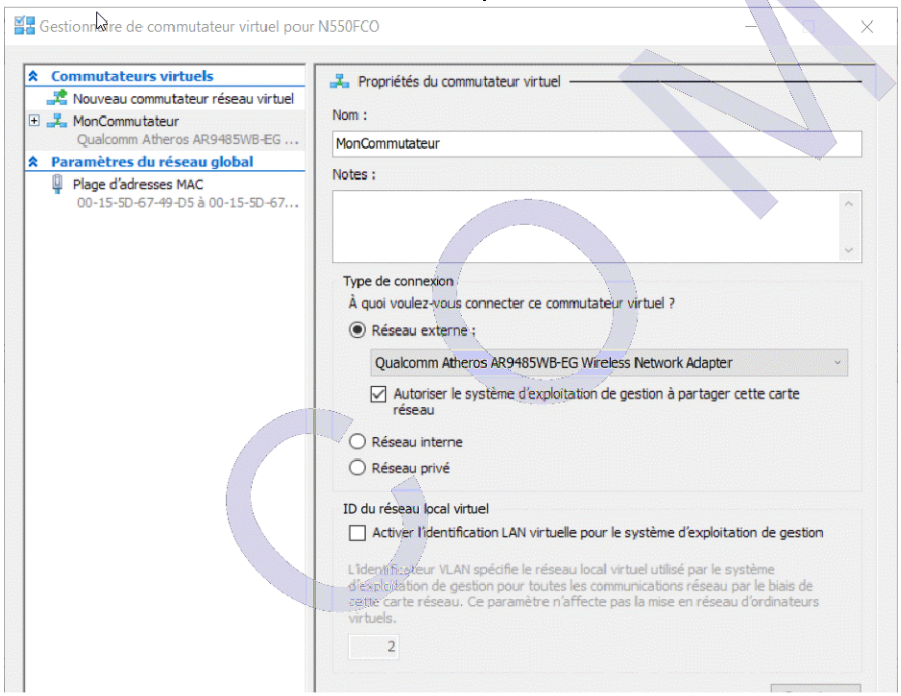




Outil de gestion de l'interface graphique utilisateur Hyper-V : Application permettant de déployer des machines virtuelles

Plateforme Hyper-V : permet de créer des ordinateurs et réseaux virtuels

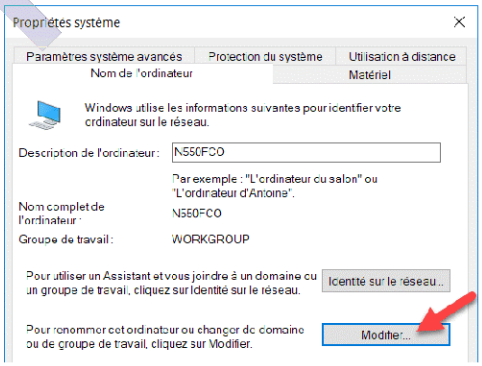
- Créer un nouveau commutateur virtuel (Actions→ Gestionnaire de commutateur virtuel)



- Installer Windows 10 en créant un nouvel ordinateur virtuel (Actions→Nouveau→Ordinateur Virtuel) (Génération 2 : UEFI, 3072Mo de mémoire, réseau : MonCommutateur, 25Go disk, installation ISO)

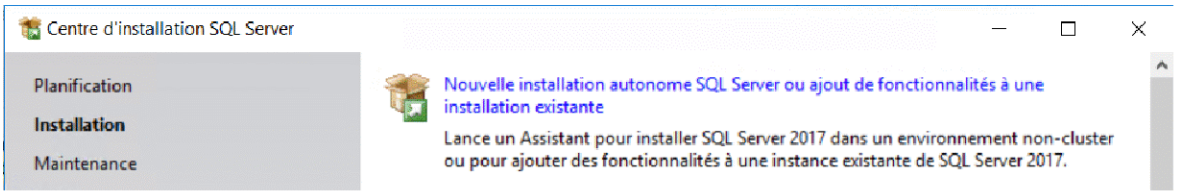
Remarque pour étendre un disque : étendre dans les paramètre de l'ordinateur virtuel PUIS dans le gestionnaire de disque de cet ordinateur une fois lancé)

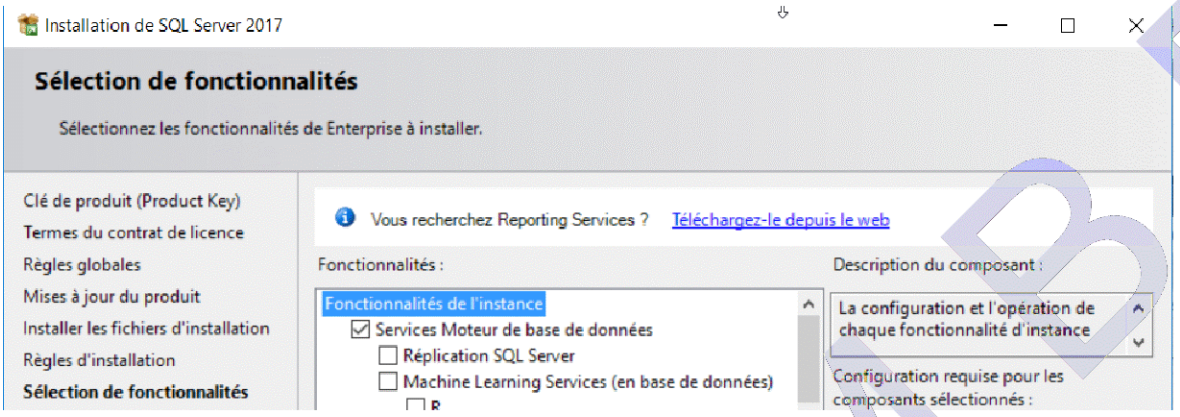
- Renommer l'ordinateur avant d'installer SQL Serveur.

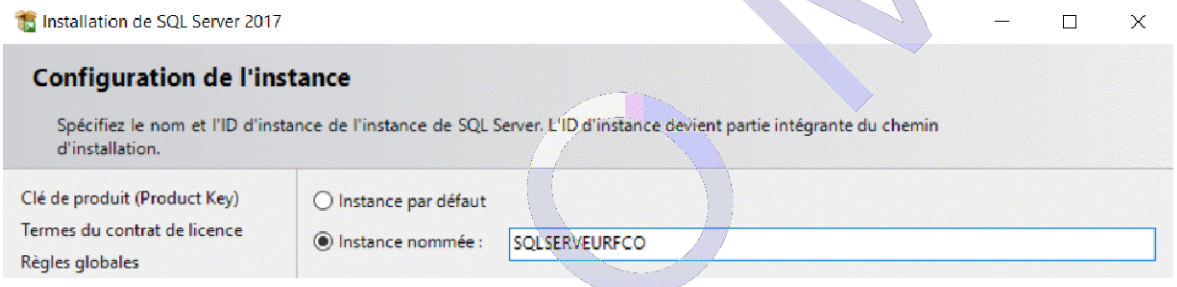


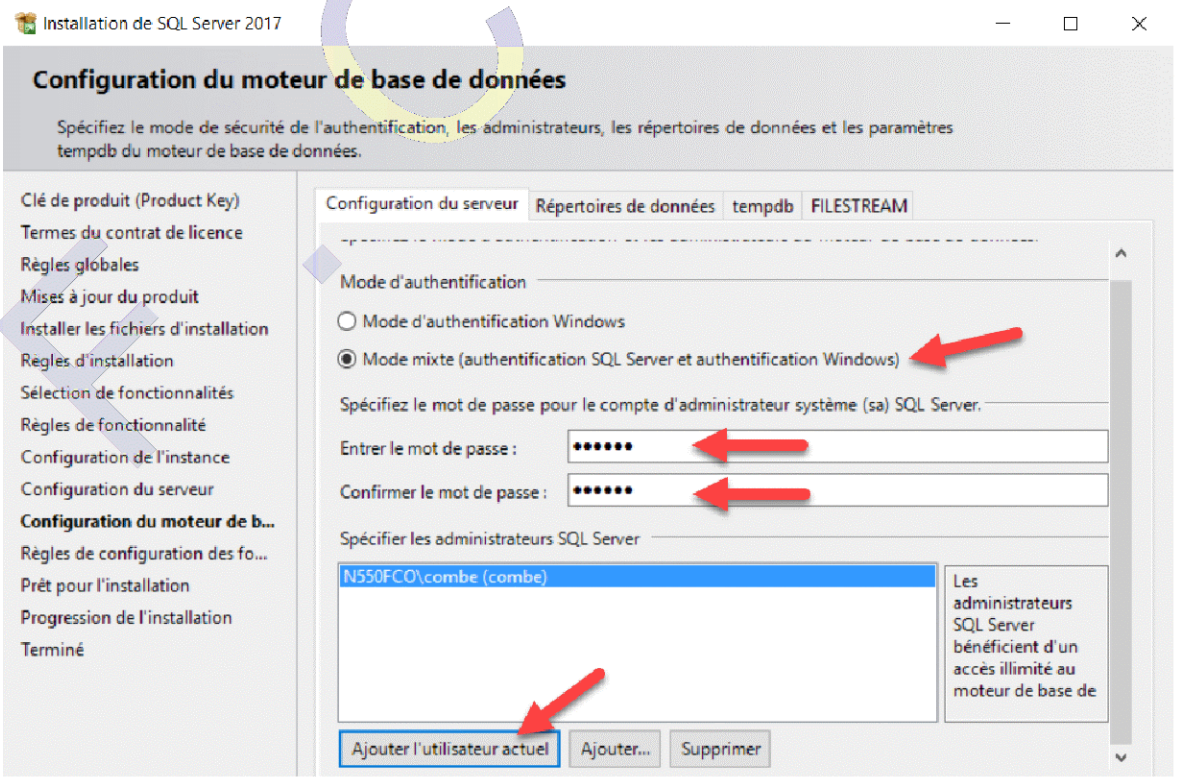
Installer SQL Serveur (Express limité à 10Go de BdD ou supérieur) : Monter l'ISO dans la VM

Sélectionner uniquement l'installation du moteur





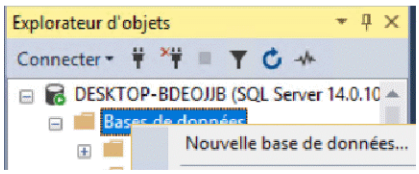




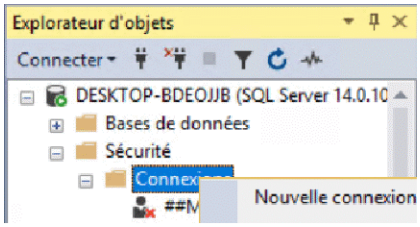
Installer SQL Management Studio (SSMS)

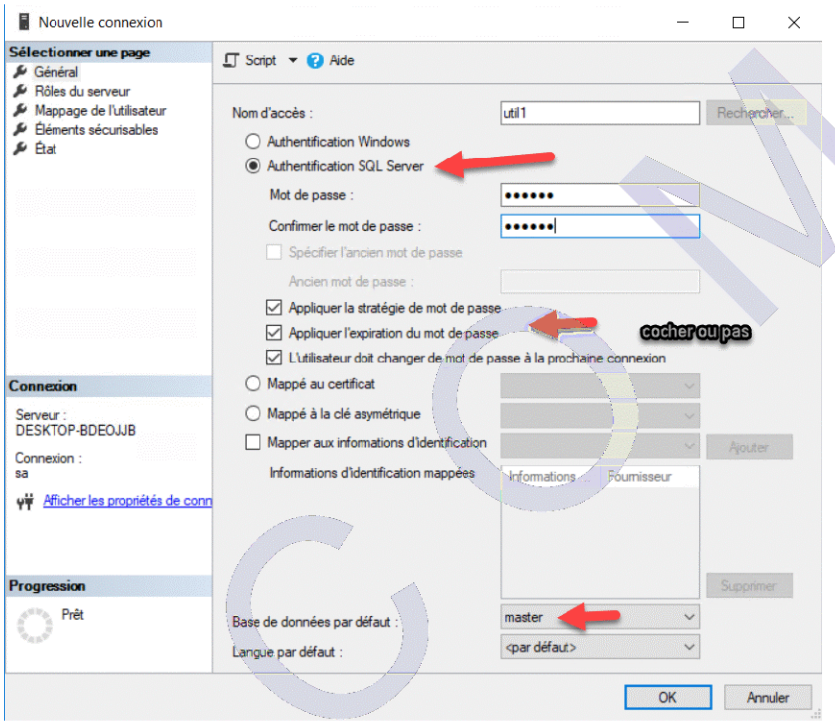
Administrer le Serveur BdD avec SSMS

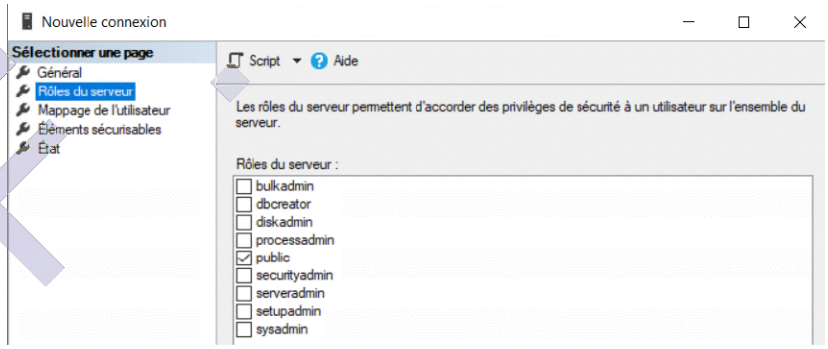
- Se connecter en tant qu'administrateur. Le login authentification SQL : sa
- Créer des bases de données

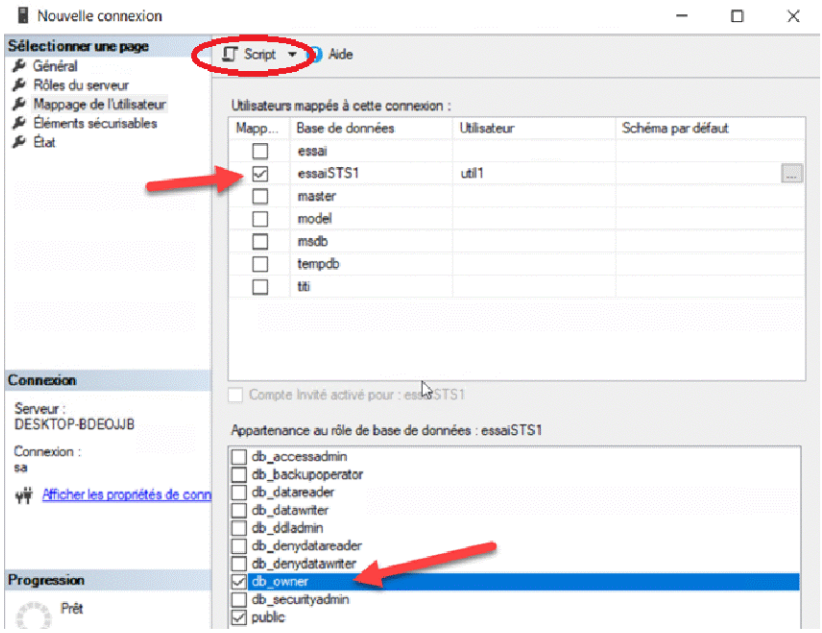


- Créer des utilisateurs pour limiter l'accès à une base



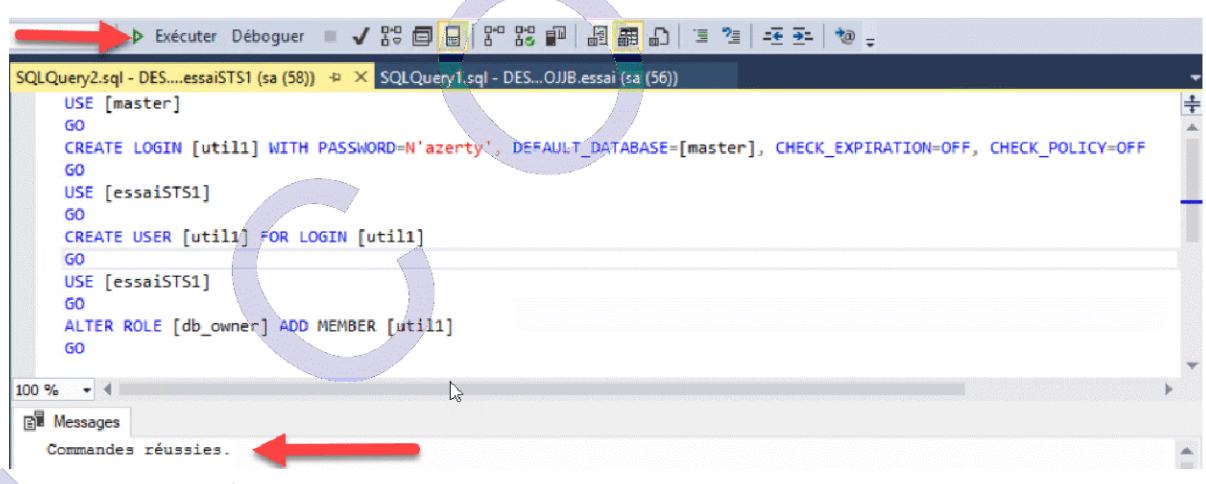






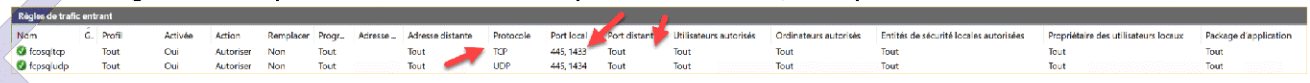
Finaliser l'opération ou Script de création d'un utilisateur

```
USE [master]
GO
CREATE LOGIN [util1] WITH PASSWORD=N'azerty', DEFAULT_DATABASE=[master], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
USE [essaiSTS1]
GO
CREATE USER [util1] FOR LOGIN [util1]
GO
USE [essaiSTS1]
GO
ALTER ROLE [db_owner] ADD MEMBER [util1]
GO
```

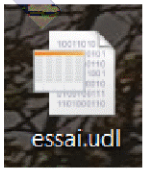


- Test de connexion distante.

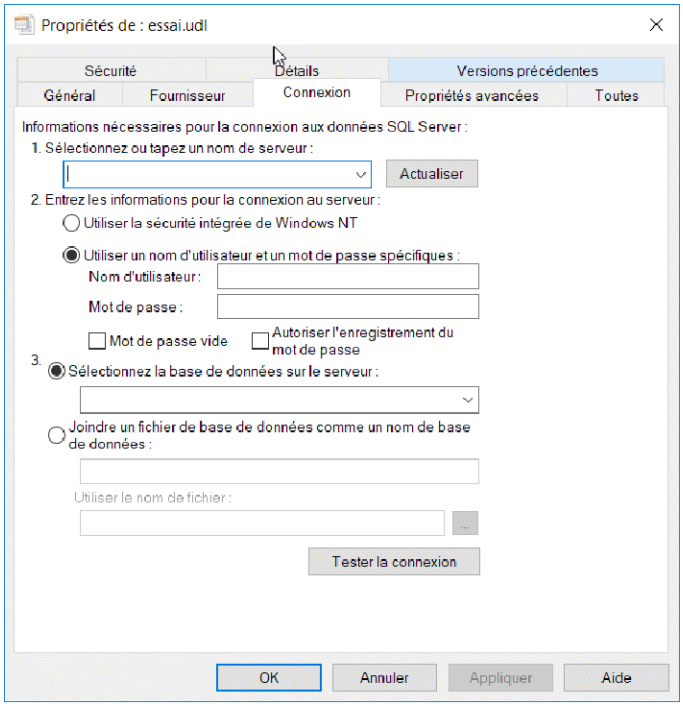
Créer 2 règles dans le pare-feu : ouverture TCP port 445 et 1433 ; UDP port 445 et 1434



Créer un fichier essai.udl sur l'ordinateur distant



puis bouton droit propriété



Si LocalDB

Cmd :

SqlLocalDB info

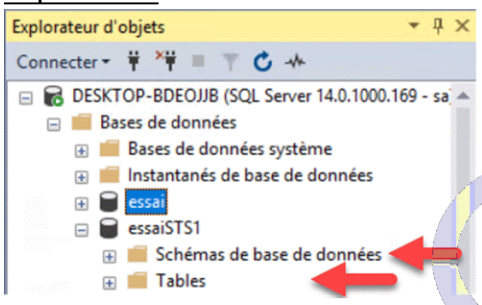
SqlLocalDB info MSSQLLOCALDB

SqlLocalDB s MSSQLLOCALDB

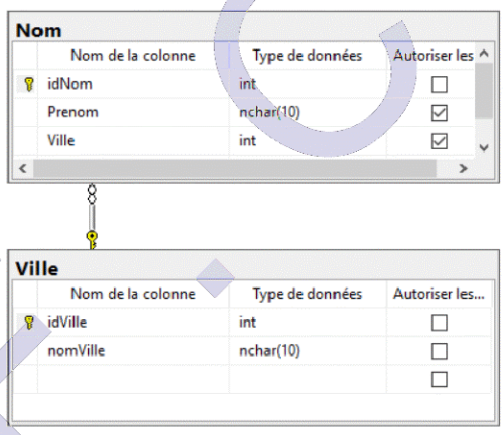
- Création des Tables

3 manières : manuelle (bouton droit sur Table) ou graphique (schéma...) depuis SSMS ou ADO Visual studio

Depuis SSMS :



Ex schéma

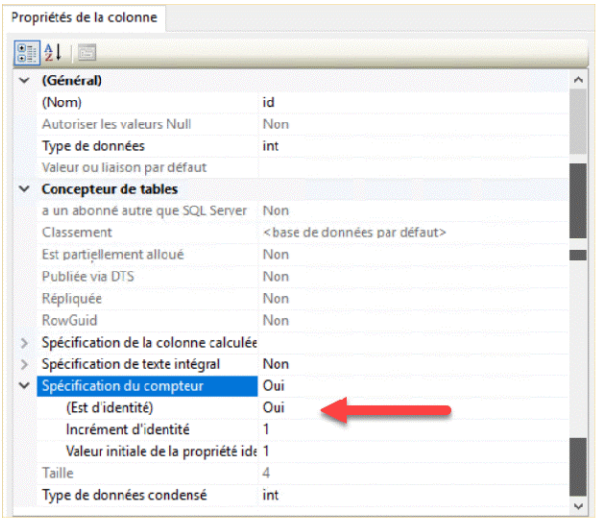


Bouton droit sur une table→vue table pour afficher différemment.

Bouton droit dans le vide→ Nouvelle table

Etirer de la clé idVille (logo clé primaire) de la table Ville vers le champ Ville de la table Nom pour créer une relation. Un Nom est associé à 1 Ville. Une Ville est associée à plusieurs Nom

Attention à la clé primaire AutoIncrément :



Depuis Visual studio :

Lien utile https://msdn.microsoft.com/en-us/library/mt715492.aspx

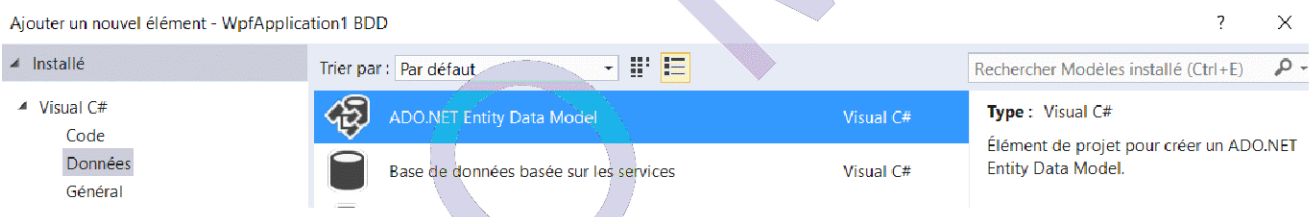
Dans Package Manager Console (menu Outil→Gestionnaire de package NUGET)

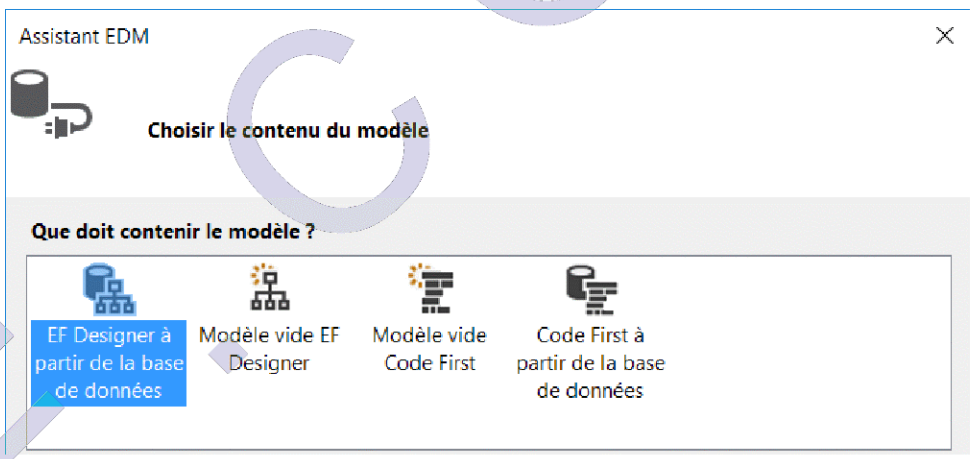
Sinon dan menu Outils→Extension de mise à jour→En ligne : rechercher entityframework (v6.x.x)

Où Click droit sur la solution→Gérer les Package NUGET pour les versions

Une fois installer entityframework (framework de persistance BdD)

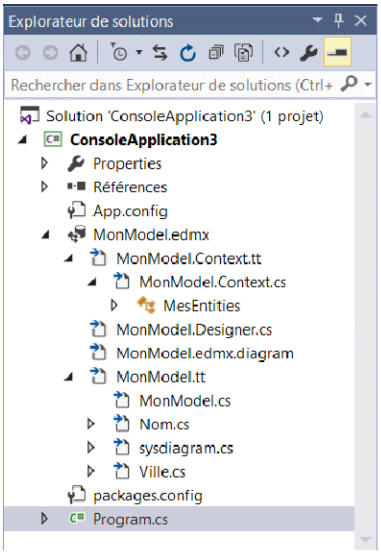
Ajouter à la solution



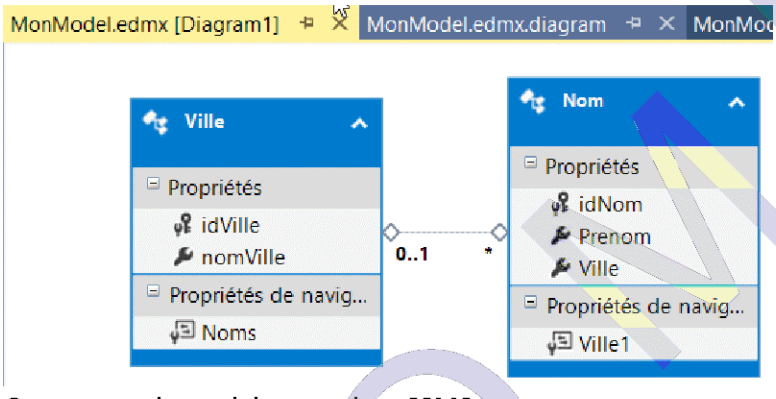


EF designer à partir de la base de données ou Modèle vide EF Designer

1. EF Designer



Double click sur MonModel.edmx (racine de ce qui a été créé par EF Designer)



On retrouve le model conçu dans SSMS.

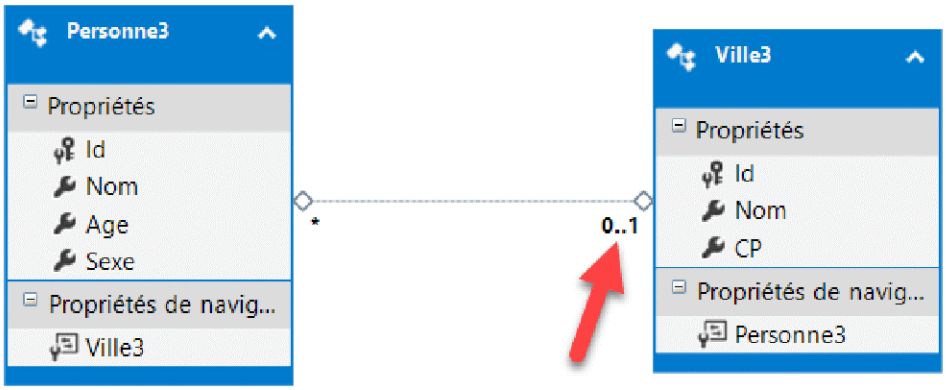
Test Console

```
class Program
{
    0 références
    static void Main(string[] args)
    {
        using (MesEntities db = new MesEntities())
        {
            List<Ville> v;
            v = db.Villes.ToList();
        }
    }
}
```

2. EF Vide

On design graphiquement depuis Visual les Tables et leurs relations.

Exemple :



Pour faire la relation :

Etirer l'Id de Ville3 vers l'entité personne3
Changer la multiplicité (1 vers 0..1) pour qu'une personne n'appartienne à aucune ville
Sauver
Bouton droit à coté du schéma et « Générer la base de données à partir du modèle
Lancer le script Model1.edmx.sql crée

Remarque additive.
La chaine de connexion est inscrite automatiquement dans App.Config
Pour faire une connexion manuelle, commenter :

MonModel.Context.cs | MonModel.Designer.cs | MonModel.edmx.diagram | MonModel.tt | App.config

```
<startup>
  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
</startup>
<!--<connectionStrings>
  <add name="MonModelContainer" connectionString="metadata=res://*/MonModel.csdl|
</connectionStrings>-->
<!--<entityFramework>
  <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnec
    <parameters>
      <parameter value="mssqllocaldb" />
    </parameters>
  </defaultConnectionFactory>
  <providers>
    <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlS
  </providers>
</entityFramework>-->
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
```

Un exemple de code C# avec SELECT, INSERT, UPDATE, DELETE

```
public partial class MonModelContainer : DbContext
{
    public MonModelContainer(string cc)
        : base(cc)
    {
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Specify the provider name, server and database.
        string providerName = "System.Data.SqlClient";
        //string serverName = @"(localdb)\MSSQLLocalDB";
        string serverName = @"DESKTOP-BDEOJJB";
        string databaseName = "essaiSTS1";
        //string databaseName = "BDDessai";

        // Initialize the connection string builder for the
        // underlying provider.
        SqlConnectionStringBuilder sqlBuilder = new SqlConnectionStringBuilder();
        sqlBuilder.UserID = "tata"; sqlBuilder.Password = "tata";
        // Set the properties for the data source.
        sqlBuilder.DataSource = serverName;
        sqlBuilder.InitialCatalog = databaseName;
        //sqlBuilder.IntegratedSecurity = true;  // true pour authentification windows, false = sql
        sqlBuilder.IntegratedSecurity = false;

        // Build the SqlConnection connection string.
        string providerString = sqlBuilder.ToString();

        // Initialize the EntityConnectionStringBuilder.
```

```
        EntityConnectionStringBuilder entityBuilder =
            new EntityConnectionStringBuilder();

        //Set the provider name.
        entityBuilder.Provider = providerName;

        // Set the provider-specific connection string.
        entityBuilder.ProviderConnectionString = providerString;

        // Set the Metadata location.
        entityBuilder.Metadata =
@"res://*/MonModel.csdl|res://*/MonModel.ssdl|res://*/MonModel.msl";

        using (MonModelContainer db = new MonModelContainer(entityBuilder.ToString()))
        {
                //Eleve e1 = new Eleve() { Id = 5, Nom = "zoe595", Prenom = "titi" };
                //db.EleveSet.Add(e1);
                //db.SaveChanges();

            Entity1 a1 = new Entity1() { nom = "toto" };
            Entity1 a2 = new Entity1() { nom = "titi" };
            Entity2 a3 = new Entity2() { numero = 54 }; a3.Entity1.Add(a1); a3.Entity1.Add(a2);
            db.Entity1Set.Add(a1);
            db.Entity1Set.Add(a2);
            db.Entity2Set.Add(a3);
            db.SaveChanges();
        }

        using (MonModelContainer db = new MonModelContainer())
        {
            //*** INSERT : Id inutile
            Eleve e1 = new Eleve() { Id = 5, Nom = "zoe59", Prenom = "titi" };
            db.EleveSet.Add(e1);
            db.SaveChanges();

            //*** SELECT
            List<Eleve> a;
            if (db.EleveSet.Count() < 10)
                a = db.EleveSet.ToList();

            //*** SELECT LANGAGE SQL
            List<Eleve> r1 = db.EleveSet.SqlQuery("SELECT * FROM EleveSet WHERE Nom LIKE
'%zoe%'").ToList();

            //*** SELECT expression lambda
            List<Eleve> r2 = (db.EleveSet.Where(x => x.Nom == "zoe")).ToList();

            //***SELECT link
            IEnumerable<Eleve> b = from x in db.EleveSet
                                   where x.Nom.Contains("ZOZO")
                                   select x;
            Eleve c = b.First();

            //***UPDATE
            c.Nom = "ZOZI";
            db.SaveChanges();

            //*** INSERT relationnel
            Absence d = new Absence() { DateDebut = DateTime.Now, DateFin = DateTime.Now, Motif =
"retard7" };
            c.Absence.Add(d);
            db.SaveChanges();
            // OU
            Absence e = new Absence() { Eleve = c, DateDebut = DateTime.Now, DateFin = DateTime.Now,
Motif = "retard8" };
            db.AbsenceSet.Add(e);
            db.SaveChanges();

            //***DELETE relationnel
```

```
            IEnumerable<Eleve> f = from x in db.EleveSet
                                   where x.Id == 6
                                   select x;
            Eleve g = f.First();
            IEnumerable<Absence> h = from x in db.AbsenceSet
                                        where x.Eleve.Id == g.Id
                                        select x;
            db.AbsenceSet.RemoveRange(h);
            db.SaveChanges();
            db.EleveSet.Remove(g);
            db.SaveChanges();
        }
    }
}
```

Requêtes SQL

INSERT

```
db.Database.ExecuteSqlCommand("INSERT INTO Personne3Set (Nom, Age, Sexe) VALUES ('titi', '56', 1)");
```

UPDATE

```
db.Database.ExecuteSqlCommand("UPDATE Personne3Set SET Nom='tutu', Age='84', Sexe=0 WHERE Id=4");
```

```
db.Database.ExecuteSqlCommand("UPDATE Personne3Set SET Ville3_id = (SELECT id FROM Ville3Set WHERE Nom = 'dfhsd') WHERE Nom = 'titi'");
```

https://openclassrooms.com/courses/administrez-vos-bases-de-donnees-avec-mysql/jointures-et-sous-requetes-modification-de-donnees :

```
UPDATE Animal       -- Classique !
INNER JOIN Espece   -- Jointure.
    ON Animal.espece_id = Espece.id
    -- Condition de la jointure.
SET Animal.commentaires = Espece.description
    -- Ensuite, la modification voulue.
WHERE Animal.commentaires IS NULL
    -- Seulement s'il n'y a pas encore de commentaire.
AND Espece.nom_courant IN ('Perroquet amazone', 'Tortue d''Hermann');
    -- Et seulement pour les perroquets et les tortues.
```

A voir car pb !!! mais le lien est bien

DELETE

```
db.Database.ExecuteSqlCommand("DELETE FROM Personne3Set WHERE Sexe = 0");
```

SELECT

```
List<Personne3> m = db.Personne3Set.SqlQuery("SELECT * FROM Personn3Set INNER JOIN VilleSet ON PersonneSet.Ville_Id = VilleSet.Id WHERE VilleSet.Nom LIKE '%l%' COLLATE French_CI_AS").ToList();
```

La syntaxe ci-dessus stipule qu'il faut sélectionner les enregistrements des tables **PersonneSet** et **VilleSet** lorsque les données de la colonne « Ville_Id » (clé étrangère) de PersonneSet est égal aux données de la colonne Id de VilleSet.

Retourne en plus les personnes dont le nom de ville contient un « a » (LIKE '%a%') et case sensitive : minuscule (COLLATE French_CI_AS)