

## Leçon 2 : INDEX, CONTRAINTES D'INTEGRITES ET MANIPULATION DES DONNEES SOUS MS SQL SERVER

### ✓ Objectifs d'apprentissage :

À la fin de cette leçon, vous serez capable de :

- Définir les Concepts fondamentaux des index
- Définir les contraintes d'intégrités
- Manipuler les données dans une table

## I- CONCEPTS FONDAMENTAUX DES INDEX

### 1- Définition

Dans la vie courante nous sommes entourés d'index : un code postal, le n° d'un immeuble dans une rue, les numéros de téléphone sont des index. Un index permet de trier, d'organiser, d'identifier, d'accélérer les recherches des données dans une table.

### 2- Structure logique d'un index

Nous avons dit que les données d'un index sont triées. Si un index est multi colonnes, le tri de la clef d'index fait que l'information y est vectorisée. En effet, chaque colonne supplémentaire précise la colonne précédente. De ce fait, la recherche dans un index multi colonne n'est accélérée que si les données cherchées sont un sous-ensemble ordonné du vecteur. Voyons cela en termes pratiques à l'aide d'un exemple....

Exemple :

CLI_NOM	CLI_PRENOM	CLI_DATE_NAISSANCE
---------	------------	--------------------

KOUAKOU	PAUL	21/01/1990
KOUAKOU	ERIC	21/01/1996
KOUAKOU	ERIC	01/06/1998
KOUAKOU	CHARLES	21/01/1999
DRO	NESTOR	11/02/1987
DRO	NESTOR	21/01/1990

...

Dans cet index composé d'un nom d'un prénom et d'une date de naissance, la recherche sera efficace pour les informations suivantes :

- CLI\_NOM
- CLI\_NOM + CLI\_PRENOM
- CLI\_NOM + CLI\_PRENOM + CLI\_DATE\_NAISSANCE

En revanche, la recherche sur une seule colonne CLI\_PRENOM ou CLI\_DATE\_NAISSANCE et a fortiori sur ces deux colonnes n'aura aucune efficacité du fait du tri relatif des colonnes entre elles (vectorisation)... En effet chercher "Paul" revient à parcourir tout l'index (balayage ou *scan*) alors que chercher "MARTIN", revient à se placer très rapidement par dichotomie au bon endroit (recherche ou *seek*).

### 3- Création automatique et manuelle d'un index

#### ▪ Création automatique

La plupart des SGBDR créent automatiquement des index lors de la pose des contraintes PRIMARY KEY et UNIQUE. En effet, le travail de vérification de l'unicité d'une clef primaire ou candidate s'avérerait extrêmement long sans un index.

En revanche la plupart des SGBD relationnels ne prévoient pas de créer des index sous les clefs étrangères (FOREIGN KEY), ni sous les colonnes auto incrémentées.

#### ▪ Création manuelle

Une des syntaxes basiques de création d'un index est la suivante :

```
CREATE INDEX #nom_index  
ON #nom_schem.#nom_table  
(#colonne1 [ { ASC | DESC } ], #colonne2 [ ... #colonneN ] )
```

Les colonnes spécifiées dans la parenthèse constituent la clef d'index, c'est-à-dire les informations recherchées.

**EXERCICE :** (Cet exercice fait appel à la recherche de la part de l'apprenant)

- 1) Quel est le rôle d'un index dans une table?
- 2) Quel est le rôle de PRIMARY KEY ?
- 3) Donner la syntaxe basique de création d'un index sous MS SQL Server ?

## II- CONTRAINTES D'INTEGRITES

Une contrainte d'intégrité est une clause permettant de contraindre la modification de tables, faite par l'intermédiaire de requêtes d'utilisateurs, afin que les données saisies dans la base soient conformes aux données attendues. Ces contraintes doivent être exprimées dès la création de la table grâce aux mots clés suivants :

- CONSTRAINT
- DEFAULT
- NOT NULL
- UNIQUE
- CHECK

#### ➤ DEFAULT

Ce langage SQL permet de définir une valeur par défaut lorsqu'un champ de la base n'est pas renseigné grâce à la clause *DEFAULT*. Cela permet notamment de faciliter la création de tables, ainsi que de garantir qu'un champ ne sera pas vide.

La clause *DEFAULT* doit être suivie par la valeur à affecter. Cette valeur peut être un des types suivants :

- constante numérique

- constante alphanumérique (chaîne de caractères)
- le mot clé **USER** (nom de l'utilisateur)
- le mot clé **NULL**
- le mot clé **CURRENT\_DATE** (date de saisie)
- le mot clé **CURRENT\_TIME** (heure de saisie)
- le mot clé **CURRENT\_TIMESTAMP** (date et heure de saisie)

➤ **NOT NULL**

Le mot clé *NOT NULL* permet de spécifier qu'un champ doit être saisi, c'est-à-dire que le SGBD refusera d'insérer des tuples dont un champ comportant la clause *NOT NULL* n'est pas renseigné.

➤ **UNIQUE**

Les contraintes sont des règles que Moteur de base de données SQL Server applique automatiquement. Par exemple, vous pouvez utiliser des contraintes **UNIQUE** pour garantir qu'aucune valeur en double n'est entrée dans des colonnes spécifiques ne faisant pas partie d'une clé primaire. Bien qu'une contrainte **UNIQUE** et une contrainte **PRIMARY KEY** assurent l'unicité, il est préférable d'avoir recours à une contrainte **UNIQUE** au lieu d'une contrainte **PRIMARY KEY** lorsque vous voulez assurer l'unicité d'une colonne (ou d'une combinaison de colonnes) qui n'est pas la clé primaire.

À la différence des contraintes **PRIMARY KEY**, les contraintes **UNIQUE** autorisent la valeur **NULL**. Cependant, comme pour toute valeur participant à une contrainte **UNIQUE**, une seule valeur **NULL** est autorisée par colonne. Une contrainte **UNIQUE** peut être référencée par une contrainte **FOREIGN KEY**.

Lorsque vous ajoutez une contrainte **UNIQUE** à une ou plusieurs colonnes dans la table, par défaut, le Moteur de base de données examine les données existantes dans les colonnes pour s'assurer que toutes les valeurs sont uniques. Si une contrainte **UNIQUE** est ajoutée à une colonne qui comporte des valeurs dupliquées, le Moteur de base de données retourne une erreur et n'ajoute pas la contrainte.

Le Moteur de base de données crée automatiquement un index **UNIQUE** pour appliquer l'impératif d'unicité de la contrainte **UNIQUE**. Dès lors, en cas de tentative d'insertion d'une ligne dupliquée, le Moteur de base de données retourne un message d'erreur indiquant que la contrainte **UNIQUE** a été violée et n'ajoute pas la ligne à la table. Sauf si un index cluster est explicitement spécifié, un index non-cluster unique est créé par défaut pour assurer l'application de la contrainte **UNIQUE**.

**EXERCICE :** (Cet exercice fait appel à la recherche de la part de l'apprenant)

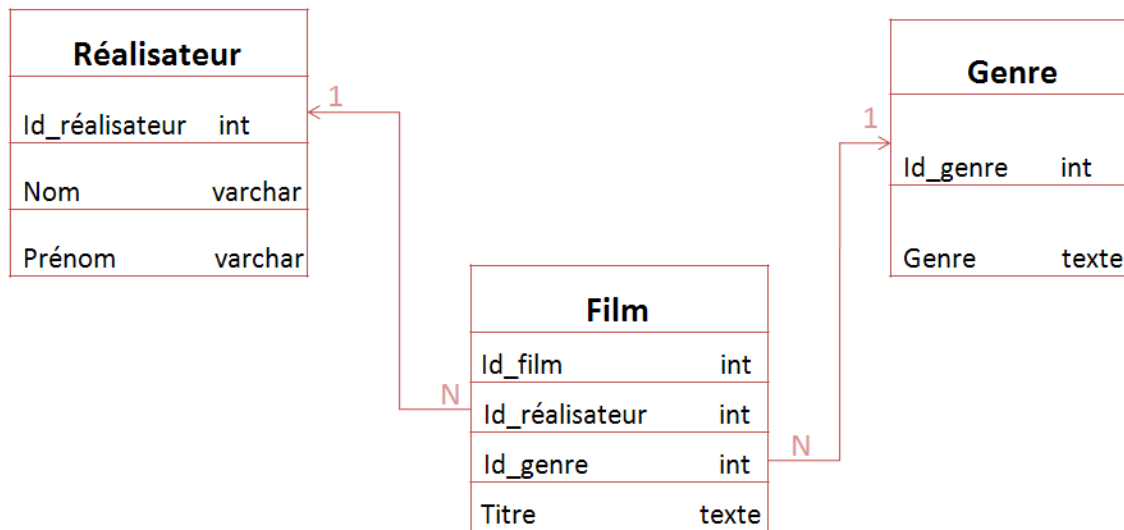
- 4) Quel est le rôle de la clause **DEFAULT** ?
- 5) Quelle est la différence entre les contraintes **PRIMARY KEY** et **UNIQUE** ?

### III- RENSEIGNER ET MANIPULER LES DONNEES

#### 1- Présentation de la base de données

Nous allons créer une petite base de données qui sera utilisée pour trier des films selon leurs genres et leurs réalisateurs.

Comme le montre l'exemple, cette base comporte trois tables : Film, Réalisateur et Genre.



- Chaque réalisateur doit avoir un identifiant en plus de son nom et de son prénom.
- On doit spécifier le genre du film et lui donner un identifiant.
- Pour chaque film, on doit donner un identifiant et un titre. Le genre et le réalisateur du film doivent être indiqués dans la table « film ».
- Chaque film doit être réalisé par un seul réalisateur et un réalisateur peut réaliser plusieurs films.
- Chaque film n'a qu'un seul genre mais un même genre peut être attribué à plusieurs films.

## 2- Création de la base de données

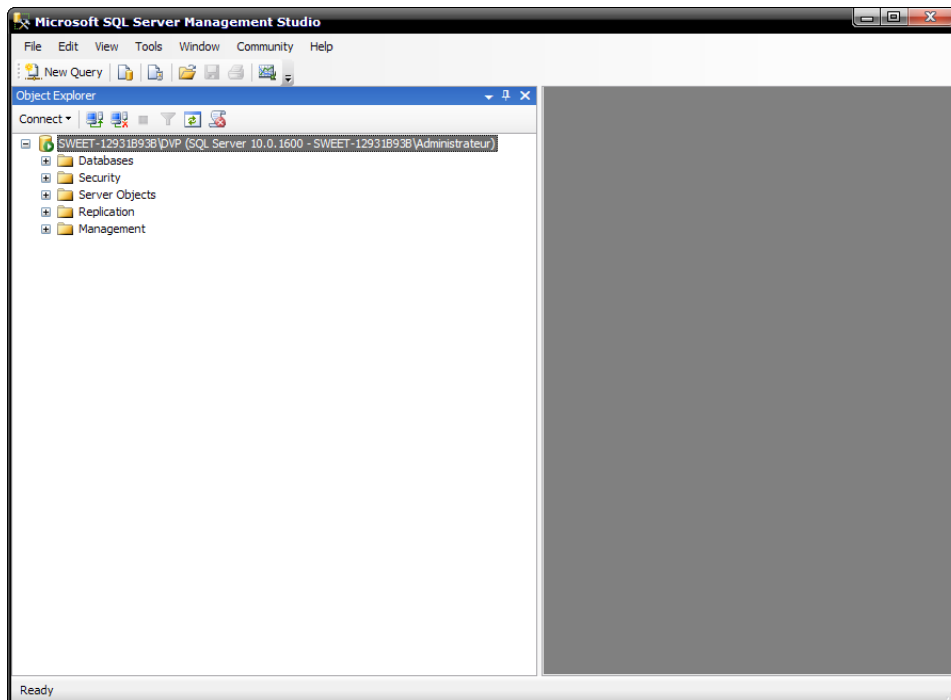
- On commence par lancer le logiciel à partir de l'icône de Microsoft SQL Server Management.

On renseigne ensuite :

- **Server type**
- **Server name**
- **Authentication**

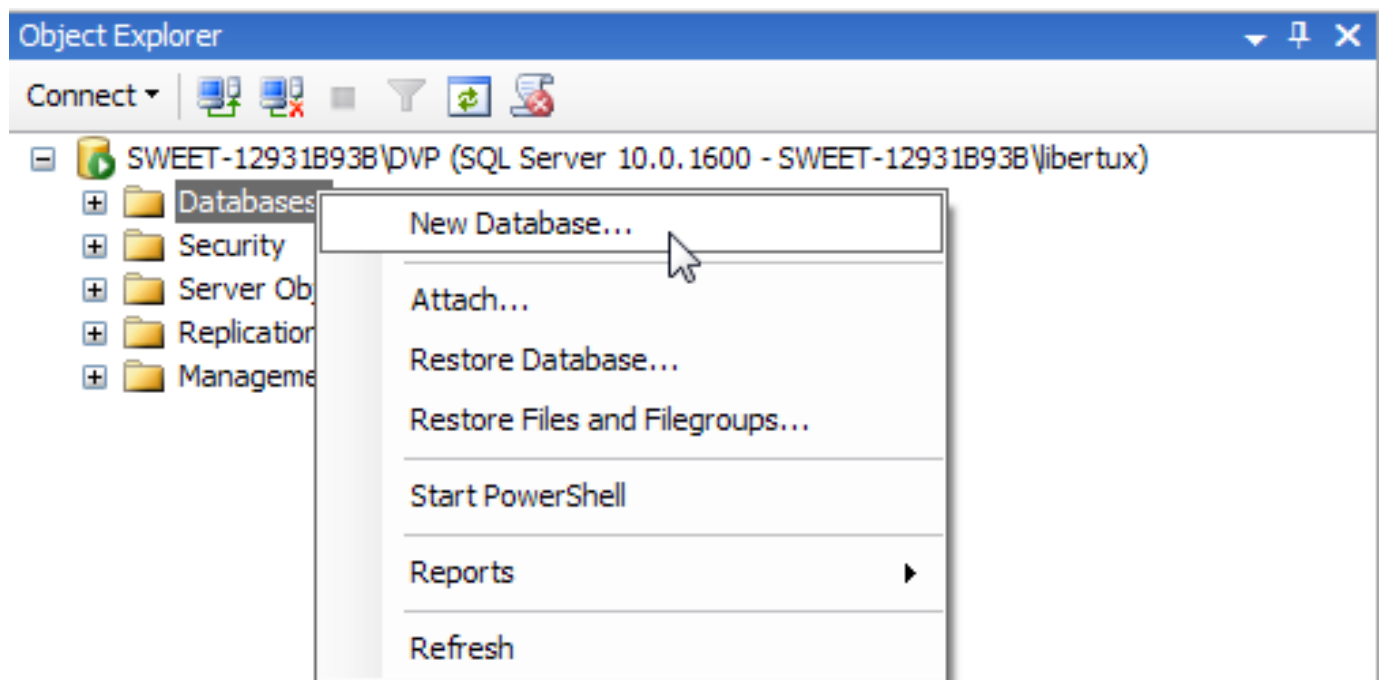
Pour se connecter au serveur, on clique sur « **Connect** »

- Une nouvelle fenêtre s'ouvre, elle représente l'interface initiale de l'outil

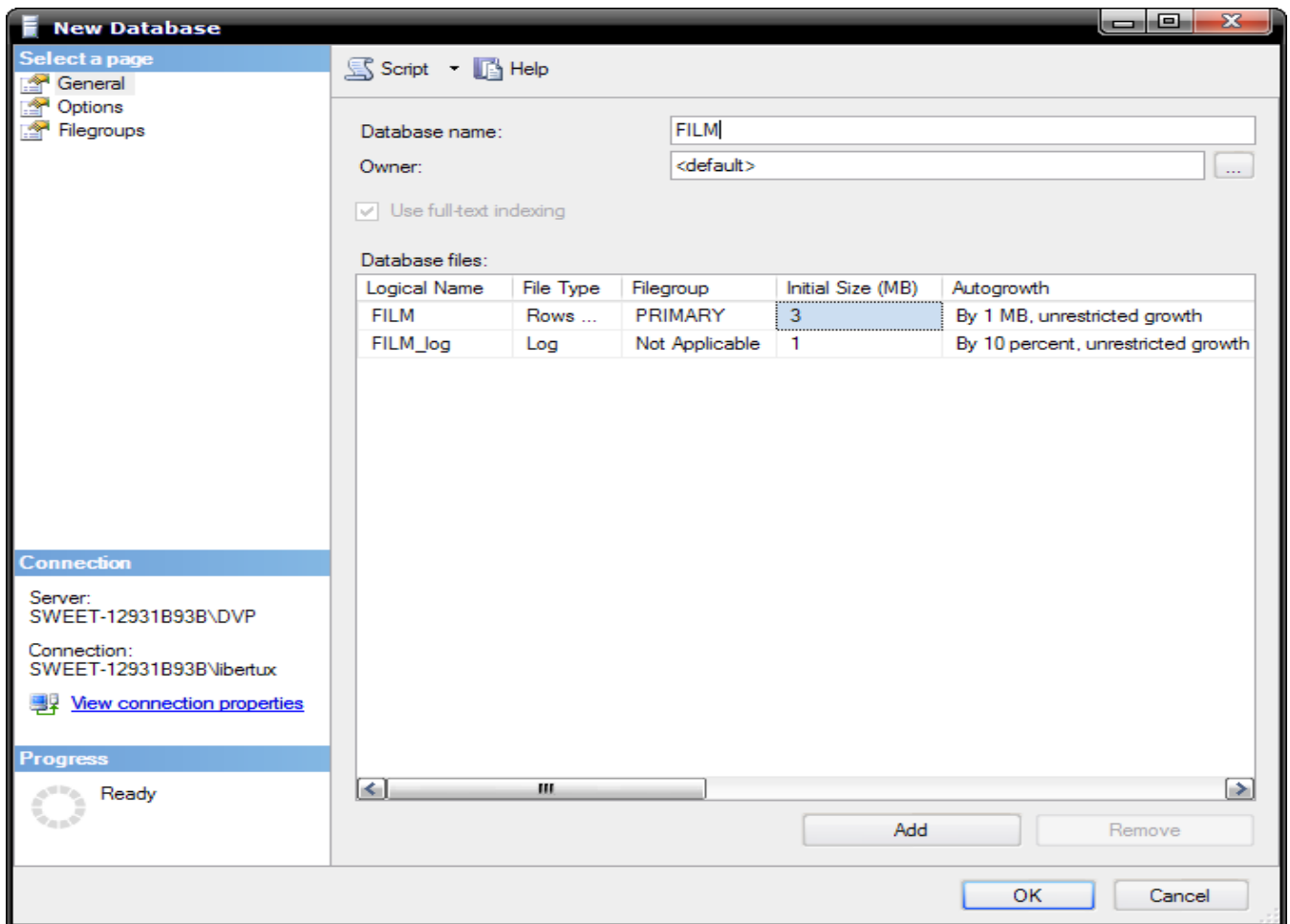


On voit que cette fenêtre est divisée en deux parties, d'une part la fenêtre « Explorer » à gauche qui nous affiche tous les objets du serveur et nous permet de naviguer entre ces différents composants ; la partie droite est une autre fenêtre qui nous montre les données et les informations sous forme de sommaire.

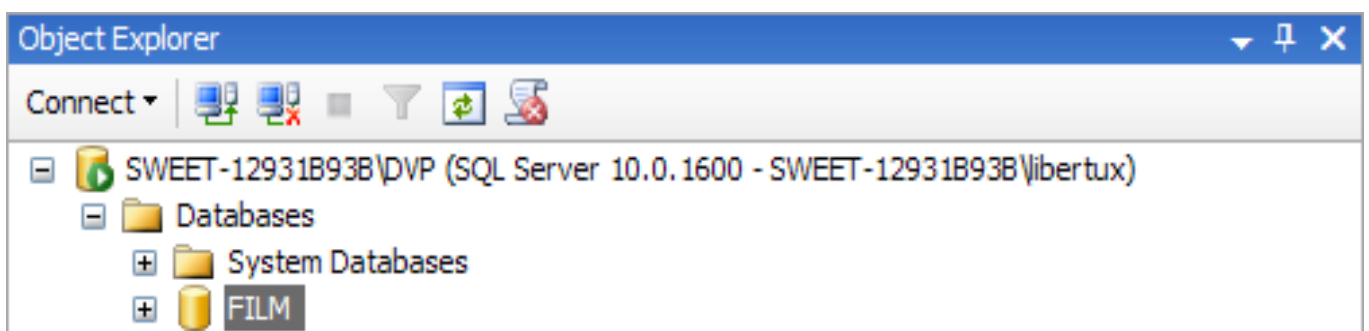
- Pour créer notre table, on doit cliquer avec le bouton droit sur « Databases » dans la fenêtre « Explorer », puis on clique sur « New Database »



- Dans la nouvelle fenêtre qui s'affiche, on va donner le nom «FILM » à notre base puis on clique sur « Ok »



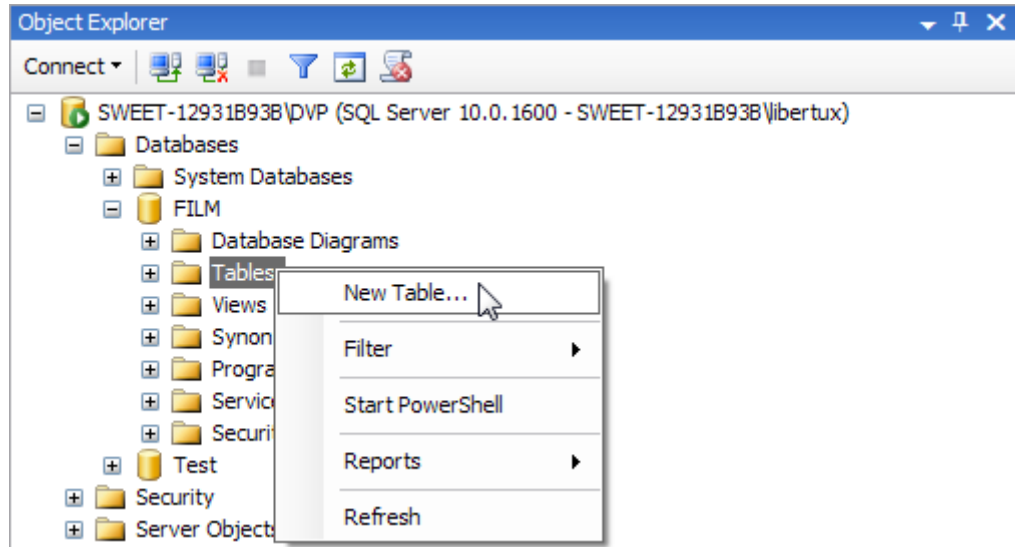
- On peut maintenant voir, à partir de la fenêtre « Explorer », que notre base est créée



### 3- Création des tables

Nous savons que notre base doit contenir trois tables, nous allons les créer maintenant.

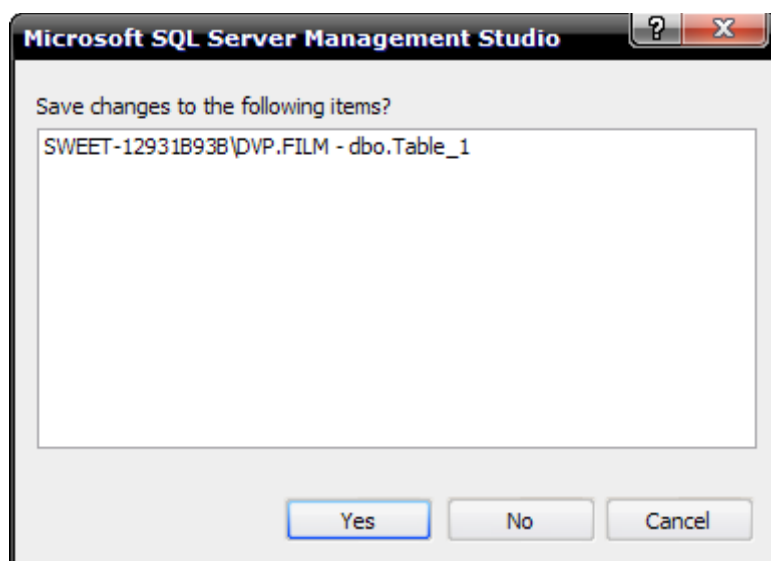
Toujours dans la fenêtre « Explorer » on clique avec le bouton droit sur « Tables » dans notre base « FILM », puis on clique sur « New Table »



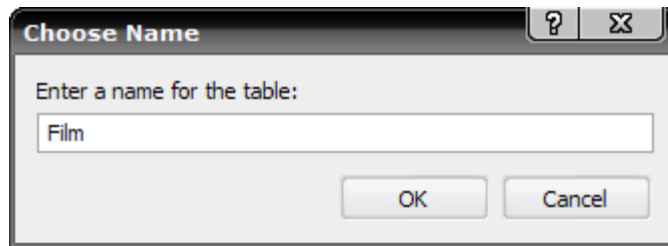
À droite et à la même place de la fenêtre « Sommaire » une nouvelle fenêtre s'ouvre, dans laquelle on doit saisir toutes les colonnes de la table

SWEET-12931B... dbo.Table_1*			
	Column Name	Data Type	Allow Nulls
	Id_film	int	<input type="checkbox"/>
	Id_réalisateur	int	<input type="checkbox"/>
	Id_genre	int	<input type="checkbox"/>
▶	Titre	text	<input type="checkbox"/>

Après avoir saisi toutes ces données, on ferme cette fenêtre. Une nouvelle fenêtre s'ouvre et nous demande si on veut enregistrer les changements effectués, on clique alors sur « Yes » pour confirmer.



Une autre fenêtre s'ouvre, dans laquelle on va saisir le nom de la table désirée puis on clique sur « Ok »

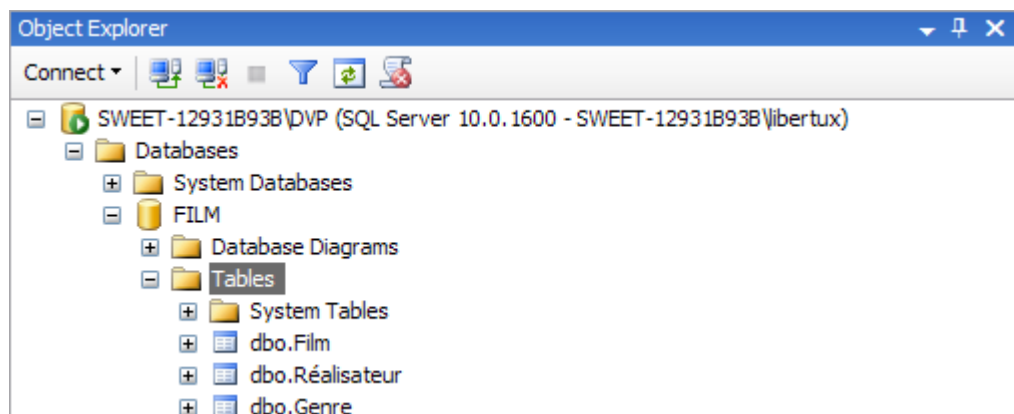


On refait les mêmes étapes avec les tables « Réalisateur » et « Genre » comme le montre les deux figures ci-dessous

SWEET-12931B... dbo.Table_1*			
Column Name	Data Type	Allow Nulls	
Id_réalisateur	int	<input type="checkbox"/>	
Nom	varchar(50)	<input type="checkbox"/>	
▶ Prénom	varchar(50)	<input type="checkbox"/>	

SWEET-12931B... dbo.Table_1*			
Column Name	Data Type	Allow Nulls	
Id_genre	int	<input type="checkbox"/>	
▶ Genre	text	<input type="checkbox"/>	

Nos trois tables sont maintenant créées. Pour vérifier que la création est bien réalisée on peut naviguer dans la fenêtre « Explorer » et voir si nos trois tables existent vraiment.



On voit bien que les trois tables sont créées sur notre base.

## 4- Création des clés

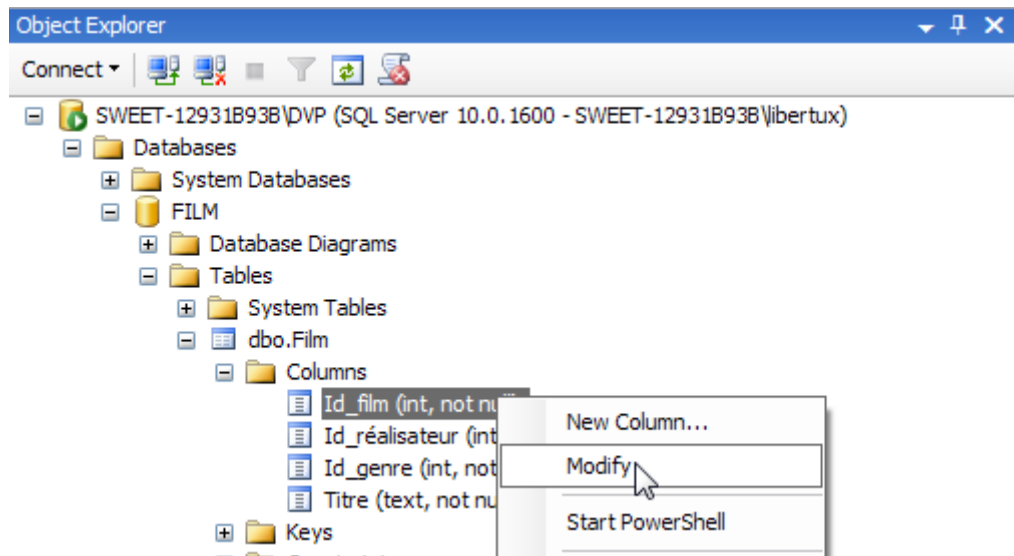
### a- Clés primaires


Notre base est formée de 3 tables, chacune des tables devant contenir une clé primaire. Dans l'étape suivante, nous allons créer ces clés.

Commençons par la table « Film », cette table a comme clé primaire la colonne « Id film ».


À partir de la fenêtre « Explorer », nous allons aller sur la colonne « Id film », un clic droit puis on clique sur « Modify »



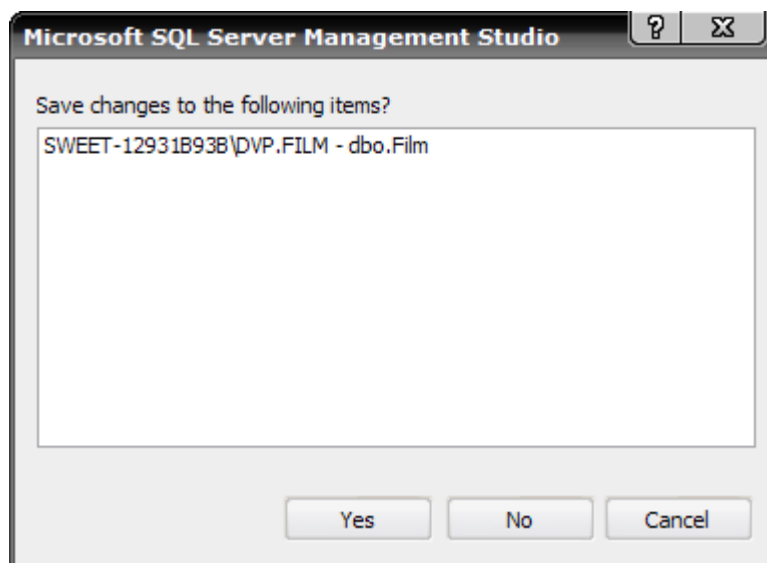


Cette table va s'afficher dans la fenêtre à droite, on sélectionne la colonne qu'on veut définir comme clé primaire puis dans la barre en haut on doit cliquer sur .

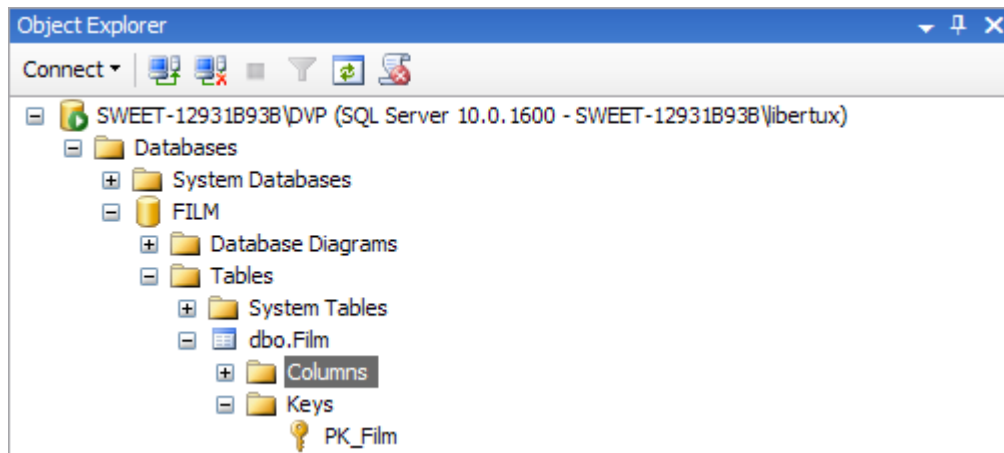
On remarque qu'une petite clé jaune s'affiche à côté de cette colonne

SWEET-12931B93B - dbo.Film*			
	Column Name	Data Type	Allow Nulls
	Id_film	int	<input type="checkbox"/>
	Id_réalisateur	int	<input type="checkbox"/>
	Id_genre	int	<input type="checkbox"/>
	Titre	text	<input type="checkbox"/>

Lorsqu'on ferme cette fenêtre une autre fenêtre s'affiche pour vérifier si on veut enregistrer les changements, alors on clique sur « Yes »



Dans la fenêtre « Explorer » on peut vérifier si la création de la clé primaire est réalisée correctement ou non.



Alors voilà comment doit s'afficher la clé primaire.

On refait le même travail pour les autres tables

SWEET-12931B9...o.Réalisateur*			
	Column Name	Data Type	Allow Nulls
PK	Id_réalisateur	int	<input type="checkbox"/>
	Nom	varchar(50)	<input type="checkbox"/>
	Prénom	varchar(50)	<input type="checkbox"/>

SWEET-12931B... - dbo.Genre*			
	Column Name	Data Type	Allow Nulls
PK	Id_genre	int	<input type="checkbox"/>
	Genre	text	<input type="checkbox"/>

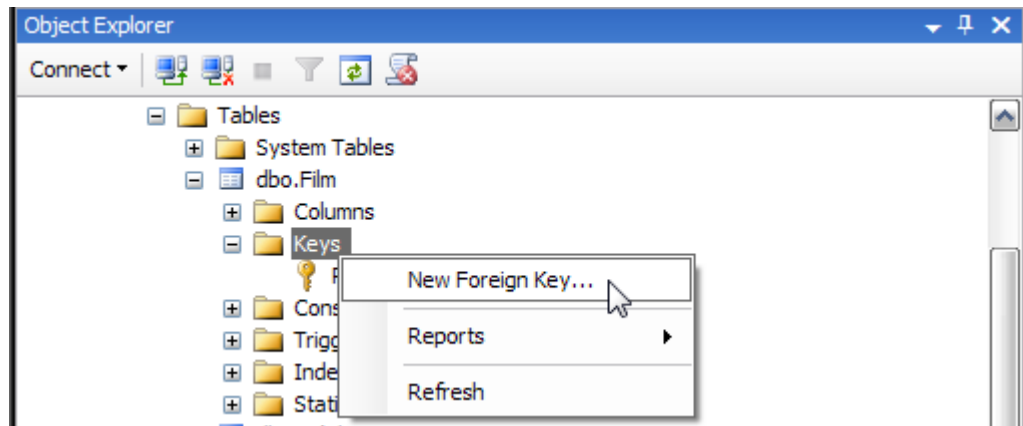
Maintenant, toutes les clés primaires sont créées.

## b- Clés étrangères

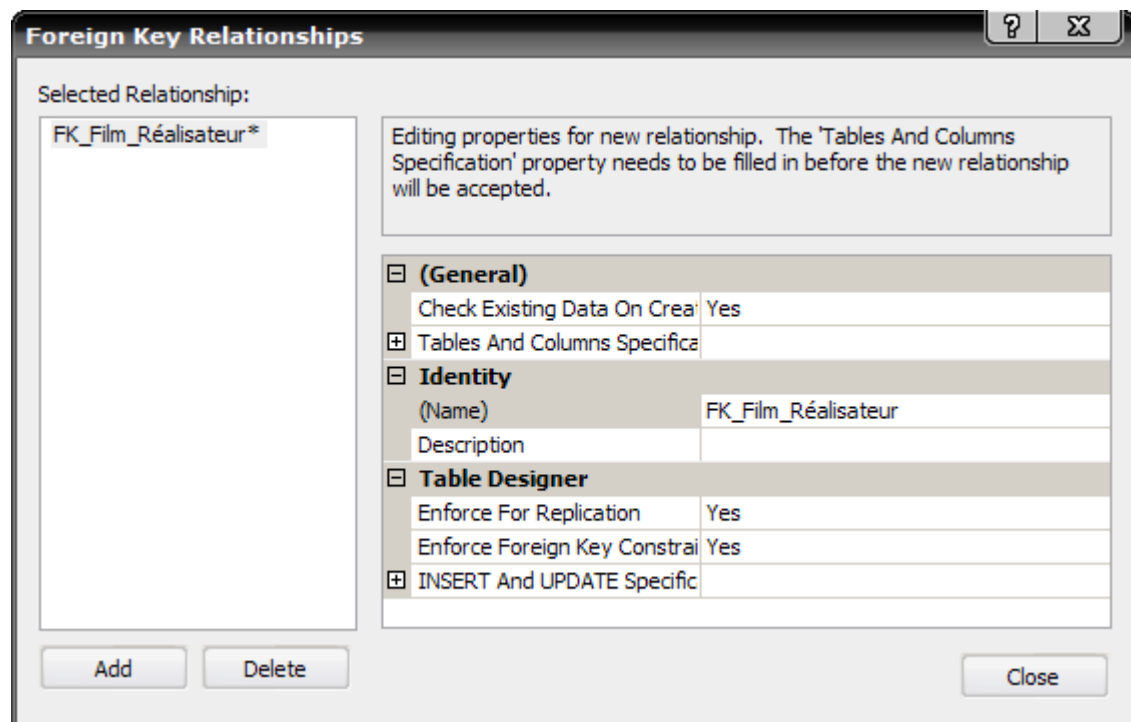
Nous savons que notre base contient trois tables, mais uniquement la table « Film » contient des clés étrangères.

Nous allons maintenant voir comment créer une clé étrangère dans Microsoft SQL Server.

Commençons par un clic droit sur « Keys » dans la fenêtre « Explorer » puis on clique sur « New Foreign Key »



Dans la nouvelle fenêtre qui s'ouvre, on va commencer par donner un nom à cette clé étrangère « FK\_Film\_Réalisateur »



Ensuite, on clique sur « Tables And Columns Specification » et une nouvelle fenêtre s'ouvre dans laquelle on va définir la clé étrangère et la clé primaire puis on clique sur « OK »

**Tables and Columns**

Relationship name:

Primary key table:

Foreign key table:

Id\_réalisateur

OK Cancel


Lorsqu'on revient à la fenêtre suivante, on peut modifier et mettre « en cascade » sur les relations entre la clé étrangère et la clé primaire pour la suppression et les mises à jour puis on clique sur « Close ».

☒ INSERT And UPDATE Specific

Delete Rule	Cascade
Update Rule	Cascade

On ferme cette table, un message pour l'enregistrement s'affiche, on clique alors sur « Yes » et une autre fenêtre s'affiche pour nous informer qu'il y a eu des changements dans deux tables de notre base. On clique sur « Yes » pour enregistrer.

**Save**

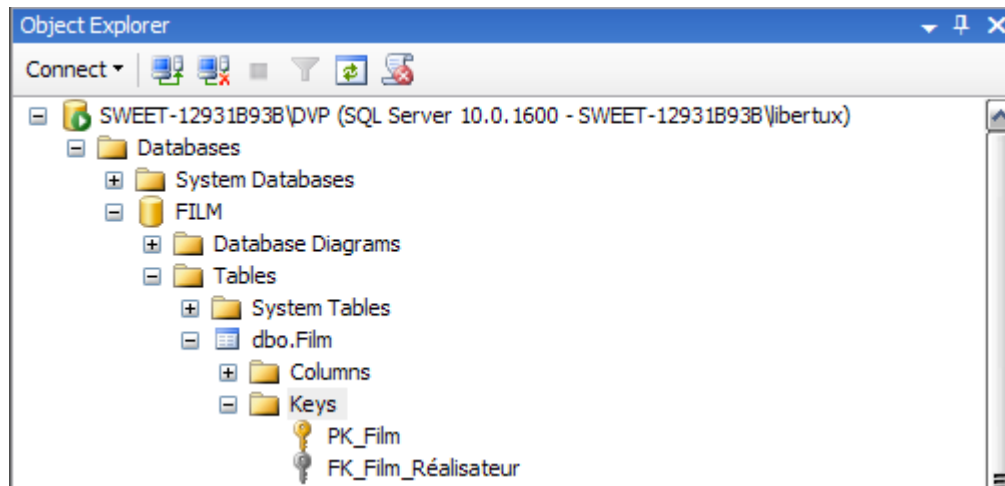
 The following tables will be saved to your database. Do you want to continue?

Réalisateur  
 Film

☒ Warn about Tables Affected

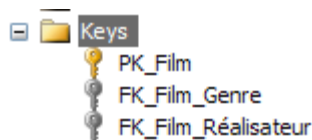
Yes No Save Text File

On peut vérifier maintenant, dans la fenêtre « Explorer », l'existence de notre clé étrangère.



Faisons à nouveau ce même travail pour créer notre deuxième clé étrangère la clé « Id\_Film\_Genre ».

Notre table doit contenir à la fin une clé primaire et deux clés étrangères.

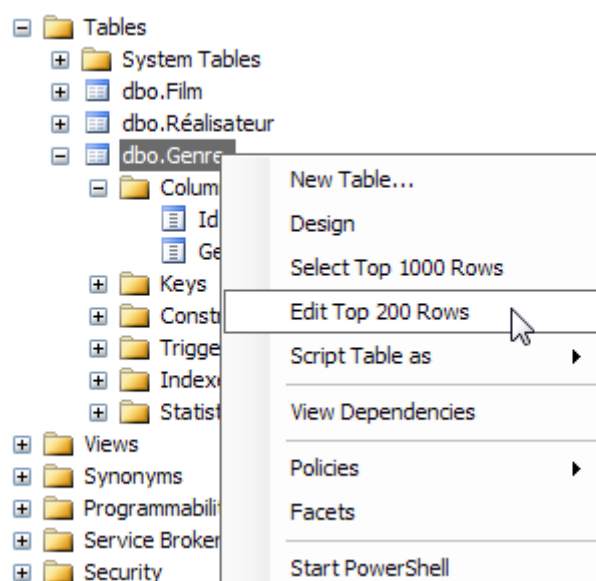


On peut dire maintenant que notre base est complète et il ne reste qu'à saisir des données dans cette base.

## 5- Saisir les données

Commençons par saisir les données dans la table genre.

Un clic droit sur la table « Genre » puis on clique sur « Edit Top 200 Rows ».



La table va s'ouvrir à droite et on saisit les données comme dans l'exemple ci-dessous

SWEET-1293189...M - dbo.Genre		
	Id_genre	Genre
	1	Action
	2	Comédie
	3	Romantique
	4	Dramatique

Notre base va comporter quatre genres de films, à savoir : Action, Comédie, Romantique et Dramatique.

Les numéros de 1 à 4 sont les identificateurs de chaque genre.

Maintenant comme la table de genre est créée on va passer à la table de réalisateur.

Pour cela, effectuons le même travail. On saisit donc les données pour que la table se présente de la façon suivante.

	Id_réalisateur	Nom	Prénom
	1	Spielberg	Steven
	2	Woo	John
	3	Scott	Ridley
	4	Fuqua	Antoine
	5	Mc Tiernan	John
	6	De Palma	Brian
	7	Abrams	Jeffrey Jacob
	8	Whitesell	John
	9	Jaoui	Agnès
	10	Fridberg	Rick
	11	Dugan	Denis
	12	Segal	Peter
	13	Shadyac	Tom
	14	Stiller	Ben
	15	Tennant	Andy
	16	Wright	Joe
	17	Rosman	Mark
	18	Mc Namara	Sean
	19	Greene	David
	20	Turteltaub	John
	21	Chadwick	Justin
	22	Muccino	Gabriele

Ainsi, la liste des réalisateurs est prête.

Pour remplir la table « Film », il faut seulement préciser l'identificateur du genre et celui de réalisateur.

Suivons les mêmes étapes pour saisir des données dans la table « Film » pour qu'elle s'affiche de la façon suivante


	Id_film	Id_réalisateur	Id_genre	Titre
	1	3	1	Gladiator
	2	4	1	Le roi Arthur
	3	15	3	A tout jamais
	4	19	4	Rendez moi mes enfants
	5	22	4	A la recherche du bonheur
	6	3	1	La chute du faucon noir
	7	8	2	Big Mamma 2
	8	11	2	Quand Chuck rencontre Larry
	9	14	2	Disjoncté
	10	6	1	Mission impossible 1
	11	2	1	Mission impossible 2
	12	7	1	Mission impossible 3
	13	16	3	Orgueil et préjugés
	14	20	4	Phénomène
	15	10	2	Agent zéro zéro
	16	13	2	Menteur menteur
	17	1	1	Les dents de la mer
	18	2	1	Chasse à l'homme
	19	18	3	Trouve ta voix
	20	1	1	Il faut sauver le soldat Rayan
	21	17	3	Comme Cendrillon
	22	9	2	Parlez-moi de la pluie
	23	5	1	Le 13ème guerrier
	24	21	4	Deux soeur pour un roi
	25	12	2	Mi-temps au mitard
	26	2	1	Les messagers du vent
	27	17	3	L'homme parfait
	28	1	1	Jurassic Park

## 6- Manipulation des données


Voyons maintenant comment nous pouvons utiliser les requêtes sur Microsoft SQL Server et exécuter quelques exemples de ces requêtes.

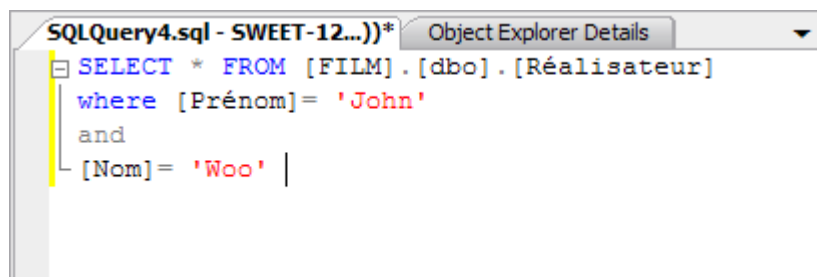
On va essayer d'afficher les films réalisés par l'un des réalisateurs de notre table.

D'abord il faut trouver l'identifiant du réalisateur dans la table «Réalisateur », par exemple le réalisateur « John Woo ».

On clique sur  New Query , une nouvelle fenêtre s'ouvre à droite et on saisit la requête suivante :

```
SELECT * FROM [FILM].[dbo].[Réalisateur] WHERE [Prénom]='John' and [Nom]='Woo'
```

On clique alors sur  Execute

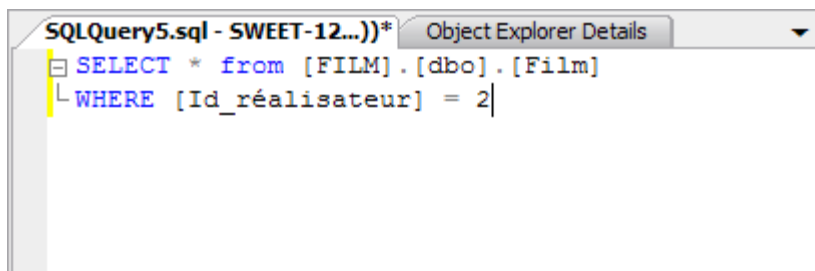


En exécutant, le résultat va s'afficher en bas

Results			
	Id_réalisateur	Nom	Prénom
1	2	Woo	John

Nous connaissons maintenant l'identifiant de ce réalisateur, ce qui nous permet de chercher tous ses films dans la table « Film » par la requête suivante :

```
SELECT * FROM [FILM].[dbo].[Film] WHERE [Id_réalisateur]=2
```



Le résultat suivant s'affiche

Results				
	Id_film	Id_réalisateur	Id_genre	Titre
1	11	2	1	Mission impossible 2
2	18	2	1	Chasse à l'homme
3	26	2	1	Les messagers du vent

**EXERCICE :** (Cet exercice fait appel à la recherche de la part de l'apprenant)

- 1) Expliquer cette requête ci-dessous :

```
SELECT * FROM [FILM].[dbo].[Réalisateur] WHERE [Prénom]='John' and [Nom]='Woo'
```

- 2) Expliquer cette requête ci-dessous :

```
SELECT * FROM [FILM].[dbo].[Film] WHERE [Id_réalisateur]=2
```

- 3) Comment accède t-on à une base de données sous MS SQL Server ?
- 4) Comment accède t-on à une table sous MS SQL Server ?