



Module 1 Développer l'interface d'une application informatique

Composants accès données	Séance	S01	Activité	A-003
--------------------------	--------	-----	----------	-------

Cette activité d'apprentissage vous fera découvrir les classes du framework d'accès aux données de l'architecture logicielle .Net désigné sous l'acronyme ADO.Net.
Dans ce troisième atelier vous apprendrez à mettre en place des mécanismes de gestion de transactions complexes.

Sommaire de l'activité proposée :

1	Présentation.....	2
2	Préparation de l'environnement	2
3	Travail à réaliser	3
3.1	Transaction explicite de type Mère-Fille.....	4

1 Présentation

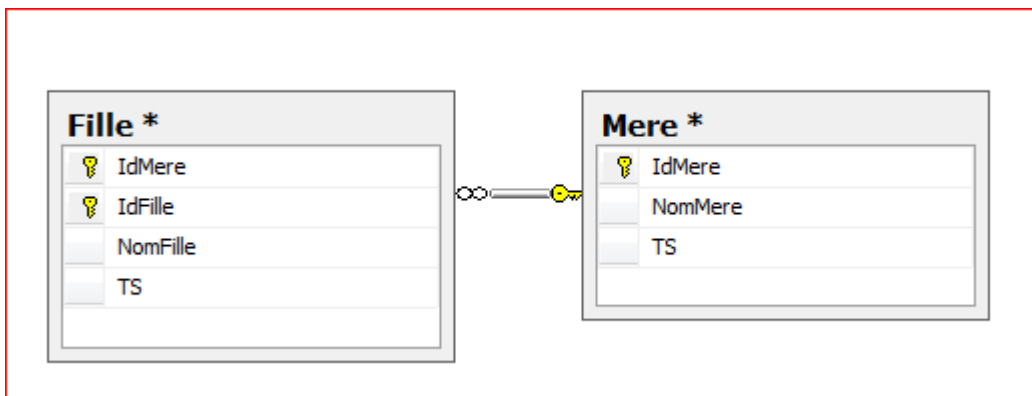
Au cours de ce troisième atelier, vous allez approfondir les techniques de programmation de composants d'accès aux données en mode connecté.

Ici, l'objectif est de comprendre comment mettre en place une transaction complexe portant sur plusieurs tables.

L'application cliente est une application Windows.

Les éléments manipulés sont d'une extrême simplicité comme en témoigne le schéma ci-dessous.

- Une table parente intitulée Mère
- Une table dépendante (enfant) dénommée Fille.



Les règles qui régissent les relations entre ces deux tables sont les suivantes :

- 1 ligne de la table Fille doit systématiquement référencer une seule ligne de la table Mere via la colonne IdMere.
- 1 ligne de la table Mere peut être en relation avec 0, 1 ou plusieurs lignes de la table Fille

Les clés primaires :

- La clé primaire de la table Mere est générée par le système (colonne identité)
- La clé primaire de la table Fille est constituée des colonnes IdMere et IdFille. La valeur IdMere est fournie par la table Mere (clé étrangère). La valeur de la colonne IdFille est une valeur numérique fournie.

Vous utiliserez le fournisseur de données propre à SQL Server tout au long des étapes de l'atelier System.Data.SqlClient.

2 Préparation de l'environnement

Préparation de l'environnement :

- Créer une nouvelle base de données ADO_Net
- Exécutez le script de création des tables et procédures stockées.
- Téléchargez et installez le squelette de l'application.
- Prenez connaissance de l'application et des procédures stockées fournies

3 Travail à réaliser

Compléter les 3 formulaires communiqués.

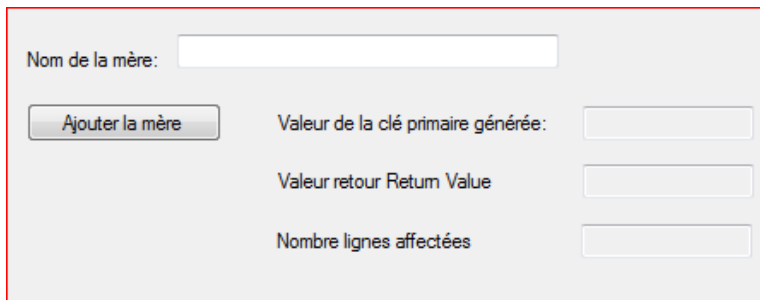
Les paramètres des objets de type Command sont obtenus par interrogation du serveur SGBDR à l'aide de l'objet **SQLCommandBuilder** et sa méthode statique **DeriveParameters**. Comme lors de l'exercice précédent, la valeur de la direction des paramètres exclusivement en sortie doit être modifiée.

A noter :

Afin d'assurer une maintenance aisée de l'application et de bonnes performances à celle-ci, il convient de dissocier le code permettant de créer les objets ADO de celui nécessaire à l'exécution de ceux-ci.

Utiliser comme base le squelette de la classe statique Transactions.

1er formulaire :



Création d'une nouvelle ligne dans la table Mere.

L'objectif est ici de savoir correctement récupérer l'identifiant affecté à une clé par le système, la valeur retour et le nombre de lignes affectées.

Le paramètre particulier permettant de récupérer la valeur de **retour** d'une procédure stockée est de type **ParameterDirection.ReturnValue** et est toujours le premier paramètre de la collection des paramètres (à l'index 0).

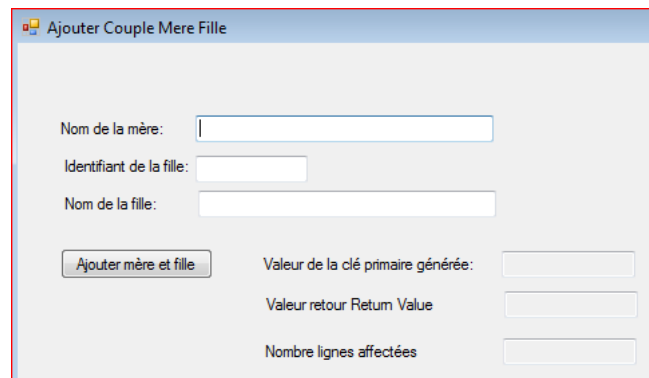
Deux points à souligner au niveau de la procédure stockée d'insertion :

L'utilisation d'une colonne de type **RowVersion (alias TimeStamp)** dont la valeur change à chaque modification (insert, update). Ce mécanisme est utile pour vérifier, avant de mettre à jour une ligne, qu'elle n'a pas été modifiée par une transaction concurrente. Le type **Rowversion** est dérivé du type **Binary** et à une taille de 8 octets.

Le recours à la méthode **Scope_Identity()** pour récupérer la valeur de la colonne clé primaire Identity sur insertion et la récupération de la valeur de la version de ligne.

*En conformité avec la nouvelle norme SQL, le type **TimeStamp** doit être remplacé par **RowVersion**.*

2^{ème} formulaire :



Dans ce deuxième formulaire, vous devez programmer l'ajout d'un enregistrement enfant dans le même temps que l'enregistrement parent auquel il est associé par la clé étrangère.

Vous vérifierez ici l'intérêt de la récupération de la valeur de la clé primaire générée pour la ligne parente IdMere.

Cette valeur devra être communiquée en argument de l'insertion d'une ligne enfant.

3.1 Transaction explicite de type Mère-Fille

Dans ce troisième exemple, vous allez mettre en place une transaction explicite afin de vous assurer que les parents ne perdent pas d'enfants en chemin où, qu'en cas d'accident il n'y ait aucun survivant...

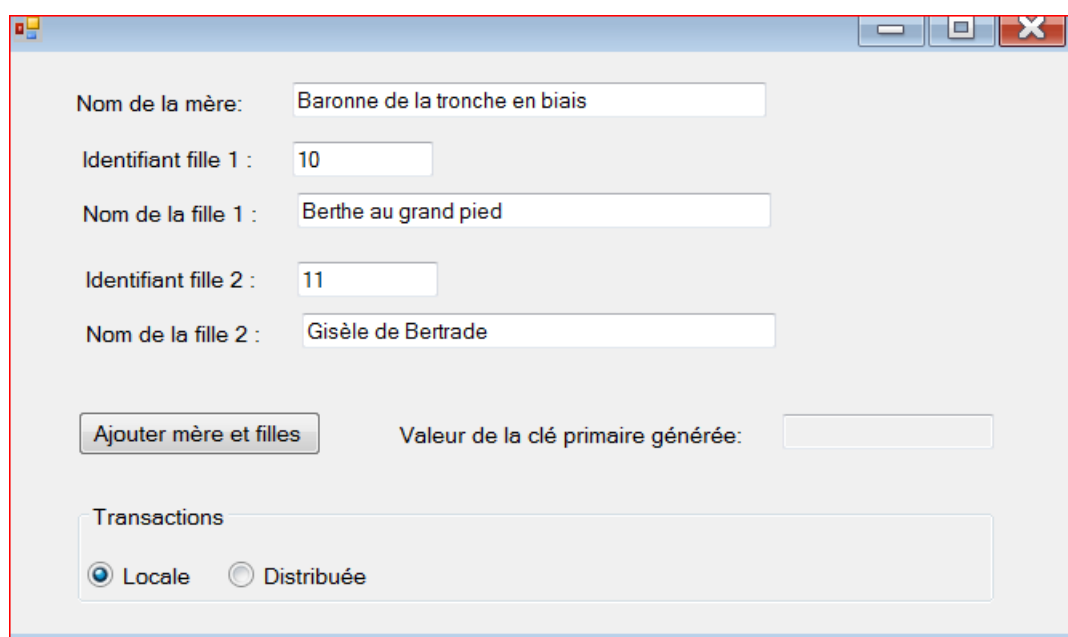
Deux approches sont possibles :

Une première approche consiste à utiliser un objet de type transaction locale et à associer celui-ci aux objets connexion et commandes.

Une deuxième approche, étendue, permet de définir une portée transactionnelle à l'aide de TransactionScope. Cette approche permet de prendre en charge des transactions distribuées sur plusieurs serveurs.

A noter : Il vous faut ajouter la dll **system.transaction**

Créer une procédure pour chaque modèle de transaction.



The screenshot shows a Windows application window with a light gray background. It contains several text input fields and a button. The fields are labeled as follows:

- Nom de la mère:
- Identifiant fille 1 :
- Nom de la fille 1 :
- Identifiant fille 2 :
- Nom de la fille 2 :

Below these fields is a button labeled "Ajouter mère et filles". To the right of the button is a label "Valeur de la clé primaire générée:" followed by an empty text box.

At the bottom, there is a section titled "Transactions" containing two radio buttons: "Locale" (which is selected) and "Distribuée".

Laissez le niveau d'isolation par défaut, **ReadCommitted**. Les processus qui souhaitent accéder en lecture aux lignes verrouillées par une transaction doivent attendre la fin de cette dernière.