Programmation de bases de données:
Procédures stockées

Plan de la séance:

- Retour sur la dernière séance:
  - Point de vue des enseignants.
  - Point de vue des étudiants.
- Procédures stockées
  - Définition
  - Avantages
  - Syntaxe
  - Exemples
    - Cas de paramètres en IN
    - Procédure SELECT
    - Cas des paramètres en OUT
  - Les fonctions stockées
    - Les fonctions scalaire
    - Les fonctions TABLE

Retour sur la séance précédente

- Point de vue des étudiants
- Point de vue des enseignants.

Procédures stockées, définition

Définition:

Une procédure stockée est un ensemble d'instructions SQL précompilées stockées dans le serveur de bases de données

Avantages:

- Rapidité d'exécution, puisque les procédures stockées sont déjà compilées.
- Réutilisation de la procédure stockée.
- Possibilité d'exécuter un ensemble de requêtes SQL
- Modularité. Facilite le travail d'équipe.
- Clarté du code
- Prévention d'injection SQL

Exemple

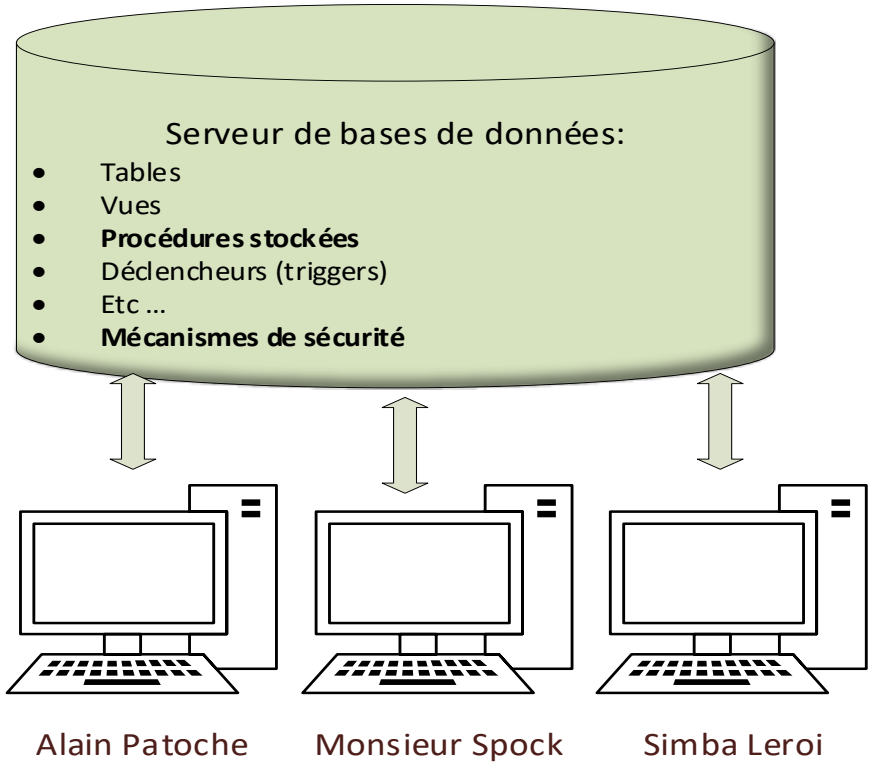
Exemple: Quel code est plus claire ?

```
SqlCommand objCommand = new SqlCommand("select nom, prenom,salaire, nomequipe from joueurs " +
    "inner join equipes on joueurs.CODEEQUIPE = equipes.CODEEQUIPE ", nomConnection);
```

Ou le code suivant:

```
SqlCommand objCommand1 = new SqlCommand("afficherJoueurs", nomConnection);
```

Exemple



- Les procédures stockées sont des objets de la BD comme les tables, ou les view.
- Une procédure stockée se crée avec la commande CREATE

Procédures stockées, syntaxe simplifiée

```
CREATE [ OR ALTER ] { PROC | PROCEDURE }
    [schema_name.] procedure_name
    [ { @parameter  data_type }
        [ OUT | OUTPUT ]
AS
{ [ BEGIN ] sql_statement [;] [ ...n ] [ END ] }
[;]
```

Procédures stockées, syntaxe simplifiée

CREATE PROCEDURE : indique que l'on veut créer une procédure stockée.

OR ALTER est optionnel, indique que l'on veut modifier la procédure stockée si celle-ci existe déjà.

@parameter data_type : On doit fournir la liste des paramètres de la procédure avec le type de données correspondant à chacun des paramètres.

[OUT | OUTPUT] : Indique la direction en OUT ou en OUTPUT des paramètres de la procédure. Par défaut les paramètres sont en IN.

AS : mot réservé qui annonce le début du corps de la procédure et la fin de la déclaration des paramètres

BEGIN

Bloc SQL ou Transact-SQL

END;

Procédures stockées: Les paramètres sont en IN

Exemple1: Les paramètres sont en IN, **INSERTION**

```
CREATE PROCEDURE insertionEtudiants(
                        @pnom VARCHAR(20),
                        @pprenom VARCHAR(30),
                        @psal MONEY,
                        @pcodep CHAR(3)
                )
AS
BEGIN
  INSERT INTO etudiants(nom , prenom ,salaire ,codep )
 VALUES(@pnom , @pprenom ,@psal ,@pcodep)
END;
```

Procédures stockées: Les paramètres sont en IN

Exécution d'une procédure dans son SGBD natif (MS SQL Server)

Pour exécuter une procédure stockée, on utilise les commandes EXECUTE ou EXEC. Il faudra fournir la valeur des paramètres.

Exemple 1:

```
EXECUTE insertionEtudiants
@pnom ='Lenouveau',
@pprenom ='lenouveau',
@psal=22.5,
@pcodep ='sim';
```

Même s'il est conseillé de passer les paramètres dans l'ordre de leur apparition dans la procédure, MS SQL Server peut accepter la passation des paramètres dans n'importe quel ordre

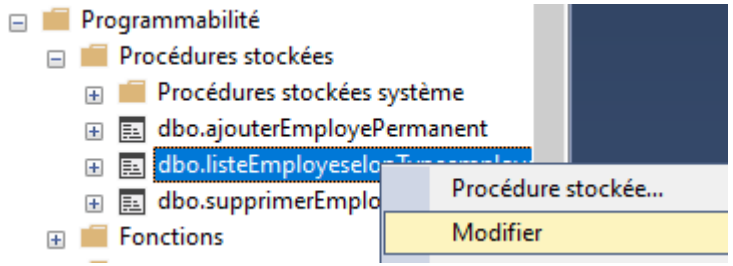
Procédures stockées: Les paramètres sont en IN

On aurait pu faire ceci (les paramètres ne sont pas passés dans l'ordre de la procédure).

```
EXECUTE insertionEtudiants
@pprenom ='Alain',
@psal=22.5,
@pcodep ='sim',
@pnom ='Patoche';
```

Mais n'oubliez pas, que c'est vous qui allez déboguer une procédure qui ne marche pas. Transmettre les paramètres dans l'ordre de la procédure est ce qui est recommandé.

Procédure stockée, par l'interface graphique



Après Modifier

```
9
10  ALTER procedure [dbo].[listeEmployeselonTypeemploye]
11  begin
```

- Vos procédures stockées se trouvent à l'onglet Programmabilié
- Pour modifier une procédure stockée soit:
  - On fait un DROP et on la recrée avec CREATE
  - On fait ALTER Procedure le nom de la procédure ..
  - Le plus simple est de faire bouton droit sur la procédure puis modifier. Vous avez le code de la procédure en ALTER

Procédures stockées: Les paramètres sont en IN

Exemple2: Les paramètres sont en IN, **SELECTION**

```
CREATE PROCEDURE lister (@pcodep CHAR(3))
AS
BEGIN
SELECT nom,prenom from etudiants WHERE codep =@pcodep ;
END;
```

Execution:

```
EXECUTE lister
@pcodep='inf'
```

Procédures stockées: Les paramètres sont en IN

Exemple3: Les paramètres sont en IN, **SELECTION**

```
CREATE PROCEDURE ChercherNom (@pnom VARCHAR (20))
AS
BEGIN
SELECT * FROM  etudiants WHERE nom Like '%'+ @pnom +'%';
END;
```

Execution

```
EXECUTE ChercherNom
@pnom='Le'
```

Procédures stockées: Les paramètres sont en OUT

Exemple4: Un paramètre en OUT

```
create procedure chercherNom(
                                @pnum int,
                                @pnom varchar(20) OUT) AS
begin
 select @pnom = nom from etudiants where numad=@pnum;
end;
```

Exécution:

```
declare @pnum int =1;
declare @pnom varchar(20);
execute chercherNom
@pnum ,
@pnom output;
 print @pnom
```

Programmation de bases de données:
Procédures stockées (suite)

Plan de la séance:

- Retour sur la dernière séance:
  - Point de vue des enseignants.
  - Point de vue des étudiants.
- Rappels
- Fonctions stockées
  - Fonction scalaire
  - Fonction table
- Exemples
- Détruire une procédure stockée
- Points clés

Procédures stockées– Les fonctions

Les fonctions stockées:

Les fonctions stockées sont des procédures stockées qui retournent des valeurs. Leurs définitions sont légèrement différentes d'une procédure stockée mais le principe général de définition reste le même.

Il existe deux types de fonctions:

- Celles qui retourne un type scalaire (associé à une valeur unique: int, char, varchar..)
- Celles qui retournent une TABLE

La syntaxe pour l'écriture de ces deux types de fonctions n'est pas la même, et ne s'exécutent pas de la même façon.

Procédures stockées, fonctions scalaires

1. Fonction scalaire

```
CREATE [ OR ALTER ] FUNCTION [ schema_name. ] function_name
( [ { @parameter_name  parameter_data_type     }   ]  )
RETURNS return_data_type

   [ AS ]
   BEGIN
      function_body
      RETURN scalar_expression
   END
[ ; ]
```

Procédures stockées: fonctions scalaires

Exemple

```
CREATE FUNCTION compteretudiants(@pcode CHAR(3)) RETURNS INT
AS
BEGIN
DECLARE @total INT;
 SELECT  @total = COUNT(*) FROM  Etudiants WHERE codep =@pcode;
 RETURN @total;
END;
```

Procédures stockées: fonctions scalaires

Les fonctions retournent un résultat, la manière d'obtenir le résultat est d'utiliser une requête SELECT.

SELECT nomFonction (liste de valeurs pour les paramètres).

Important: pour l'exécution des fonction, nous avons besoin de préciser le schéma (nom schema.nom de l'objet). Pour l'instant, le nom du schéma est dbo(Data Base Owner)

Donc:

Pour afficher les résultat de la fonction précédente :

SELECT **dbo**.compteretudiants('inf');

- Remarquez le dbo
- Il n' y a pas de from (pour Oracle from DUAL)

Procédures stockées, les fonctions TABLE

2- Fonction TABLE.

```
CREATE [ OR ALTER ] FUNCTION [ schema_name. ] function_name
( [ { @parameter_name  parameter_data_type    }   ]  )
RETURNS TABLE

   [ AS ]
   RETURN [ ( ] select_stmt [ ) ]
[ ; ]
```

Il n’ y pas de BEGIN --END

Procédures stockées, les fonctions TABLE

Exemple:

```
CREATE FUNCTION ChercherEMPLOYE (@code CHAR(3)) RETURNS TABLE
AS
        RETURN
    (
                SELECT nom,prenom,salaire
                  FROM employes
                 WHERE @code  =CodeDep
    ) ;
```

Exécution:

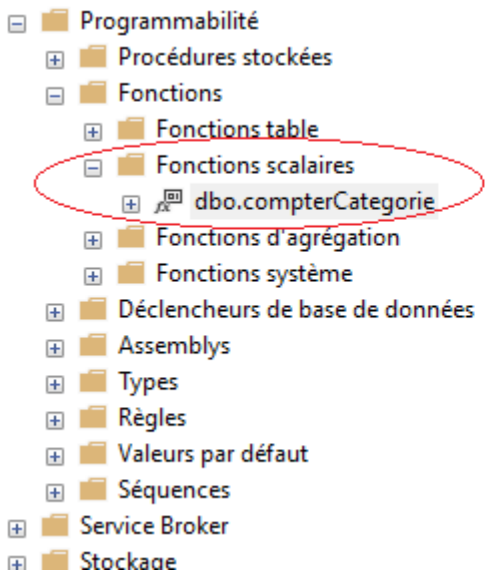
```
SELECT * FROM ChercherEMPLOYE('inf');
```

La fonction TABLE se comporte comme une table.

Procédures stockées, les fonctions

Les fonctions stockées sont aussi dans l'onglet programmabilité. Vous remarquerez que vous avez les deux types de fonctions: TABLE et SCALAIRE.

Vous pouvez également modifier une fonction par le bouton droit, puis modifier → vous obtiendrez le code de la fonction en ALTER FUNCTION




```
9
10
11
12  ALTER function [dbo].[compterCategorie]()
13  begin
14  declare
15  @totalCategorie int
16  select @totalCategorie = count(*) from Cat
17  return @totalCategorie;
18  end;
```

Procédures stockées, destruction

Pour détruire une procédure ou une fonction :

DROP PROCEDURE nomProcedure

DROP FUNCTION nomFonction

Pour modifier une procédure ou une fonction.

ALTER PROCEDURE nomProcedure

ALTER FUNCTION nomFonction

Points clés:

- Dans la littérature des BD, le terme « procédures stockées » englobe les procédures et les fonctions.
- Pour les procédures et les fonctions les paramètres sont précédés de @
- Le type IN est par défaut. L'indiquer provoque une erreur
- Lorsque le paramètre est en OUT ou OUTPUT, il faudra l'indiquer clairement.
- Les procédures et fonctions sont terminées par **GO**. Il n'est cependant pas obligatoire.
- Le mot réservé DECLARE est obligatoire pour déclarer des variables.
- Les fonctions peuvent retourner des tables. Elles NE comportent PAS les mots réservés BEGIN et END
- Pour exécuter une procédure il faut utiliser execute ou exec
- Pour exécuter une fonction scalaire il faut utiliser select dbo.nomfonction (valeurs paramètres)
- Pour exécuter une fonction table c'est SELECT …. FROM nomFonction (valeurs paramètres)
- Vos fonctions et procédures se trouvent à Programmabilité de la BD
- Vous pouvez modifier les procédures/fonctions par le bouton modifier de votre SSMS

CONCLUSION

QUESTIONS ??