Programmation Transact SQL

Procédure

Ecrire une procédure qui permet de rendre un film emprunté. Cette procédure admet en paramètre le numéro d'exemplaire du film emprunté et se charge de compléter la table des détails afin d'enregistrer le retour à la date et heure courante.

La procédure porte le nom de restituer et le paramètre sera nommé numeroExemplaire.

La procédure peut être créée de la façon suivante:

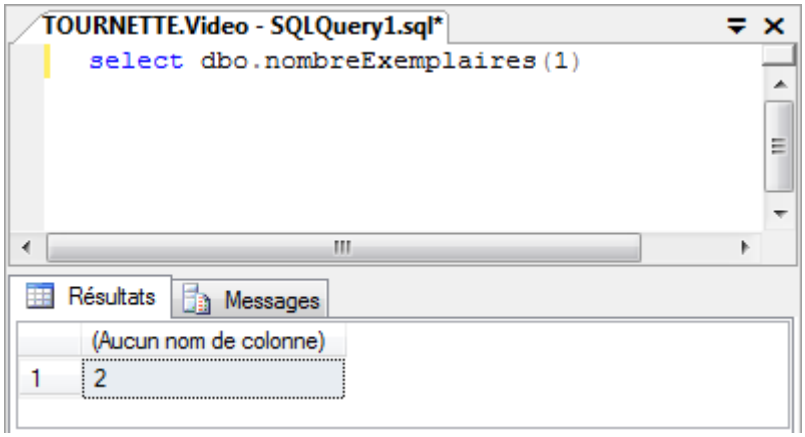
```
USE video;
go
CREATE PROCEDURE restituer(@numeroExemplaire int) AS
BEGIN
  UPDATE Magasin.Details
      SET retourEffectifLe=getdate()
      WHERE retourEffectifLe is null
      AND exemplaire=@numeroExemplaire;
END;
```

Fonction qui retourne une valeur scalaire

Ecrire une fonction qui accepte en paramètre un numéro de film et qui retourne le nombre d'exemplaires de ce film.

Cette fonction portera le nom de nombreExemplaires et le paramètre sera nommé numeroFilm.

Exemple d'utilisation de la fonction:



La fonction peut être créée de la façon suivante

```
USE video;
go
CREATE FUNCTION nombreExemplaires (@numeroFilm int) RETURNS int AS
BEGIN
      DECLARE @nombre int;
```

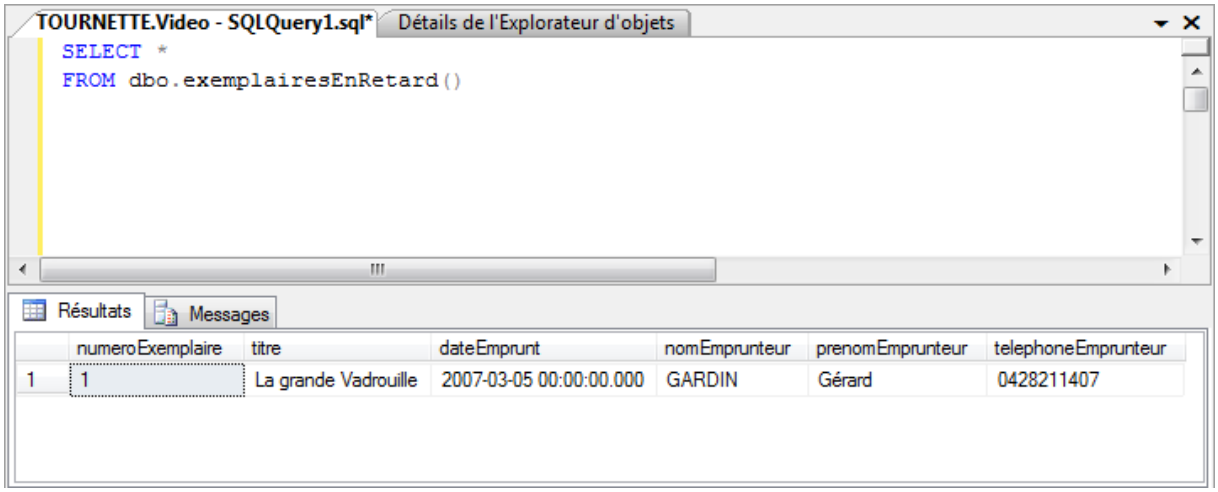
```
        SELECT @nombre=count(*)
        FROM Magasin.Exemplaires
        WHERE film=@numeroFilm;

        RETURN @nombre;

END;
```

Fonction qui retourne une table

Ecrire une fonction qui retourne la liste de tous les exemplaires qui devrait être rendu et qui ne le sont toujours pas. En face du numéro de l'exemplaire, il est nécessaire de trouver le titre du film, la date d'emprunt ainsi que les coordonnées de l'emprunteur (nom, prénom, téléphone) du film emprunté.

La fonction portera le nom **exemplairesEnRetard**.

Exemple d'utilisation de la fonction:



	numeroExemplaire	titre	dateEmprunt	nomEmprunteur	prenomEmprunteur	telephoneEmprunteur
1	1	La grande Vadrouille	2007-03-05 00:00:00.000	GARDIN	Gérard	0428211407

Note: La ligne n°3 concernant l'exemplaire n°1 à était ajoutée sur la fiche n°4 afin de pouvoir tester la fonction en cours de création

Cette fonction peut être définie à l'aide du script suivant:

```
USE video;
go
CREATE FUNCTION exemplairesEnRetard() RETURNS @detailExemplaire
      TABLE (numeroExemplaire int,
                  titre nvarchar(80),
                  dateEmprunt datetime,
                  nomEmprunteur nvarchar(50),
                  prenomEmprunteur nvarchar(50),
                  telephoneEmprunteur char(14))AS
BEGIN
      INSERT INTO @detailExemplaire(numeroExemplaire, titre, dateEmprunt,
nomEmprunteur, prenomEmprunteur, telephoneEmprunteur)
      SELECT md.exemplaire, f.titre, mf.creele, nom, prenom, telephone
        FROM Magasin.Details md
      INNER JOIN Magasin.Exemplaires e ON e.numero=md.exemplaire
      INNER JOIN Films f ON e.film=f.numero
```

```
      INNER JOIN Magasin.Fiches mf ON md.fiche=mf.numero
      INNER JOIN Magasin.Clients mc ON mf.client=mc.numero
       WHERE retourEffectifLe is null
         AND retourPrevuLe<getdate();
      RETURN;
END;
```

Etant donné que cette fonction ne contient qu'une seule instruction, il est possible de la définir de la façon suivante:

```
USE video;
go
CREATE FUNCTION exemplairesEnRetard() RETURNS TABLE AS
      RETURN SELECT numeroExemplaire=md.exemplaire, f.titre,
                    dateEmprunt=mf.creele,
                    nomEmprunteur=nom,
                    prenomEmprunteur=prenom,
                    telephoneEmprunteur=telephone
        FROM Magasin.Details md
      INNER JOIN Magasin.Exemplaires e ON e.numero=md.exemplaire
      INNER JOIN Films f ON e.film=f.numero
      INNER JOIN Magasin.Fiches mf ON md.fiche=mf.numero
      INNER JOIN Magasin.Clients mc ON mf.client=mc.numero
       WHERE retourEffectifLe is null
         AND retourPrevuLe<getdate();
```

Trigger

Calculer automatiquement le montant dû pour chaque film emprunté

Ajouter la colonne montant au niveau de la table Details et valoriser cette colonne lorsque l'exemplaire est effectivement rendu. Compte tenu du fait que cette colonne va stocker une valeur monétaire c'est le type money qui sera choisi.

La modification de la structure de la table Magasin.Details peut être réalisée à l'aide du script suivant:

```
USE video;
go
ALTER TABLE Magasin.Details
  ADD montant money null;
```

Par la suite le déclencheur de base de données doit être défini sur la table Magasin.Details et doit s'exécuter lors de la valorisation de la colonne retourEffectifLe. Cette valorisation à lieu au cours de la mise à jour de la ligne c'est dire par l'exécution de l'instruction UPDATE.

```
USE video;
go
CREATE TRIGGER Magasin.DetailsMontant
  ON Magasin.Details
  AFTER UPDATE AS
  -- S'assurer que la mise à jour porte sur la date de retour
  IF (UPDATE (retourEffectifLe))
  BEGIN
      -- Curseur pour traiter toutes les lignes concernées par la mise à
jour
```

```
    DECLARE cApres CURSOR FOR SELECT numero, fiche,exemplaire,
retourEffectifLe FROM inserted;
      DECLARE @numero INT;
      DECLARE @fiche INT;
      DECLARE @exemplaire INT;
      DECLARE @retourEffectifLe DATETIME;
      DECLARE @valeurInitialeRetour DATETIME;
      OPEN cApres;
      -- Obtenir les informations sur la première ligne
      FETCH NEXT FROM cApres INTO @numero, @fiche, @exemplaire,
@retourEffectifLe;
      WHILE (@@FETCH_STATUS=0)
      BEGIN
        -- interdire la modification de la date de retour si elle est déjà
valorisée
        SELECT @valeurInitialeRetour=retourEffectifLe
            FROM deleted
            WHERE numero=@numero AND fiche=@fiche;
        IF (@valeurInitialeRetour is not null )
            RAISERROR ('Le changement de date de retour est
interdit',16,1);
        -- Calcul du montant de la ligne
        DECLARE @montant MONEY;
        SELECT
@montant=(prixLocationJour*datediff(day,creeLe,@retourEffectifLe))
          FROM Magasin.Fiches f, Magasin.Exemplaires e
            INNER JOIN Films on Films.numero=e.film
        WHERE e.numero=@exemplaire
            AND f.numero=@fiche;
        UPDATE Magasin.Details SET montant=@montant
            WHERE numero=@numero AND fiche=@fiche;
        -- Traiter la ligne suivante de la table inserted
        FETCH NEXT FROM cApres INTO @numero, @fiche, @exemplaire,
@retourEffectifLe;
      END;
      CLOSE cApres;
      DEALLOCATE cApres;
  END;
```

Est-il possible d'utiliser les colonnes calculées?

SQL Server propose une fonctionnalité intéressante qui est celle des colonnes calculées. Lors de la définition de la colonne dans la table, il est possible de préciser la formule de calcul à utiliser. Toutefois les colonnes qui servent de base au calcul doivent être présentent dans la même table que la colonne calculée ce qui n'est pas le cas ici. Donc il n'est pas possible d'utiliser une colonne calculée.

Garantir que le montant payé est bien égal au montant dû

Toujours à l'aide d'un trigger de base de données mettez en place un contrôle automatique pour garantir que le montant payé (qui est enregistré au niveau de la fiche) correspond bien au montant dû. Il est bien sûr possible de régler une fiche uniquement si la totalité des films empruntés ont été restitués.

Avant d'écrire le trigger de base de données sur la table Magasin.Fiches, il est préférable d'écrire la fonction Magasin.Rendue qui accepte en paramètre un numéro de fiche et qui retourne la valeur 1 si toute les films présents dans la fiche sont restitué et 0 sinon.

```
USE video;
go
CREATE FUNCTION Magasin.Rendue(@numeroFiche int) RETURNS tinyint AS
BEGIN
      DECLARE @retour tinyint;
      DECLARE @nombre int;
      SELECT @nombre=count(*)
        FROM Magasin.Details
       WHERE fiche=@numeroFiche
         AND retourEffectifLe is null;
      IF (@nombre=0)
            SET @retour=1;
      ELSE
            SET @retour=0;
      RETURN @retour
END;
```

Ensuite la fonction Magasin.MontantFiche qui accepte en paramètre le numéro d'une fiche et qui retourne le montant à payer est définie:

```
USE video;
go
CREATE FUNCTION Magasin.MontantFiche(@numeroFiche int) RETURNS money AS
BEGIN
      -- Toutes les films ont ils bien restitués?
    IF (Magasin.rendue(@numeroFiche)!=1)
      BEGIN
            -- Tous les films ne sont pas encrore rendus
            RETURN 0;
      END;
      -- Calculer le montant de la fiche
      DECLARE @montant money;
      SELECT @montant=SUM(montant)
        FROM Magasin.Details
       WHERE fiche=@numeroFiche;
      RETURN @montant;
END;
```

Il est maintenant possible d'écrire le déclencheur. Ce déclencheur sera associé à l'instruction UPDATE sur la table Magasin.Fiches.

```
USE video;
go
CREATE TRIGGER Magasin.tr_upd_fiches
      ON Magasin.Fiches
      FOR UPDATE
      AS
IF UPDATE(montantPaye)
BEGIN
      -- Parcourir toutes les lignes mises à jour
      DECLARE  cNouveau CURSOR FOR SELECT numero, montantPaye FROM
inserted;
      DECLARE @numero int;
      DECLARE @montantPaye money;
      OPEN cNouveau;
      -- Travailler avec la première ligne
      FETCH cNouveau INTO @numero,@montantPaye;
      -- Tant qu'il reste des lignes
      WHILE(@@FETCH_STATUS=0)
      BEGIN
            IF (@montantPaye>Magasin.MontantFiche(@numero))
                  RAISERROR ('Montant trop important',15,1);
```

```
            -- Passer à l'enregistrement suivant
            FETCH cNouveau INTO @numero,@montantPaye;
      END;
      CLOSE cNouveau;
      DEALLOCATE cNouveau;
END;
```