M21 : SGBD 2

Transact-SQL(P.S & Trig)

Formateur : Driouch Bouazza
Etablissement : OFPPT/CFMOTI
09/11/2009 V1.0
http://www.driouchb.0fees.net/

Plan

Introduction - Définition
Types de données
Instruction (Déclaration, Affectation, Affichage)

Transact-SQL

Définition

Transact-SQL est une extension de SQL vers un langage de programmation, il est aussi le point central de l'utilisation de Microsoft SQL Server. Toutes les applications qui communiquent avec SQL Server le font en envoyant des instructions Transact-SQL au serveur, quelle que soit l'interface utilisateur de l'application.

En plus de ça, pour soulager les postes client, une partie des applications client on les programme au niveau Serveur (SGBD) avec plus de performance processeur et mémoire.

Transact-SQL

Types de Données

Données numériques entières	
Bit	Nombre entier dont la valeur est 1 ou 0. Peut aussi être null.
Int	Nombre entier dont la valeur est comprise entre -2^{31} (-2 147 483 648) et 2^{31} - 1 (2 147 483 647).
Smallint	Nombre entier dont la valeur est comprise entre -2^{15} (-32 768) et 2^{15} - 1 (32 767).
Tinyint	Nombre entier dont la valeur est comprise entre 0 et 255.
Decimal	Données numériques de précision et d'échelle fixes comprises entre -10^{38} -1 et 10^{38} -1.
Numeric	Synonyme de **decimal**.
Money	Valeurs de données monétaires comprises entre -2^{63} (-922 337 203 685 477,380 8) et 2^{63} - 1 (+922 337 203 685 477,580 7), avec une précision d'un dix-millième d'unité monétaire.
SmallMoney	Valeurs de données monétaires comprises entre - 214 748,3648 et +214 748,3647, avec une précision d'un dix-millième d'unité monétaire.
Données numériques approchées	
Float	Données numériques de précision en virgule flottante comprises entre -1.79E + 308 et 1.79E + 308.
Real	Données numériques de précision en virgule flottante comprises entre -3.40E + 38 et 3.40E + 38.
Datetime	Données de date et d'heure comprises entre le 1er janvier 1753 et le 31 décembre 9999, avec une précision de trois centièmes de seconde ou de 3,33 millisecondes.
Smalldatetime	Données de date et d'heure comprise entre le 1er janvier 1900 et le 6 juin 2079, avec une précision d'une minute.
Chaînes de caractères	
Char	Données non Unicode de longueur fixe d'un maximum de 8 000 caractères.
Varchar	Données non Unicode de longueur variable.
Text	Données non Unicode de longueur variable ne pouvant pas dépasser 2^{31} - 1 (2 147 483 647) caractères.
nchar	Données Unicode de longueur fixe ne pouvant pas dépasser 4 000 caractères.
nvarchar	Données Unicode de longueur variable ne pouvant pas dépasser 4 000 caractères.
Ntext	Données Unicode de longueur variable ne pouvant pas dépasser 2^{30} - 1 (1 073 741 823) caractères. Chaînes binaires.
Binary	Données binaires de longueur fixe.
Varbinary	Données binaires de longueur variable.
Image	Données binaires de longueur variable ne pouvant pas dépasser 2^{31} - 1 (2 147 483 647) octets.

Transact-SQL

Autre Types complexes

Table : Type de donn ées sp écial qui permet de stocker un jeu de r ésultats pour un traitement ultérieur. Son utilisation principale concerne le stockage temporaire d'un ensemble de lignes, qui doivent être renvoy ées sous forme de jeu de résultats d'une fonction table.

Une variable de type table se comporte comme une variable locale. Elle poss ède une port ée bien définie, qui repr ésente la fonction, la proc édure stockée ou le lot d'instructions dans lequel elle est déclarée.

Transact-SQL

Déclaration des variables locales

Synataxe génarale : Declare @nom_variable type
Le caractère @ est obligatoire
Exemple : Declare @maVariable int
Déclaration multiple :
Declare @var1 type1, @var2 type2
Exemple :Declare @x int, @y int, @z char
NB : Declare @x ,@y int est incorrecte

Affectation

Syntaxe générale : Select @variable=Valeur où set @variable=valeur
Exemple :select @i=3
set @j=4
select @str='TSDI'
affectation multiple
select @i=3,@j=4,@str='TSDI' est correcte, mais par contre set @i=3, @j=4, @str='TSDI' est une affectation incorrecte.

Transact-SQL

Affichage des valeurs

Pour afficher le contenu d'une variable on utilise la même instruction select.

Select @i

Affichage multiple : Select @i,@j,@str

NB:On peut utiliser select pour affecter une valeur ou bien pour afficher une autre, mais pas pour faire les deux, donc linstruction select @i=20, @str est incorrecte.

Exemple de variable de type table :

```
Declare @stg table(numInsc int primary key, nom varchar(20), prenom
    varchar(20),moyenne numeric(4,2))
/*la particularité des variables de type table, est qu 'on peut utiliser des
    commandes insert, select,update, delete */
insert into @stg values(103,'LAAROUSSI','SALAH',14)
insert into @stg values(107,'AADISSA','Youness',14.5)
insert into @stg values(200,'SOQRAT','Sanaa',12.5)
select * from @stg
```

	numInsc	nom	prenom	moyenne
1	103	LAAROUSSI	SALAH	14.00
2	107	AADISSA	Youness	14.50
3	200	SOQRAT	Sanaa	12.50

Transact-SQL

Les variables globales

Les variables globales sont affectées directement par le serveur, elle retournent une seul valeur,elle sont utilisées pour communiquer une information au client, elle sont notées @@nom_variable

Exemple :

@@error :indique le type d'erreur survenu lors de la dernère instruction.

@@rowcount : indique le nombre de lignes affectées par la dernière instruction.

@@identity :

Bloc d'instructions

Un bloc d'instruction est une ensemble d'instruction T-SQL qui sont considéré comme un tout (une seule).

Un bloc d'instruction peut contenir d'autres sous blocs.

Pour déclarer un bloc d'instructions en T-SQL en utilise :

Transact-SQL

Bloc d'instructions

```
Begin
    --instruction(1)
    --instruction(2)
    …
    --instruction(N)
end
example:
declare @i int,@j int
begin
    set @i=2
    set @j=3
    select @i=@i+1
    set @j=@j-1
    select @i as 'i', @j as 'j'
end
```

Transact-SQL

Structure alternative

La structure alternative est une structure de contrôle qui permet d'exécuter un de deux actions suivant une condition. Syntaxe :

```
If(condition)
    -instruction ou bloc d'instruction
else
    -instruction ou bloc d'instruction
```

NB : la partie « else » est optionnelle. Il est possible d'imbriquer des if.

Exemple :

```
Declare @stg table(numInsc int primary key, nom varchar(20), prenom
    varchar(20),moyenne numeric(4,2))
insert into @stg values(103,'LAAROUSSI','SALAH',14)
If not  exits(select * from @stg )
    Print 'la table est vide'
Else
    Print 'la table n''est pas vide'
```


Transact-SQL

Structure itérative

La structure itérative est une structure qui permet d'exécuter un même traitement plusieurs fois. Syntaxe générale :

```
While(condition)
   -instruction ou bloc d'instructions
```

où

```
Etiquette :
   -instruction ou bloc d'instructions
goto etiquette
```

exmple : calcule de la factorielle d'un nombre

```
declare @i int, @f int,@n int
set @n=6
set @f=1
set @i=1
while (@i<=@n)
   begin
```


Transact-SQL

Structure itérative

```
            set @f=@f*@i
            set @i=@i+1
    end
select @f as "la factorielle"
deuxieme solution :
declare @i int, @f int,@n int
set @n=6
set @f=1
set @i=1
label:
     set @f=@f*@i
     set @i=@i+1
if(@i<=@n) goto label
select @f as "la factorielle"
```

Transact-SQL

Structure de choix (CASE)

La fonction CASE est une expression Transact-SQL spéciale qui permet l'affichage d'une valeur de remplacement en fonction de la valeur d'une colonne. Ce changement est temporaire. Par conséquent, aucune modification permanente n'est apportée aux données.

Cet exemple affiche dans le jeu de résultats d'une requête, le nom complet de la ville dans laquelle vit chaque Formateur :

```
SELECT nom,
    CASE ville
        WHEN 'CA' THEN 'Casblanca'
        WHEN 'Kn' THEN 'Kenitra'
        WHEN 'RB' THEN 'Rabat'
    END AS 'ville d''affectation'
FROM Auditeur ORDER BY nom
```


Transact-SQL

Traitement des transactions

Une transaction est une suite d'opérations effectuées comme une seule unité logique de travail. Une unité logique de travail doit posséder quatre propriétés appelées propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité).

Atomicité : succès ou échec

Consistance : tout est fait ou rien n'est fait

Isolation : indépendant d'autres transactions ou événements

Durabilité : les changements, une fois traités, ne peuvent pas être annulés

Syntaxes

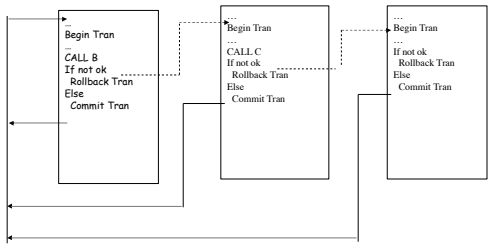
Début de transaction :	BEGIN TRAN[SACTION] [nomtransaction]
Validation de transaction :	COMMIT TRAN[SACTION] [nomtransaction]
un point de contrôle (P.C.) :	SAVE TRAN[SACTION] [nom P.C.]
Annulation de transaction :	ROLLBACK TRAN[SACTION] [nomtransaction|nom P.C.]

Exemple :

```
begin TRAN a
    insert into Passe_ex values(1,3,12.3)
    save tran ab
    insert into Passe_ex values(1,5,12.3)
    rollback tran ab
    insert into Passe_ex values(2,3,12.3)
Commit Tran a
```


Transact-SQL

Transactions imbriquées



Transact-SQL

Gestion des verrous

Lors de transactions concurrentes, SQLServergère automatiquement des verrous afin de garantir la cohérence des données de chaque transaction.

Une transaction ne pourra pas modifier des pages accessibles parune autre transaction, et ne pourra lire des pages en cours de modifications (lecture cohérente).

On peut agir sur les verrous de plusieurs façon, au niveau de la configuration et au niveau des transactions.

Mode de verrouillage	Description
Partagé	Protège une ressource en accordant uniquement l'accès en lecture. Aucune autre transaction ne peut modifier les données tant qu'il existe des verrous partagés sur la ressource.
Exclusif	Indique une modification de données, telle qu'une insertion, une mise à jour ou une suppression. Assure que plusieurs mises à jour ne puissent pas être effectuées simultanément sur la même ressource.
Mise à jour	Empêche une forme commune de blocage. Une seule transaction à la fois peut obtenir un verrou de mise à jour sur une ressource. Si la transaction modifie la ressource, le verrou de mise à jour est transformé en verrou exclusif.
Schéma	Utilisé lors de l'exécution d'une opération dépendant du schéma d'une table. Les types de verrous de schéma sont Modification du schéma (Sch-M) et Stabilité du schéma (Sch-S).
Intention	Établit une hiérarchie de verrous. Les types de verrous intentionnels les plus fréquents sont le verrou intentionnel de partage, le verrou intentionnel de mise à jour et le verrou intentionnel d'accès exclusif. Ces verrous indiquent qu'une transaction opère sur certaines (pas toutes) ressources inférieures de la hiérarchie. Les ressources de niveau inférieur auront un verrou Partagé, Mise à jour ou Exclusif.

Transact-SQL

Messages d'erreur

```
USE master
GO
sp_addmessage @msgnum = 50005,
        @severity = 10,
        @msgtext = N'rien',
        @lang='us_english';
go
sp_addmessage @msgnum = 50005,
        @severity = 10,
        @msgtext = N'RIEN',
        @lang='french';
GO
--utilisation de RAISERROR
RAISERROR (50005,10,1)
--demande l'enregistrement du message dans le journal des événements
RAISERROR(50005,16,1) With log
Go
sp_dropmessage @msgnum = 50005, @lang='all';
GO
```


Transact-SQL

Niveau de Gravité

Niveau de gravité	Description
0-9	Messages qui retournent des informations d'état ou qui signalent des erreurs sans gravité. Le Moteur de base de données ne déclenche pas d'erreurs système dont les niveaux de gravité sont compris entre 0 et 9.
10	Messages qui retournent des informations d'état ou qui signalent des erreurs sans gravité. Pour des raisons de compatibilité, le Moteur de base de données convertit le niveau de gravité 10 en 0 avant de retourner les informations d'erreur à l'application appelante.
11-16	Indique les erreurs pouvant être corrigées par l'utilisateur.
17-19	Indique des erreurs logicielles qui ne peuvent pas être corrigées par l'utilisateur. Informez votre administrateur système de l'existence du problème.
20-25	Indique l'existence de problèmes système et d'erreurs irrécupérables, ce qui signifie que la tâche du Moteur de base de données exécutant une instruction ou un lot n'est plus en cours d'exécution. La tâche enregistre des informations sur ce qui s'est produit, puis se termine. Dans la plupart des cas, la connexion de l'application à l'instance du Moteur de base de données se termine également.

Transact-SQL

les Exceptions:

TRY...CATCH :Implémente la gestion des erreurs pourTransact-SQL

```
BEGIN TRY
    SELECT 1/0;
END TRY
BEGIN CATCH
    SELECT
      ERROR_NUMBER() AS ErrorNumber,
      ERROR_SEVERITY() AS ErrorSeverity,
      ERROR_STATE() AS ErrorState,
      ERROR_PROCEDURE() AS ErrorProcedure,
      ERROR_LINE() AS ErrorLine,
      ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

ERROR_NUMBER() renvoie le num éro de l'erreur.
ERROR_SEVERITY() renvoie la gravit é de l'erreur.
ERROR_STATE() renvoie le num éro d'état de l'erreur.
ERROR_PROCEDURE() renvoie le nom de la proc édure stock ée ou du d éclencheur dans lequel s'est produit l'erreur.
ERROR_LINE() renvoie le num éro de ligne au sein de la routine qui a entra îné l'erreur.
ERROR_MESSAGE() renvoie le texte complet du message d'erreur. Ce texte comprend les valeurs fournies pour tous les param ètres substituables, tels que les longueurs, les noms d'objet ou les heures.

Exercices

- Factoriel de n
- Equation 2eme degré
- Somme = x^1/1! + x^2 / 2! +…+ x^n/n! pour x et n
- Pour une valeur A, chercher le plus petit n qui vérifie n!>=A
- PGDC de a et b
- Tableau de multiplication de N
- Nombre premier <=N

Solution sur le Forum du site :
http://www.driouchb.0fees.net/

Les Curseurs

Les curseurs permettent de réaliser des traitements itératifs sur des jeux de résultats, comme le balayage d'une table, enregistrement par enregistrement.

~~Syntaxe~~

```
DECLARE cursor_name CURSOR FOR select_statement
```

~~Arguments :~~

cursor_name: Nom du curseur de serveur Transact-SQLdéfini. L'argument cursor_name doit respecter les conventions se rapportant aux identificateurs.

select_statement: Instruction SELECT standard qui définit le jeu de résultats du curseur. Les mots-clés COMPUTE, COMPUTE BY, FOR BROWSE et INTO ne sont pas autorisés dans l'instruction SELECT d'une déclaration de curseur.

L'instruction OPEN remplit le jeu de résultats tandis que l'instruction FETCH renvoie une ligne àpartir de ce jeu de résultats.

Les autorisations DECLARE CURSOR sont octroyées par défaut àtout utilisateur qui a des autorisations SELECT sur les vues, les tables et les colonnes utilisées par le curseur.

Le variable globale @@FETCH_STATUS établit un rapport d'état de la dernière instruction FETCH

Les Curseurs

Exemple:

```
Declare @NuAud int, @nom varchar(20)
DECLARE Auditeur_cursor CURSOR FOR SELECT NuAud, nom  FROM Auditeur

OPEN Auditeur_cursor
FETCH NEXT FROM Auditeur_cursor INTO @NuAud, @nom

While @@FETCH_STATUS=0
begin
     print 'Num : ' + Cast(@NuAud as varchar(20)) + ' -Nom: '  + @nom
     FETCH NEXT FROM Auditeur_cursor INTO @NuAud, @nom
end

CLOSE Auditeur_cursor
DEALLOCATE Auditeur_cursor

GO
```

```
Num : 1 -Nom: Ali
Num : 2 -Nom: Ahmed
Num : 3 -Nom: Karim
```


Exercices

Soit le schéma relationnel suivant:

- Stagiaire(IdStg, Nom, Moyenne)
- Module(IdMod, Libelle, Coeff)
- Note(IdStg, IdMod, Note)

Écrire un programme qui met à jour la moyenne de chaque stagiaire, sans affichage détailler (affichage à partir de la table stagiaire)

Procédures Stockées

Une procédure stockée (StoredProcedurepour SQL Server) est une suite d'instructions SQL stockées dans la base de données et pouvant être exécutée par appel de son nom.

Les procédures stockées diffèrent des instructions SQL :

- Leur syntaxe est vérifiée et elles sont compilées. C'est le code compiléqui est utilisélors des appels
- Ne sont pas appelées automatiquement, mais doivent faire l'objet d'un appel explicite de la part de l'utilisateur.
- Peuvent être appelées par plusieurs applications frontales

Avantages :

- Améliorent les performances par utilisation du code compilé
- Renforcent l'intégrité de la base : en centralisant les traitements en un endroit unique unicitédu code

Procédures Stockées

Syntaxe

CREATE PROC [EDURE] procedure_name
[{ @parameterdata_type}
[VARYING] [= default] [OUTPUT]
] [,...n]
[WITH
{ RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION }]
[FOR REPLICATION]
AS sql_statement[...n]

@parameter: Un paramètre de la procédure.

Default: Valeur par défaut pour le paramètre. La valeur par défaut peut contenir des caractères génériques (%, _, [] et [^]) si la procédure utilise le nom du paramètre avec le mot cléLIKE

OUTPUT: Indique que le paramètre est un paramètre de retour renvoyé par la procédure.

Procédures Stockées

{RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}

RECOMPILE indique que SQL Server n'utilise pas le cache pour le plan de cette procédure et que la procédure est recompilée àl'exécution. Utilisez l'option RECOMPILE lorsque vous utilisez des valeurs temporairesou atypiques sans remplacer le plan d'exécution placéen mémoire cache.

ENCRYPTION

indique que SQL Server crypte l'entrée de la table syscommentscontenant le texte de l'instruction CREATE PROCEDURE. L'utilisation de l'argument ENCRYPTION évite la publication de la procédure dans le cadre de la réplication SQL Server.

FOR REPLICATION

Indique que les procédures stockées créées pour la réplication ne peuvent pas être exécutées sur l'abonné. Une procédure stockée créée avec l'option FOR REPLICATION est utilisée comme filtre de procédure stockée et n'est exécutée que pendant la réplication. Cette option ne peut pas être utilisée avec l'option WITH RECOMPILE.

La taille maximale d'une procédure stockée est limitée à128 Mo.

Procédures Stockées

Exemple : Factoriel

```
Create Proc Factoriel @n int
as
begin
  declare @f int, @i int
  select @f=1, @i=1
  while (@i<=@n)
     begin
           set @f=@f*@i
           set @i=@i+1
     end
  select @n as "N", @f as "Factoriel de N"
end

Exec Factoriel @n=5
```

Procédures Stockées

Exemple : FactorielS, Param. Sortie

```
Create Proc FactorielS @n int, @f int OUTPUT
as
begin
declare @i int
select @f=1, @i=1
while (@i<=@n)
    begin
        set @f=@f*@i
        set @i=@i+1
    end
end
Go
declare @fact int
Exec FactorielS 8, @fact OUTPUT
select @fact
Go
```


Procédures Stockées

Exemple

Sur la base auditeur en ajoute une colonne moyenne, on cr ée une procédure stocké pour le résultat de la moyenne des examens que l 'auditeur a passé.

```
-- Ajout de la colonne moyenne
alter table auditeur
add moyenne real constraint ct_my check (moyenne between 0 and 2 0)

create PROC CalcMy
as
begin
    update auditeur set moyenne=(select avg(note) from Passe_Ex Wher e
    passe_ex.nuAud=auditeur.NuAud)
End
Exec CalcMy
```


Procédures Stockées

Exemple avec paramètre

procédure qui fait le même calcule, mais pour un seul auditeur, son n uméro est passé en paramètre.

```
create PROC CalcMyP @Aud real= null
as
begin
 if @Aud is null
    update auditeur set moyenne=(select avg(note) from Passe_Ex Wher e
    passe_ex.nuAud=auditeur.NuAud);
 else
    update auditeur set moyenne=(select avg(note) from Passe_Ex Wher e
    passe_ex.nuAud=auditeur.NuAud) where auditeur.NuAud=@Aud;
end
exec CalcMyP 2
exec CalcMyP @Aud=2
exec CalcMyP
```

Fonctions

```
CREATE FUNCTION [ schema_name. ] function_name (
[ { @parameter_name [ AS ][ type_schema_name. ] parameter_data_type     [ = default ] }     [ ,...n
    ] ] )
    Scalar  Functions
RETURNS return_data_type     [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    BEGIN
            function_body
        RETURN scalar_expression
    END [ ; ]
    Inline  Table -valued  Functions
RETURNS TABLE     [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    RETURN [ ( ] select_stmt [ ) ] [ ; ]
    Multistatement    Table -valued  Functions
RETURNS @return_variable TABLE < table_type_definition >
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]    BEGIN
            function_body
        RETURN
    END [ ; ]
```

Les fonctions ne permette pas les instruction qui produisent un effet secondaire, tel que la modification d'une table (update)

Fonctions

Exemple:

```
Drop Function Fact
CREATE FUNCTION Fact (@n int)
RETURNS bigint
AS
BEGIN
  declare @f int, @i int
  select @f=1, @i=1
  while (@i<=@n)
    begin
          select @f=@f*@i, @i=@i+1
    end
  RETURN @f
END
select dbo.Fact(6);
```


Fonctions

Exemple : une fonction table enligne

```
USE GestStg

Drop Function NoteStg

CREATE FUNCTION NoteStg (@idstg int)
RETURNS table
AS
  RETURN (select libelle, note from module inner join note on
    module.idmod=note.idmod where idstg=@idstg);
Go
select * from dbo.NoteStg(2);
Go
```

Fonctions

Exp : une fonction table à instructions multiples

```
CREATE FUNCTION NoteStg2 (@idstg int)
RETURNS @rt table (Module varchar(50), Note decimal(4,2))
AS
begin
    declare @stable table(Module varchar(50), Note decimal(4,2))
    insert into @stable select libelle, note from module inner join note on
    module.idmod=note.idmod where idstg=@idstg
    insert into @stable select 'Moyenne :', Moyenne from stagiaire where idstg=@idstg
    insert @rt select * from @stable
    RETURN
end
Go

select * from dbo.NoteStg2(1);
Go
```


Exercice

Écrire une fonction qui retourne pour un stagiaire donnée, la liste des modules avec la moyenne des notes pour chaque module.

Triggers (Déclencheurs)

Forme évoluée de règles utilisées pour renforcer l'intégritéde la base de données, particulièrement l'intégritéréférentielle.

Les triggers sont un type particulier de procédure mémorisée.

-sont attachés àdes tables

- réagissent aux fonctions de création (Insert), modification (update) et suppression (delete)

-ne peuvent pas être appelés explicitement dans les applications

Les triggers sont déclenchés automatiquement par le noyau SQL à chaque intervention sur la table qui les supportent.

Un trigger est toujours associéàune table, qui peut avoir au maximum trois triggers pour (Insertion, modification et suppression de ligne)

La suppression d'une table entraîne la destruction de ses triggers

Avec SQL 2005 et plus, en peut avoir des déclencheurs sur LDD(Langage de Définition de Donnée) comme (Create, Alter, Drop) et à certaines procédures stockées système qui effectuent des opérations de type LDD

Triggers (Déclencheurs)

Principe de fonctionnement.

Deux tables virtuelles sont créées au moment de la MAJ sur la table (INSERTED, DELETED),Elles sont destinées à contenir les lignes de la table sur lesquelles ont étéeffectuées des opérations.

Les tables INSERTED et DELETED peuvent être utilisées par le trigger pour déterminer comment le traitement doit se dérouler. Ce traitement est àécrire par le développeur.

Cas de suppression d'une ligne de table (delete)

La/les lignes supprimées sont placées dans la table temporaire DELETED et supprimées de la table réelle;

Cas de création d'une ligne de table (insert)

La/les lignes nouvelles sont placées dans la table temporaire INSERTED et dans la table réelle;

Cas de modification d'une ligne de table (update)

La/les lignes avant modification sont placées dans la table temporaire DELETED et la/les lignes après modification sont placées dans la table temporaire INSERTED et dans la table réelle.

Triggers (Déclencheurs)

Syntaxe :

```
CREATE TRIGGER nom_trigger
ON nom_table
FOR INSERT
AS
    bloc d'instruction SQL
FOR UPDATE
AS
    bloc d'instruction SQL
FOR DELETE
AS
    bloc d'instruction SQL
OU
CREATE TRIGGER nom_trigger
ON nom_table
FOR INSERT, UPDATE
AS
    bloc d'instruction SQL
```

Suppression d'un trigger, Syntaxe : DROP TRIGGER nom_trigger

Triggers

Exemple :

Pour implémenté la contraint: de limiter le nombre de note par module à 3, sur la base GestStg, on doit utilisé les triggers:

```
CREATE TRIGGER LimitNote
ON note
FOR INSERT
AS
begin
   if (select count(*) from note,inserted where note.idstg=inserted.idstg and
note.idmod=inserted.idmod)>3
      begin
        print 'nombre limite 3 par module, insertion annuler'
        rollback tran
      end
end
Go
```

Exercice

Sur la base GestStg,

refaire l'implémentation des contraintes d'intégrité

- Référentiel
- De domaine

par des triggers