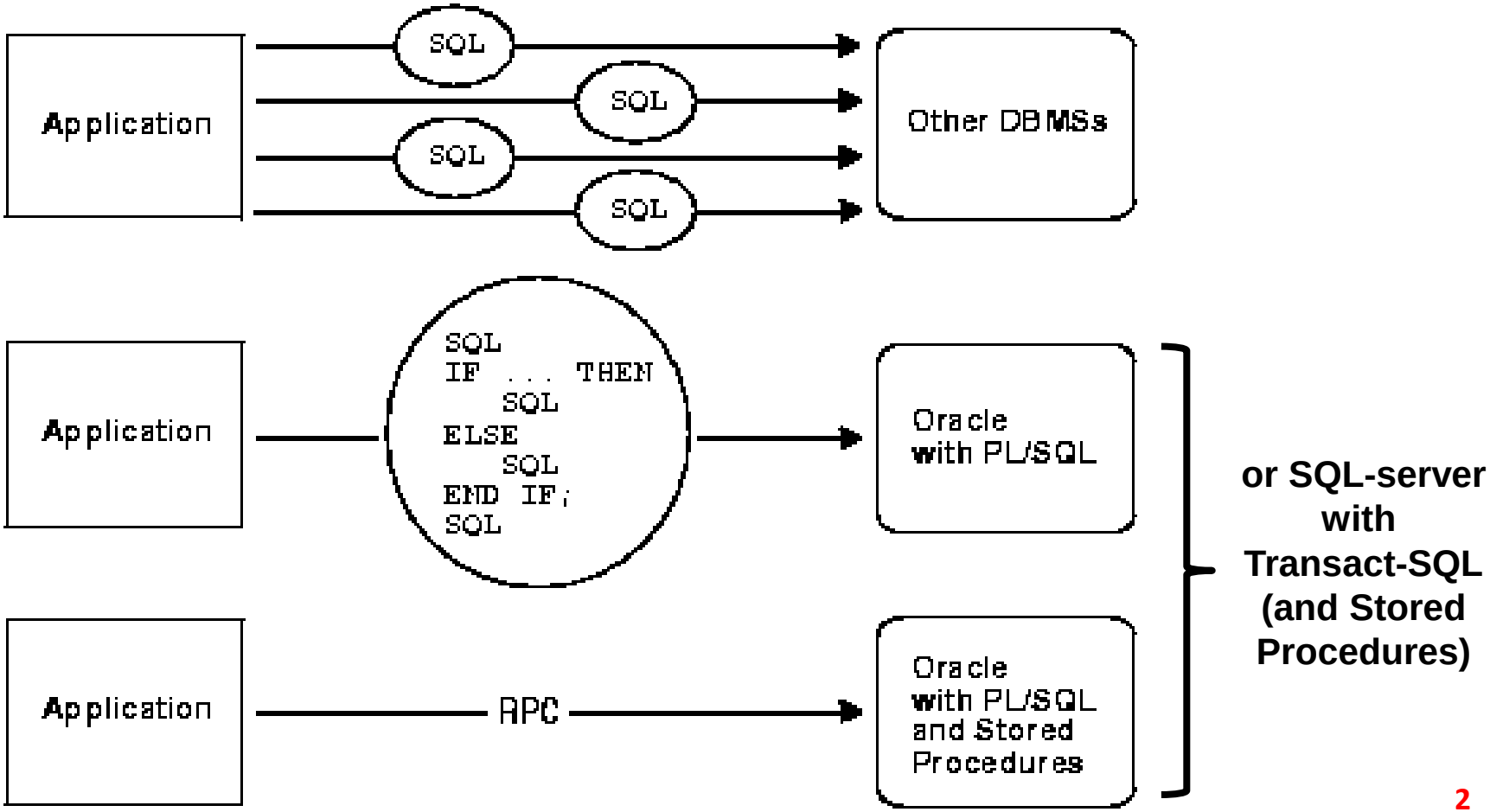
SGBD(R)O

- Procédures stockées
- Le langage Transact-SQL
- Déclencheurs

Procédure stockée

Une **procédure stockée** est une procédure applicative qui manipule la base directement sur le serveur.



Les variables

Déclaration de variables :

```
declare @a int
declare @b char(20)
declare @c float
```

Initialisation :

```
select @a = 231
select @b = 'bonjour'
select @c = 3.14
```

Affectation :

```
declare @aa int
select @aa = 17 * @a + 21
```

Les curseurs

Déclaration d'un curseur :

```
DECLARE curseur CURSOR
FOR SELECT NUM_ETU, NOM_ETU FROM ETUDIANTS
```

Utilisation :

```
OPEN curseur
FETCH curseur INTO @numetd, @nometd
WHILE @@FETCH_STATUS = 0
      BEGIN
      -- Traitement
      FETCH curseur INTO @numetd, @nometd
      END
CLOSE curseur
DEALLOCATE curseur
```

Structures de contrôle

```
IF condition
    BEGIN
      -- traitements
    END
```

```
IF condition
    BEGIN
      -- traitements
    END
ELSE
    BEGIN
      -- traitements
    END
```

```
WHILE condition
   BEGIN
      -- traitements
      -- BREAK / CONTINUE
   END
```

Gestion des erreurs

Construction TRY ... CATCH :

```
BEGIN TRY
   -- sql_statement | statement_block
END TRY
BEGIN CATCH
   [ -- sql_statement | statement_block ]
END CATCH
```

Plus d'information :

https://docs.microsoft.com/fr-fr/sql/t-sql/language-elements/

TSQL : structure d'une procédure stockée

```
CREATE [OR ALTER] PROCEDURE <nom_procedure>
       (arg1 type1, arg2 type2, ...) AS
BEGIN
      [ -- Déclaration des variables ]
       -- traitements
END;
```

```
CREATE [OR ALTER] FUNCTION <nom_fonction>
       (arg1 type1, arg2 type2, ...) RETURN type AS
BEGIN
       [ -- Déclaration des variables ]
       -- Traitements (RETURN value)
END;
```

TSQL : appel d'une procédure stockée

Pour appeler la procédure :

```
CREATE OR ALTER PROCEDURE Sauvegarde
    (@numEleve int) AS ...
```

on écrira par exemple :

```
EXEC Sauvegarde @numEleve = 4;
```

Pour appeler la fonction :

```
CREATE OR ALTER FUNCTION Moyenne (@numEleve int)
    RETURNS float AS ...
```

on écrira par exemple, pour afficher son résultat :

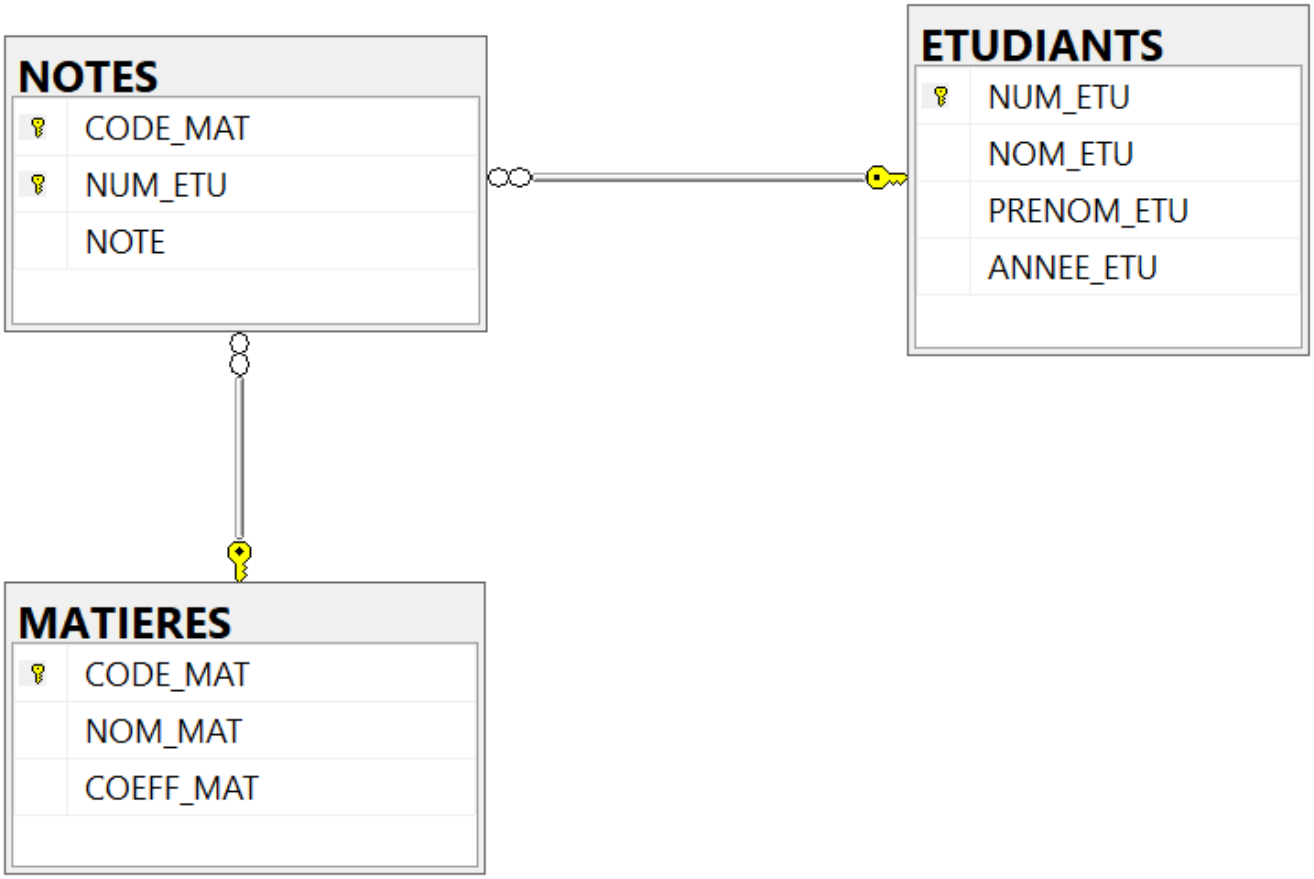
```
SELECT dbo.Moyenne(3) AS "Moyenne élève 3";
```

ou encore, pour l'utiliser :

```
SELECT nomE, dbo.Moyenne(numE) AS Moyenne FROM ELEVES;
```

Une base exemple, pour illustrer...

Etudiants / Matières (coefficient) / Notes



Un exemple de fonction (1/3)

Calcul de la moyenne (avec coefficients) d'un étudiant

```
CREATE OR ALTER FUNCTION [dbo].[MOYENNE]
  ( @numetd int ) RETURNS float AS
BEGIN
  DECLARE @somme float
  DECLARE @sommecoeff float
  DECLARE @note float
  DECLARE @coeff float
  select @somme = 0
  select @sommecoeff = 0
  ...
```

Un exemple de fonction (2/3)

```
...
  DECLARE curseurM CURSOR
  FOR SELECT NOTE, COEFF_MAT FROM NOTES INNER JOIN MATIERES
        ON NOTES.CODE_MAT = MATIERES.CODE_MAT
        WHERE NOTES.NUM_ETU = @numetd
  OPEN curseurM
  FETCH curseurM into @note,@coeff
  WHILE @@FETCH_STATUS = 0
    BEGIN
    select @somme = @somme + @note*@coeff
    select @sommecoeff = @sommecoeff + @coeff
    FETCH curseurM into @note,@coeff
    END
...
```

Un exemple de fonction (3/3)

```
...
  CLOSE curseurM
  DEALLOCATE curseurM
  IF @sommecoeff = 0
    BEGIN
    select @sommecoeff = 1
    END
  RETURN @somme/@sommecoeff
  END
```

Un exemple de procédure (1/2)

Bilan de fin d'année :

- moyenne > 11 : passe en année supérieure
- moyenne < 8 : exclus (supprimé de la table)
- sinon : rien (redoublement)

```
CREATE OR ALTER PROCEDURE [dbo].[PASSAGE] AS
BEGIN
  DECLARE @moyetd float
  DECLARE @numetd int
  DECLARE curseurP CURSOR
  FOR SELECT NUM_ETU FROM ETUDIANTS
  OPEN curseurP
  FETCH curseurP INTO @numetd
...
```

Un exemple de procédure (2/2)

```
WHILE @@FETCH_STATUS = 0
  BEGIN
    select @moyetd = dbo.MOYENNE(@numetd)
    IF @moyetd > 11 BEGIN
      UPDATE ETUDIANTS SET ANNEE_ETU = ANNEE_ETU + 1
      WHERE NUM_ETU = @numetd
    END
    ELSE IF @moyetd < 8 BEGIN
      DELETE FROM NOTES WHERE NUM_ETU = @numetd;
      DELETE FROM ETUDIANTS WHERE NUM_ETU = @numetd;
    END
  FETCH curseurP INTO @numetd
  END
CLOSE curseurP
DEALLOCATE curseurP
END
```

Déclencheurs

Un **déclencheur (trigger)** est un mécanisme, associé à une table, permettant de lancer un programme avant, après ou à la place d'événements particuliers (INSERT, UPDATE ou DELETE).

```
CREATE [OR ALTER] TRIGGER <nom_trigger> ON <table>
FOR (ou AFTER ou INSTEAD OF)
INSERT, et/ou UPDATE, et/ou DELETE) AS
BEGIN
        [ -- Déclaration des variables ]
        -- Traitements (RETURN value)
END;
```

Déclencheurs - Recommandations

- Uniquement des actions qui ne peuvent être définies à l'aide de contraintes d'intégrité
- Pas de déclencheur trop long. Si le traitement est long, placez-le dans une procédure stockée qui sera appelée par le déclencheur
- Évitez les déclencheurs récursifs !...
- Attention : un déclencheur sur une table T ne peut pas manipuler la table T… même avec un SELECT !

Un exemple de déclencheur (1/2)

Ne pas sauter un niveau... ni reculer !

```
CREATE OR ALTER TRIGGER [dbo].[UPD_ETU] ON [dbo].[ETUDIANTS]
INSTEAD OF UPDATE AS
  DECLARE @num int
  DECLARE @nom varchar(20)
  DECLARE @prenom varchar(20)
  DECLARE @newAnnee int
  DECLARE @oldAnnee int
BEGIN
  DECLARE curseurU CURSOR
  FOR select NUM_ETU, NOM_ETU, PRENOM_ETU, ANNEE_ETU
      from inserted
  OPEN curseurU
  FETCH curseurU into @num, @nom, @prenom, @newAnnee
```

Un exemple de déclencheur (2/2)

```
WHILE @@FETCH_STATUS = 0
  BEGIN
  select @oldAnnee =
       (select ANNEE_ETU from deleted where NUM_ETU = @num)
  IF @newAnnee < @oldAnnee OR @newAnnee > @oldAnnee + 1
    BEGIN
      select @newAnnee = @oldAnnee
    END
  UPDATE ETUDIANTS
    SET NOM_ETU = @nom, PRENOM_ETU = @prenom,
       ANNEE_ETU = @newAnnee  WHERE NUM_ETU = @num
  FETCH curseurU into @num, @nom, @prenom, @newAnnee
  END
CLOSE curseurU
DEALLOCATE curseurU
END
```

Un deuxième exemple de déclencheur

Sauvegarde dans ETUD_OLD des étudiants supprimés

```
CREATE OR ALTER TRIGGER [dbo].[DEL_ETU] ON [dbo].[ETUDIANTS]
AFTER DELETE AS
BEGIN
  insert into ETUD_OLD(NUM_OLD, NOM_OLD, PRENOM_OLD,
                       DATE_SORTIE)
        select NUM_ETU, NOM_ETU, PRENOM_ETU, getdate()
        from deleted
END
```

Note. *Le script de création de la base « Modules » et tous les exemples précédents sont sur Moodle.*

Et pour finir...

Exercices à réaliser sur la base Championnat

(en vous inspirant des exemples vus précédemment)

1. Une procédure qui affiche les informations d'un joueur (nom, ville de l'équipe et salaire) à partir de son identifiant (*voir exemple 3.1 du cours Moodle « Principes »*)
2. Une fonction qui renvoie la masse salariale d'une équipe à partir de son identifiant
3. Une procédure qui affiche la liste des rencontres d'une équipe (équipe qui reçoit, équipe reçue et score) à partir de son identifiant (*avec curseur*)
4. Un déclencheur qui vérifie que le salaire d'un joueur est toujours inférieur à 100 000 (après une insertion ou une mise à jour) et qui, dans le cas contraire, le met à 100 000