



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de **HONORIS UNITED UNIVERSITIES**

Cours SQL SERVER

Introduction à Transact-SQL –Elements de Base

Présenté par : Safaa AChour

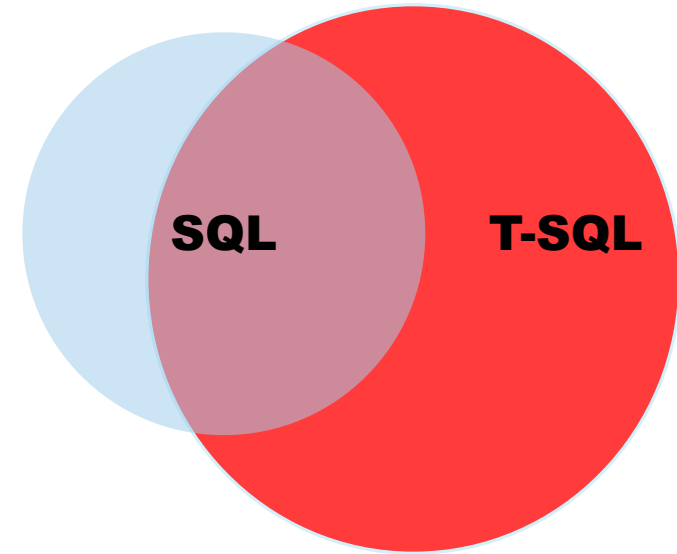
INTRODUCTION À T-SQL

Microsoft SQL Server

- utilise le T-SQL (Transact-SQL)

T - SQL

- Créé dans les années 80
- étend les fonctionnalités du SQL de base
- Transact-SQL (T-SQL) est un langage procédural (≠ de SQL Déclaratif)
- Créé pour répondre au besoin de programmer des algorithmes au sein du SGBD SQL Server
 - En particulier : procédures stockées, triggers, curseurs, transactions



T-SQL - IDENTIFIANTS

Identifiants

- Permet de faire référence à des objets
 - variable globale, variable locale, tables, etc.
- Règles dans la création de identifiants
 - Pas plus de 128 caractères
 - Pas de caractère spéciaux, ni d'espace
 - La casse n'a pas d'importance

T-SQL : LES VARIABLES

- Les noms de variables sont précédés du symbole @
- @@ précède le nom d'une variable globale (Variables du système)
- Les types de variables, sont les types SQL
- Les variables sont déclarées en utilisant le mot clé **declare**

```
Declare nom_var type [..., nom_varn type];
```

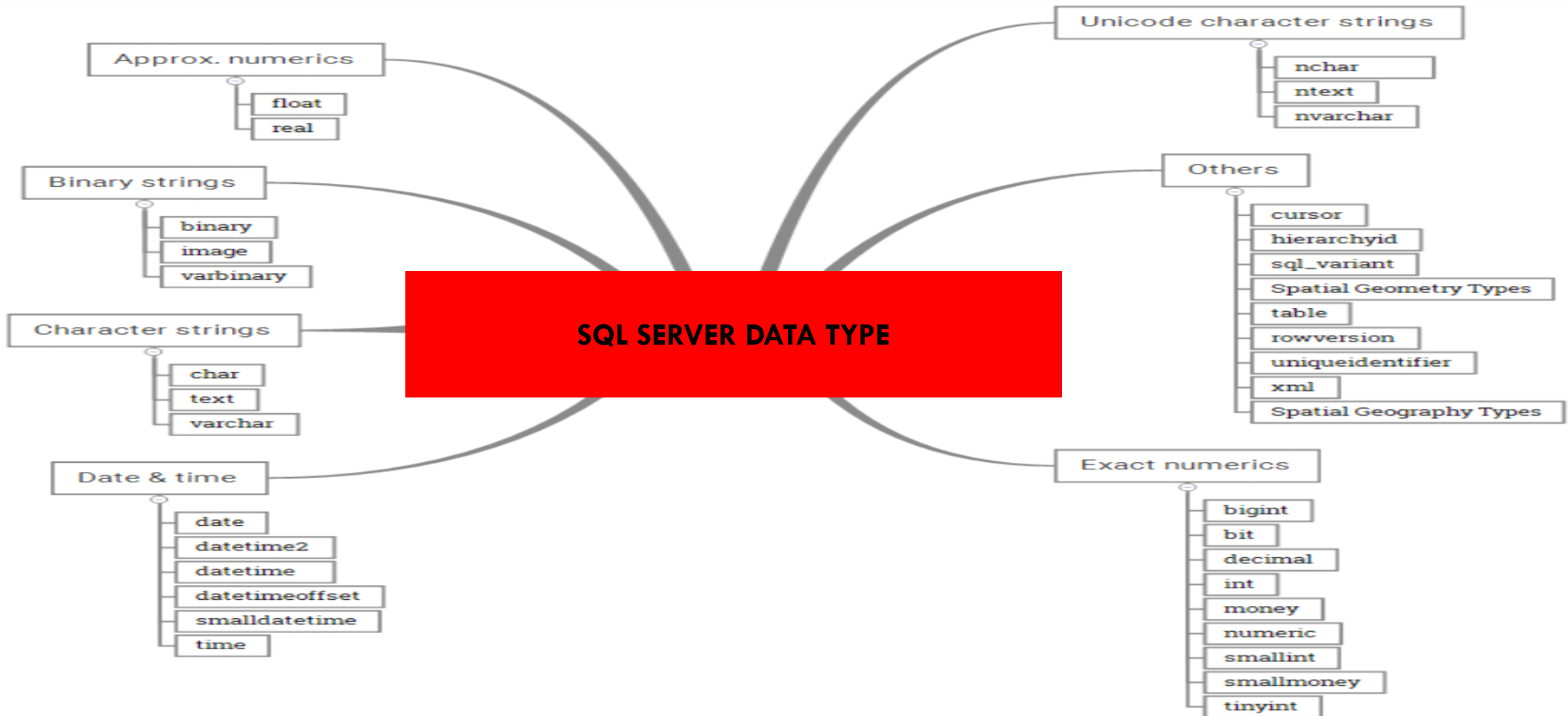
Avec le mot clé **declare**, on peut déclarer plusieurs variables dans une même ligne

Exemple :

```
Declare @var1 int, @var2 int , @var3  
varchar(20)
```

T-SQL : TYPE VARIABLES SQL SERVER

SQL SERVER DATA TYPE



T-SQL - AFFECTATION

AFFECTATION

- Affecte une valeur à une variable
- Utiliser le mot clé **SET** ou **SELECT**
- Une requête peut servir à affecter une ou plusieurs variables
 - La requête ne doit produire qu'une ligne
 - Autrement, seule la dernière ligne est prise en compte

```
Declare @id int, @nom  
varchar(20);
```

```
SET @id = 21
```

```
Set @nom = (SELECT nom  
FROM Client  
WHERE id = @id)
```

```
Declare @id int, @nom  
varchar(20);
```

```
Select @id = 21  
- - ou SET @id = 21
```

```
SELECT @nom = nom  
FROM Client  
WHERE id = @id
```

T-SQL — BLOC D'INSTRUCTIONS

Instructions

- Encadrées au sein d'un bloc d'instruction qui constitue un groupe au moment de l'exécution
- Délimité par **BEGIN** et **END**

Syntaxe :

```
BEGIN  
    /* Vos instructions seront ici */  
END
```

T-SQL - COMMENTAIRE

Commentaires

- Permet de commenter une partie du code T-SQL
- Le commentaire n'est pas interprété
 - Placer les instructions entre `/*` et `*/` pour un bloc d'instructions
 - Précédé de `--` quand il s'agit d'une ligne de commentaire

T-SQL - AFFICHAGE

Affichage du contenu d'une variable

- Utilisation de la fonction **PRINT**
- Utilisation de la clause **SELECT**

EXAMPLE

```
declare @var_name varchar(30);

select @var_name = 'hey';
print @var_name;

set @var_name = 'bonjour';
print @var_name;

select @var_name = firstname from Students;

print @var_name;

select @var_name = firstname from Students where Stdno = 'S0211';

print @var_name;

set @var_name = (select firstname from Students where Stdno = 'S0421');

print @var_name;
```



**Table de la BD
«CollegeDB »**

T-SQL — LES CONDITIONS



```
IF Boolean_expression
    { sql_statement | statement_block }
[ ELSE
    { sql_statement | statement_block } ]
```

Une seule instruction permise sinon utiliser BEGIN/ END

EXEMPLE : STRUCTURE CONDITIONNELLE

If-Else avec une seule instruction

```
DECLARE @Course_ID INT = 4

IF (@Course_ID = 4)
Select * from Course
where course_ID = 4
ELSE
Select * from Course
where course_ID != 4
```

If-Else avec un bloc d'instructions

```
DECLARE @Course_ID INT = 2

IF (@Course_ID <=2)
BEGIN
Select * from Course where course_ID = 1
Select * from Course where course_ID = 2
END
ELSE
BEGIN
Select * from Course where course_ID = 3
Select * from Course where course_ID = 4
END
```

T-SQL : L'INSTRUCTION CASE

Syntaxe :

```
CASE input_expression  
    WHEN when_expression THEN result_expression [ ...n ]  
    [ ELSE else_result_expression ]  
END
```

Ou bien.

```
CASE  
    WHEN Boolean_expression THEN result_expression [ ..n ]  
    [ ELSE else_result_expression ]  
END
```

T-SQL : L'INSTRUCTION CASE

Exemple

```
SELECT Course_ID, Course_name,  
CASE course_name  
WHEN 'SQL' THEN 'SQL is developed by IBM'  
WHEN 'PL/SQL' THEN 'PL/SQL is developed by Oracle Corporation.'  
WHEN 'MS-SQL' THEN 'MS-SQL is developed by Microsoft  
Corporation.'  
ELSE 'This is NO SQL language.'  
END AS Description  
FROM Course
```

T-SQL : CONDITIONS RÉPÉTITIVES

Syntaxe :

```
WHILE Boolean_expression  
    { sql_statement | statement_block | BREAK | CONTINUE }
```

- Le mot clé Break est utilisé dans une boucle while pour forcer l'arrêt de la boucle
- Le mot clé continue est utilisé dans une boucle while pour annuler l'itération en cours et passer aux autres itérations

EXEMPLE : CONDITION RÉPÉTITIVE

```
BEGIN
WHILE (Select avg(salaire) from employes) <= 15000)
BEGIN
  UPDATE employes SET salaire = salaire + 1000;
  IF(select MAX(Salaire) from employes ) > 50000
    BREAK;
  ELSE
    Continue;
END
END
```

Tant que la condition de la boucle while est satisfaite :

➤ on met à jour le salaire des employés

Tant que la condition de la boucle while est satisfaite et le maximum des salaire ne dépasse pas 50000 :

➤ on continue à exécuter la boucle

Si le salaire dépasse la somme de 50000, on sort de la boucle

APPLICATION 1

1. Déclarez une variable
2. Initialisez la variable à l'âge maximale des étudiants :
3. Si l'étudiant appartient au département de 'CS' alors afficher le nom, le département, et les cours qu'il suit; sinon afficher que l'étudiant n'appartient pas au département 'CS'
4. Afficher selon la note
 - Min -> la note minimal
 - En dessous de la moyenne de toutes les notes
 - En dessus de la moyenne de toutes les notes
 - La note max
5. Afficher le semestreID, L'année du semestre, les cours de ce semestre, les noms des étudiants ayant un cours dans ce semestre sans utiliser group by.

APPLICATION 2

- Si un étudiant suit le cours 'IS3511' alors on affiche Les étudiants qui ont suivi le même cours que cet étudiant; sinon afficher l'étudiant ne suit pas ce cours
- Selon 'grade' de l'étudiant
 - ❖ A ou A+ excellent
 - ❖ B ou B+ bien
 - ❖ C ou C+ assez bien
 - ❖ sinon tu as besoin davantage d'effort dans cette matière

LES TABLES TEMPORAIRES

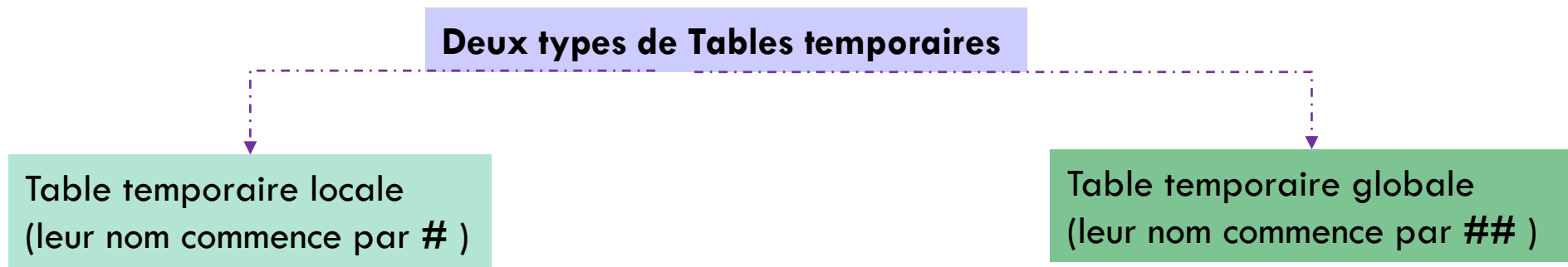
- Les tables temporaires sont un peu similaires aux tables permanentes.

La différence :

- Les tables temporaires sont créées dans la base de données système **TempDB** et sont automatiquement supprimées dès que la connexion en cours est terminée.

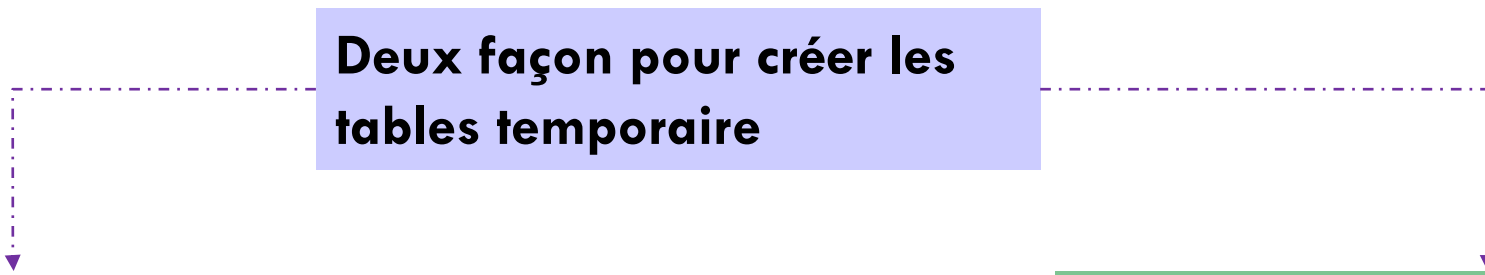
Utilité :

- Les tables temporaires permettent de stocker et traiter les résultats intermédiaires.
 - i.e. Les tables temporaires sont très utiles lorsque nous avons besoin de stocker des données temporaires.



LES TABLES TEMPORAIRES

Deux façon pour créer les tables temporaire



1ère façon : L'approche SELECT INTO

```
SELECT column_1, column_2, column_3,...  
INTO #name_of_temp_table  
FROM table_name  
WHERE condition
```


2ème façon : L'approche CREATE TABLE

```
CREATE TABLE #name_of_temp_table  
(  
    column_1 datatype,  
    column_2 datatype,  
    column_3 datatype,  
    .....  
    column_n datatype  
)
```

EXEMPLE

Exemple 1 :

```
SELECT product_id, product_name, price  
INTO #products_temp_table  
FROM products  
WHERE price > 300
```

- 
- Le nom de la table créée
 - Elle se crée automatiquement à cette étape

Exemple 2 :

```
CREATE TABLE #products2_temp_table (  
    product_id int primary key,  
    product_name nvarchar(50),  
    price int  
)
```

INSERTION DANS UNE TABLE TEMPORAIRE

Deux façons pour insérer les données dans une table temporaire

INSERT INTO #name_of_temp_table (column_1, column_2, column_3,...)

SELECT * | Column1,Column2...ColumnN
INTO #TempDestinationTable
FROM Source_Table
WHERE Condition

L'instruction SELECT permet également l'insertion des données; en effet :

- Elle crée une table « copie » de la table source avec exactement les mêmes noms de colonne et types de données
- Lit les données de la table source
- Insère des données dans la table nouvellement créée

APPLICATION 3



1. Ecrire un programme qui calcule le montant de la commande numéro 10 et affiche un message 'Commande Normale' ou 'Commande Spéciale' selon que le montant est inférieur ou supérieur à 100000 DH
2. Ecrire un programme qui supprime l'article numéro 8 de la commande numéro 5 et met à jour le stock. Si après la suppression de cet article, la commande numéro 5 n'a plus d'articles associés, la supprimer.
3. Ecrire un programme qui affiche la liste des commandes et indique pour chaque commande dans une colonne Type s'il s'agit d'une commande normale (montant ≤ 100000 DH) ou d'une commande spéciale (montant > 100000 DH)