Partie 4

DML – DATA MANIPULATION LANGUAGE

Insertion de données Mise à jour de données Suppression de données OUTPUT

Insertion de données

```
INSERT INTO table (col1, col2, ..., colN) VALUES (valeur1_col1, valeur1_col2, ..., valeur1_colN), (valeur2_col1, valeur2_col2, ..., valeur2_colN), ...
```

- L'ordre « INSERT » permet d'insérer des nouvelles lignes de données dans une table
- La liste des colonnes concernées par l'insertion n'est pas obligatoire, mais dans ce cas, les valeurs insérées doivent l'être dans le même ordre que celui dans lequel les colonnes apparaissent dans la table
- Il est possible de *ne pas insérer de valeur dans l'une des colonnes* de la tables. Il suffit pour ce faire de ne pas indiquer le nom de la colonne dans la liste des colonnes spécifiées après le nom de la table
- Sous SQL Server, il est possible d'insérer *plusieurs lignes* en une seule requête en séparant les lignes à insérer par des virgules
- L'insertion doit respecter les contraintes posées sur la table...

Insertion de données

section_id	section_name	delegate_id
1010	BSc Management	12
1020	MSc Management	9
1110	BSc Economics	15
1120	MSc Economics	6
1310	BA Sociology	23
1320	MA Sociology	6
1415	SQL Déclaratif	23
1516	NULL	12
1617	Administration SQL Server	4

Insertion de 3 nouvelles lignes de données dans la table « section »

Insertion de données

```
INSERT INTO section (section_name, section_id)
VALUES ('SQL Procédural', 1718)

INSERT INTO section VALUES (1819, 'Business Intelligence', 23)
```

section_id	section_name	delegate_id
1010	BSc Management	12
1020	MSc Management	9
1110	BSc Economics	15
1120	MSc Economics	6
1310	BA Sociology	23
1320	MA Sociology	6
1415	SQL Déclaratif	23
1516	NULL	12
1617	Administration SQL Server	4
1718	SQL Procédural	NULL
1819	Business Intelligence	23
1920	NULL	NULL

Insertion de données : DEFAULT

```
INSERT INTO section VALUES (1920, DEFAULT, DEFAULT)
INSERT INTO section VALUES (2020, DEFAULT, NULL)
```

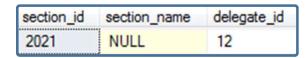
- Si l'une des colonnes possède une *valeur par défaut* ou accepte les valeurs « *NULL* », il est possible de ne pas insérer manuellement de valeur dans cette colonne en indiquant comme valeur le mot-clé « *DEFAULT* ». Il faut également procéder de cette façon pour fournir les valeurs à une colonne dont les valeurs sont *auto-incrémentées*
- L'instruction INSERT INTO table DEFAULT VALUES peut être utilisée si toutes les colonnes de la table peuvent prendre une valeur par défaut

section_id	section_name	delegate_id
1920	NULL	NULL
2020	NULL	NULL

Insertion de données : SELECT

Le résultat d'un ordre « **SELECT** » peut être utilisé comme valeur pour l'une des colonnes si cet ordre renvoie bien **une seule valeur**, du même type que la colonne correspondante

Rappel: un ordre « SELECT » utilisé comme sous-requête est toujours placé entre parenthèses



Insertion de données : SELECT

```
INSERT INTO section_archives (section_id, delegate_id)
SELECT DISTINCT s.section_id, s.delegate_id
FROM section S JOIN professor P ON P.section_id = S.section_id
```

- L'ordre « SELECT » permet également d'insérer plusieurs lignes en une seule fois dans une table
- Dans ce cas, le mot-clé « VALUES » doit être omis
- La requête doit bien entendu renvoyer le même nombre de colonnes que les colonnes à fournir

section_id	section_name	delegate_id
1020	NULL	9
1110	NULL	15
1120	NULL	6
1310	NULL	23

Mise à jour de données

```
UPDATE table
SET col1 = nouvelle_valeur_col1, col2 = nouvelle_valeur_col2, ...
WHERE ...
```

- L'ordre « UPDATE » permet de mettre à jour des données existantes dans une table
- La clause « WHERE » n'est pas obligatoire, mais elle permet de spécifier la ou les lignes auxquelles les mises à jour doivent avoir lieu
- Un ordre « SELECT » peut bien sûr être utilisé pour renvoyer la valeur à utiliser pour la mise à
 jour

Mise à jour de données

Rappel: un ordre « SELECT » utilisé comme sous-requête est toujours placé entre parenthèses

section_id	section_name	delegate_id
1010	BSc Management	12
1020	MSc Management	9
1110	SQL Déclaratif	13
1120	SQL Déclaratif	13
1310	BA Sociology	23
1320	MA Sociology	6

195

Mise à jour de données

```
UPDATE section
   SET delegate_id = std.student_id
   , section_name = 'SQL Déclaratif'
   FROM section s, student std
   WHERE CONVERT(VARCHAR,s.section_id) LIKE '11%'
   AND std.last_name LIKE 'Cruise'
```

Il est également possible d'utiliser une syntaxe semblable à celle de l'ordre « **SELECT** » pour l'exécution de l'ordre « **UPDATE** », jointures comprises. Le « **SELECT** » devient alors un « **SET** » et les colonnes ne sont pas affichées mais fournissent la valeur aux colonnes à mettre à jour

section_id	section_name	delegate_id
1010	BSc Management	12
1020	MSc Management	9
1110	SQL Déclaratif	13
1120	SQL Déclaratif	13
1310	BA Sociology	23
1320	MA Sociology	6

Suppression de données

DELETE FROM *table* **WHERE** ...

- L'ordre « **DELETE** » permet de supprimer les lignes d'une table
- La clause « WHERE » n'est pas obligatoire, mais elle permet de spécifier la ou les lignes auxquelles la suppression s'applique

DELETE FROM student

DELETE FROM student WHERE student_id = 20

OUTPUT

- Sous SQL-Server, la clause « OUTPUT » permet, à la suite d'un ordre DML, d'immédiatement renvoyer les lignes modifiées par l'ordre DML, comme si on exécutait un ordre « SELECT » sur ces données, à la suite de l'insertion
- Lors de l'utilisation d'un ordre DML, 2 tables sont utilisées pour stocker momentanément l'information manipulées. La table « INSERTED » est utilisée pour stocker momentanément toutes les nouvelles données (lors des ordres « INSERT » et « UPDATE »). La table « DELETED » stockera les données amenées à disparaître (lors d'un « UPDATE » ou d'un « DELETE »). Notons qu'il n'existe pas de table « UPDATED »

```
INSERT INTO section OUTPUT INSERTED.* VALUES (3030, NULL, 10)

UPDATE section SET section_name = 'SQL Déclaratif'
OUTPUT INSERTED.*, DELETED.section_id
WHERE section_id IN (1010,1020)

DELETE FROM section OUTPUT DELETED.section_name, DELETED.delegate_id
```

Partie 5

NOTIONS AVANCÉES

Gestion des transactions Fusion de données

Gestion des transactions

Une **Transaction** est représentée par un ordre ou un ensemble d'ordres qui **modifient l'état** de la base de données

Toute transaction au sein d'un SGBD relationnel, répond à la loi « ACID » :

ATOMIQUE

L'ensemble des ordres d'une transaction sont validés ou bien aucun ne l'est. Si un ordre échoue, l'ensemble de la transaction est annulée

COHÉRENTE

Une transaction fait toujours passer le système d'un état valide à un autre état valide dans lequel l'ensemble des règles définies pour la base de données sont respectées

ISOLÉE

Les transactions s'exécutent les unes après les autres et il n'existe qu'une seule transaction par programme client. Pour qu'une nouvelle transaction commence, la précédente doit se terminer

DURABLE

Une transaction validée l'est de façon définitive, survivant à toute défaillance technique du système

Gestion des transactions

- La gestion des transactions s'effectue principalement selon deux ordres : « COMMIT » (pour valider une transaction) et « ROLLBACK » (pour annuler une transaction)
- La plupart des système travaillent en mode « AUTO-COMMIT », ce qui signifie qu'un ordre « COMMIT » implicite est exécuté à la suite de chaque ordre visant à modifier l'état de la base de données
- Afin d'éviter le mode « AUTO-COMMIT » sous SQL-Server, il sera nécessaire de commencer l'ensemble des ordres par l'instruction « BEGIN TRANSACTION » et de terminer la transaction explicite par un « COMMIT TRANSACTION » ou un « ROLLBACK TRANSACTION »
- Un ordre « SELECT » peut faire partie d'une transaction explicite, mais il n'a aucun impact sur la transaction elle-même
- Lorsque deux transactions concurrentes essayent d'atteindre la même information au même instant, un problème de concurrence d'accès peut avoir lieu (« DEADLOCK »). Ces problèmes de concurrence d'accès peuvent être gérés par des verrous (« LOCKS ») ou en modifiant la visibilité qu'une transaction a d'une autre, c'est-à-dire le mode d'exécution des transactions. Ces notions ne seront pas abordées en détail dans le cadre de ce cours

Gestion des transactions

Cette transaction explicite ne modifie rien au niveau du système :

```
BEGIN TRANSACTION
DELETE FROM student WHERE section_id IN
( SELECT section_id FROM student
GROUP BY section id
  HAVING AVG(year_result) >= 10)
SELECT * FROM student
DELETE FROM student
SELECT * FROM student
ROLLBACK TRANSACTION
```

Fusion de données

MERGE INTO table_cible AS alias_table_cible

USING (données_à_comparer) AS alias_table_source

ON alias_table_cible.colonne_comparée = alias_table_source.colonne_comparée

WHEN MATCHED THEN ...

WHEN NOT MATCHED THEN ...

- L'ordre « MERGE » permet de comparer deux jeux de données, en se basant sur une condition de jointure entre ces jeux de données et d'agir en fonction d'un résultat semblable ou différent
- Cet ordre est notamment utilisé pour mettre à jour une table, ne modifier les données que si elles existent déjà et rajouter les lignes qui n'existent pas encore

Fusion de données

```
MERGE INTO dbo.A AS table_cible

USING (SELECT * FROM B GROUP BY col1, col2)

AS table_source (colonne1, colonne2)

ON table_cible.col1 = table_source.colonne1

WHEN MATCHED THEN UPDATE SET col2 = 'MATCH'

WHEN NOT MATCHED THEN INSERT (col1,col2)

VALUES (colonne1, colonne2);
```

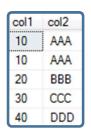


Table « A »

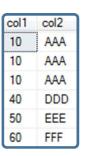


Table « B »

col1	col2
10	MATCH
10	MATCH
20	BBB
30	CCC
40	MATCH
50	EEE
60	FFF

Table « A » Après le « MERGE »

Auto-Evaluation

N'oubliez pas de prendre le temps d'évaluer le niveau de maîtrise que vous estimez avoir acquis personnellement concernant les notions abordées dans ce module !

Rappel de la signification des lettres dans les tableaux d'auto-évaluation :

- Parfait (P): vous avez parfaitement compris cette notion et vous vous sentez à votre aise
- Satisfaisant (S): vous avez compris de quoi il s'agit mais la pratique vous manque
- Vague (V): vous savez de quoi il s'agit, mais cela reste un peu vague dans votre esprit.
 Une explication supplémentaire du formateur ou une bonne révision de votre part s'impose
- Insatisfaisant (I): Vous n'avez pas du tout compris la notion abordée, il faut tout faire pour y remédier!

Auto-Evaluation

Notions à évaluer

Notions	Р	S	V	
Insertion de données ligne par ligne (VALUES)				
Insertion de données par lot (SELECT)				
Mise à jour simple de données				
Mise à jour sous forme de jointure				
Suppression de données				
Clause « OUTPUT »				
Transactions et loi ACID (en théorie)				
Gestion de transactions explicites (« BEGIN/COMMIT/ROLLBACK »)				

Références

- ELMASRI R., NAVATHE S., <u>Fundamentals of Databases Systems: Pearson New International</u> Edition, États-Unis, Pearson, 2013, 6th Edition
- BEN-GAN I., *Microsoft SQL Server 2012 T-SQL Fundamentals*, États-Unis, Microsoft Press, 2012, 1st Edition
- BEN-GAN I., KOLLAR L., SARKA D., KASS S., <u>Inside Microsoft SQL Server 2008 T-SQL Querying</u>, États-Unis, Microsoft Press, 2009, 1st Edition
- O'HEARN S., <u>OCA Oracle Database SQL SQL Certified Expert Exam Guide</u>, États-Unis, Microsoft Press, 2009, 1st Edition
- <u>MSDN-the microsoft developer network</u>, site de Microsoft : <u>msdn.microsoft.com</u>
- Oracle Database Online Documentation 12c Release (12.1), site d'Oracle: docs.oracle.com