

Secteur Tertiaire Informatique
Filière « Etude et développement »

Séquence « Développer des composants dans le langage d'une base de données »

SQL Server et Transact SQL
Le langage DML partie 4 – requêtes complexes

Apprentissage

Mise en pratique

Evaluation

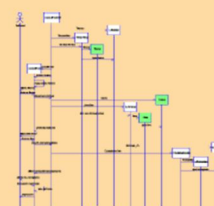
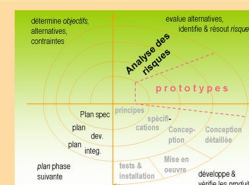
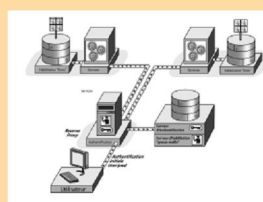


TABLE DES MATIERES

Table des matières	3
1. L'opération de sélection de données : SELECT	5
1.1 Syntaxe générale	5
1.2 Selection conditionnelle avec WHERE	5
1.3 L'opérateur BETWEEN	6
1.4 L'opérateur IN	6
1.5 L'opérateur LIKE	7
1.6 L'opérateur IS NULL	8
1.7 La clause ORDER BY	8
1.8 La clause GROUP BY	8
1.8.1 Analyse du GROUP BY	9
1.9 La clause HAVING	10
1.9.1 Analyse du HAVING	12
2. L'opérateur UNION	13
3. Les opérateurs EXCEPT ET INTERSECT	14
4. Les sous requêtes	15
4.1 Sous requête imbriquée : Renvoi d'une valeur unique	15
4.2 SOUS REQUETE IMBRIQUEE : Renvoi d'une liste de valeurs	16
4.2.1 Résultats SubSELECT > 1 ligne	17
4.3 Sous requête EN CORRELATION	18
4.4 Sous requête EN CORRELATION : Utilisation de EXISTS et NOT EXISTS	18
5. Extension de la clause de 'projection'	19
6. Sous-requête en clause from	20
7. Equivalences entre requêtes imbriquées, corrélées et jointures	20
7.1 Requêtes imbriquées et corrélées	21
7.2 Jointures et sous-requêtes	22
7.3 Requêtes imbriquées et successives	22
7.4 Critères de choix entre différentes formulations équivalentes	22

Objectifs

Ce document a pour but de vous faire découvrir les variantes de l'instruction select SQL qui permettent en une seule requête d'extraire la '*substantifique moelle*' des informations contenues dans les tables. On abordera ici principalement les notions de sous-requêtes '*imbriquées*' et '*corrélées*'.

Pour mémoire, une première partie illustrée revient sur les notions de base des requêtes select.

Pré requis

Maîtriser la logique des requêtes SQL de base.

Outils de développement

SQL Server (2012) et son IDE SQL Server Management Studio.

Méthodologie

Ce document est conçu comme un aide-mémoire pour les principales techniques à utiliser dans la conception de requêtes SQL complexes ; il complète sans la remplacer la documentation de référence.

Mode d'emploi

Symboles utilisés :



Renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur.



Propose des exercices ou des mises en situation pratiques.



Point important qui mérite d'être souligné !

Ressources

Lectures conseillées

Au fur et à mesure des besoins, la documentation de référence accessible directement par l'aide intégrée et indexée dans SQL Server Management Studio (touche F1).

1. L'OPERATION DE SELECTION DE DONNEES : SELECT

1.1 SYNTAXE GENERALE



L'instruction *SELECT* permet d'extraire des données et de les présenter triées et/ou regroupées suivant certains critères. Les enregistrements extraits devront donc vérifier certains critères exprimés dans des expressions conditionnelles.

La syntaxe de l'instruction *SELECT* présente les différentes clauses utilisables dans les sélections. Certaines clauses sont facultatives, mais il est important de respecter l'ordre d'apparition des clauses dans la requête.



SELECT <NOMS DE COLONNES OU EXPRESSIONS>	projection
FROM <NOMS DE TABLES>	jointure
WHERE <CONDITIONS DE RECHERCHE>	restriction/sélection
GROUP BY <NOMS DE COLONNE DU SELECT>	agrégation
HAVING <CONDITION DE REGROUPEMENT>	
ORDER BY <NOM OU POSITION DE COLONNE DANS L'ORDRE SELECT>	tri

Le résultat de la clause *SELECT* est une table, sous-ensemble de la (ou des) table(s) de départ :

- dont les colonnes dépendent des rubriques citées après *SELECT* (colonnes directement issues de la table d'origine, valeurs calculées, constantes, etc. ...);
- dont les lignes satisfont tant par leur contenu que pour leur présentation, aux options suivant; le cas échéant, le nom de la table.

1.2 SELECTION CONDITIONNELLE AVEC WHERE



La clause *WHERE* permet de préciser les conditions de sélection sur les lignes de la table.

Les conditions peuvent contenir une liste illimitée de prédicats renvoyant une valeur logique, combinés à l'aide des opérateurs logiques **AND** ou **OR** ; chaque prédicat permet d'exprimer une condition à l'aide d'opérateurs conditionnels.

Description	Opérateurs conditionnels
Opérateurs de comparaison	=, <>, !=, >, >=, !>, <, <=, !<
Comparaisons de plage	BETWEEN et NOT BETWEEN
Comparaisons de listes	IN et NOT IN , EXISTS et NOT EXISTS
Comparaisons de chaîne de caractères	LIKE et NOT LIKE
Valeurs inconnues	IS NULL et IS NOT NULL

1.3 L'OPERATEUR BETWEEN

 **L'opérateur BETWEEN de la clause WHERE permet d'extraire des lignes appartenant à une plage de valeurs donnée.**

Ex : lister le prénom et le nom des employés dont le salaire est compris entre 1200 et 1600 euros.

```
SELECT PRENOM, NOM FROM EMPLOYES
WHERE SALAIRE BETWEEN 1200 AND 1600
```

PRENOM	NOM	SALAIRE
PAUL	BALZAC	1200
ANNE	MARTIN	1500
PIERRE	DURAND	1530
ARTHUR	ZOLA	1600

- BETWEEN précise les bornes (comprises) entre lesquelles s'effectuera la sélection.
 - NOT BETWEEN précise les bornes en dehors desquelles s'effectuera la sélection.
- Attention : les conditions négatives ralentissent l'extraction des données.*

On aurait pu écrire :

```
SELECT PRENOM, NOM FROM EMPLOYES
WHERE SALAIRE >= 1200 AND SALAIRE <= 1600.
```

1.4 L'OPERATEUR IN

 **L'opérateur IN de la clause WHERE permet d'extraire des lignes correspondant à une liste de valeurs donnée.**

EXEMPLE 1:

```
SELECT * FROM EMPLOYES
WHERE NOEMP IN ('00010', '00020', '00050', '00100')
```

EXEMPLE 2:

```
SELECT * FROM EMPLOYES
WHERE WDEPT IN (1, 2, 3, 4, 5)
```

ou

```
SELECT * FROM EMPLOYES
WHERE WDEPT BETWEEN 1 AND 5
```

ou

```
SELECT * FROM EMPLOYES
WHERE WDEPT >= 1 AND WDEPT <= 5
```


SQL Server et Transact SQL – Requêtes complexes

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

NB : On peut aussi utiliser **NOT IN**

Attention : les conditions négatives ralentissent l'extraction des données.

1.5 L'OPERATEUR LIKE

 **L'opérateur LIKE de la clause WHERE conjointement aux caractères génériques, permet de comparer des chaînes de caractères inexactes.**

Pour une comparaison de chaîne exacte, remplacer l'opérateur LIKE par un opérateur de comparaison classique, par exemple, utiliser **NOM = 'BALZAC'** plutôt que **NOM LIKE 'BALZAC'**

LIKE ne peut être utilisé qu'avec des données de type *char*, *nchar*, *varchar*, *nvarchar* ou *datetime*

Types de caractères génériques	
Caractères génériques	Description
%	N'importe quelle chaîne comprise entre zéro et plusieurs caractères
_(trait de soulignement)	N'importe quel caractère unique
[]	N'importe quel caractère unique dans la plage (par exemple [s-w] ou [aeiouy])
[^]	N'importe quel caractère unique n'appartenant pas à la plage (par exemple [^s-w] ou [^aeiouy])

EXEMPLE

Lister les données de la table EMPLOYES dont le nom commence par la lettre "B"

```
SELECT * FROM EMPLOYES
WHERE NOM LIKE 'B %'
```

NOEMP	PRENOM	NOM	WDEPT	SALAIRE
-----	-----	-----	-----	-----
00010	PAUL	BALZAC	A00	1200
00030	ANDRE	BOULIN	C01	2100

LIKE 'BAL%'	tous les noms commençant par les lettres BAL
LIKE '%BAL%'	tous les noms contenant les lettres BAL
LIKE '___LZ__'	tous les noms de 6 caractères contenant LZ en 3ème et 4ème positions
LIKE '[S_V]ENT'	tous les noms de 4 lettres se terminant par les lettres ENT et commençant par n'importe quelle lettre comprise entre S et V.

NOT LIKE inverse de **LIKE**

1.6 L'OPERATEUR IS NULL



L'opérateur IS NULL de la clause WHERE est utilisé pour extraire des lignes pour lesquelles il manque des informations dans une colonne donnée.

Une colonne prend la valeur NULL si aucune valeur n'y est entrée au moment de la saisie des données et si aucune valeur par défaut n'a été définie pour cette colonne (c'est dans l'instruction CREATE TABLE que les colonnes sont autorisées à recevoir des valeurs NULL).

Exemple: Lister le prénom et le nom des employés dont on ne connaît pas le salaire.

```
SELECT PRENOM, NOM FROM EMPLOYES  
WHERE SALAIRE IS NULL
```

PRENOM	NOM
-----	-----
JEAN	CASENEUVE

L'opérateur IS (NOT) NULL permet de tester la ou les valeurs NULL contenues dans une colonne.

1.7 LA CLAUSE ORDER BY



La clause ORDER BY permet de préciser une séquence de tri pour le résultat d'une requête.

- **ASC séquence croissante (valeur par défaut)**
- **DESC séquence décroissante**

La clause ORDER BY doit être la dernière clause dans l'ordre SELECT et peut être spécifiée avec n'importe quelle colonne.

1.8 LA CLAUSE GROUP BY



Cette clause permet d'effectuer des regroupements de lignes en fournissant des informations statistiques au niveau des groupes.

L'intérêt d'un groupe est d'analyser les éléments qu'il contient, par exemple pour les dénombrer ou effectuer des opérations telles que somme ou moyenne.

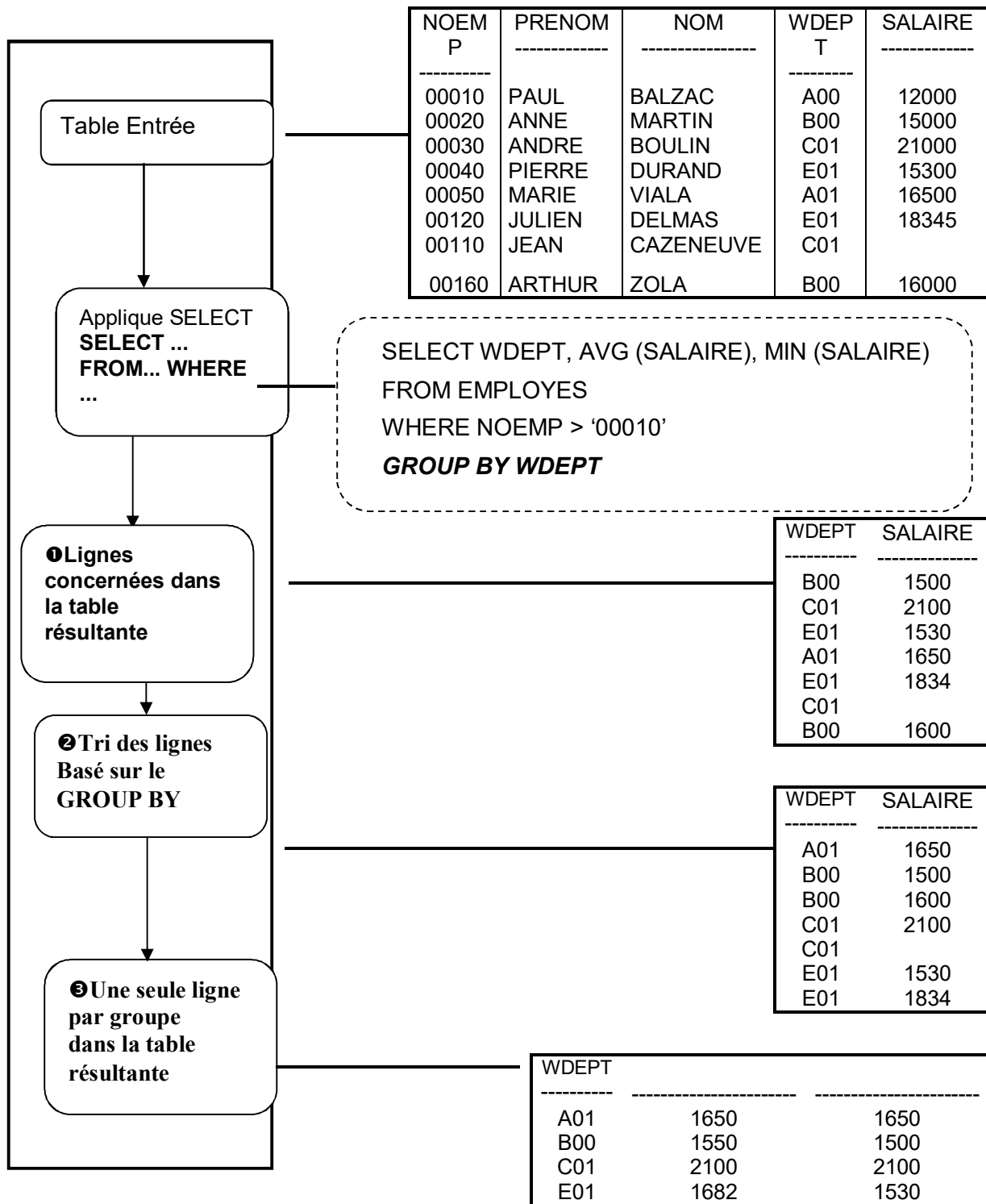
Un groupe est formé à partir d'un ensemble de lignes d'une table ayant une ou plusieurs caractéristiques communes.

Ces groupes apparaissent en séquence croissante car un classement est nécessaire en interne pour constituer les groupes.

GROUP BY est suivi du nom d'une ou plusieurs colonnes présentes dans le SELECT appelées colonnes de regroupement.


La liste de colonnes suivant SELECT ne peut comporter que les noms des colonnes de regroupement, ou des noms de fonctions.

1.8.1 Analyse du GROUP BY



- ❶ Les colonnes WDEPT et SALAIRE sont sélectionnés seulement pour les N° d'employés supérieurs à 00010.
- ❷ Ces lignes sont triées dans la séquence GROUP BY
- ❸ La moyenne et le Min des salaires sont calculés, et une ligne par groupe est générée.

1.9 LA CLAUSE HAVING

 **La clause HAVING est utilisée en conjonction avec la clause GROUP BY.**
La clause HAVING agit comme critère de sélection pour les groupes renvoyés avec la clause GROUP BY.

EXEMPLE 1 Quel est le salaire moyen et le salaire minimum des employés à l'intérieur de chaque département pour les n° employés > 00010 ?

Lister uniquement les groupes pour lesquels la moyenne est supérieure à 16 000

```
SELECT WDEPT, AVG (SALAIRE), MIN (SALAIRE) FROM EMPLOYES
WHERE NOEMP > 00010
GROUP BY WDEPT
HAVING AVG (SALAIRE) >= 16000
```

WDEPT		
-----	-----	-----
A01	1650	1650
C01	2100	2100
E01	1682	1530

La condition de recherche suivant HAVING ne peut porter que sur des colonnes de regroupement définies par la clause GROUP BY, ou sur des fonctions.

Il ne faut pas confondre les clauses WHERE et HAVING :

- WHERE permet de sélectionner des lignes avant la formation des groupes.
- HAVING permet de ne retenir que certains des groupes constitués par la clause GROUP BY.

AUTRES EXEMPLES :

Nombre d'avions dans la table AVION

```
SELECT COUNT (*)
FROM AVION
```

Le résultat est 16 (avec mon jeu d'essai)

Je peux aussi donner un nom de colonne au résultat de l'expression COUNT en ayant recours au mécanisme des alias : NomColonne AS NomAlias. Nous verrons que nous devons recourir de nouveau aux alias dans d'autres contextes.

```
SELECT COUNT(*) as "Nombre Avions"
FROM AVION
```

Nombre Avions	
1	16

Je peux aussi vouloir comptabiliser le nombre d'avions par marque.

La clause GROUP BY me permet alors de définir les conditions de regroupement des calculs récapitulatifs :

```
SELECT COUNT(*) as "Nombre Avions"
FROM AVION
GROUP BY Marque
```

	Nombre Avions
1	8
2	3
3	5

Il est préférable de faire figurer le nom de la marque dans le résultat et de trier les valeurs récapitulatives :

```
SELECT Marque, COUNT(*) as "Nombre Avions"
FROM AVION
GROUP BY Marque
ORDER BY "Nombre Avions" DESC
```

Qui peut s'écrire aussi :

```
SELECT Marque, COUNT(*) as "Nombre Avions"
FROM AVION
GROUP BY Marque
ORDER BY COUNT(*) DESC
```

	Marque	Nombre Avions
1	AIRBUS	8
2	BOEING	5
3	ATR	3

Je peux aussi ne pas vouloir conserver dans mon résultat les marques dont le nombre d'avions est inférieur à 4.

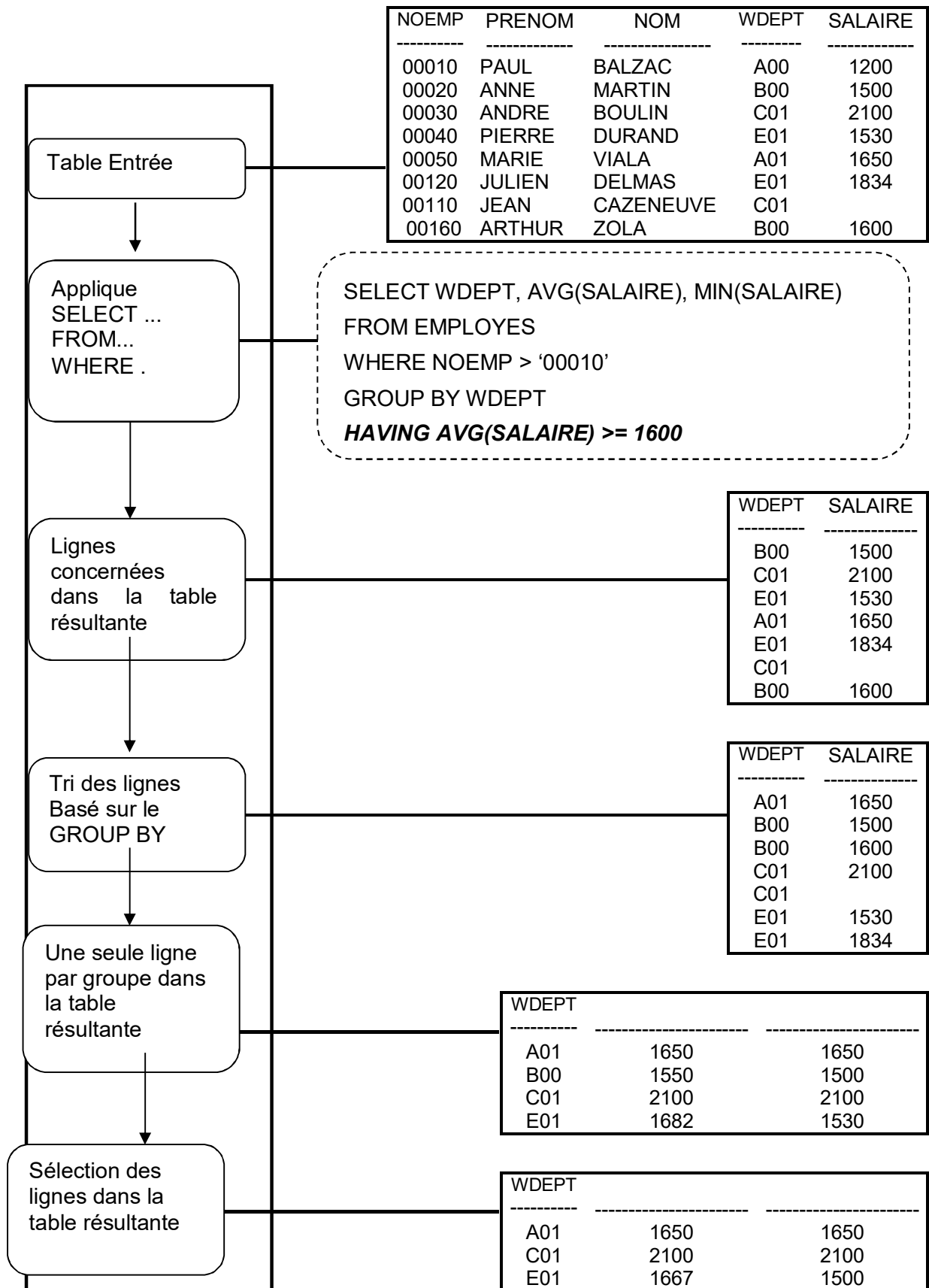
J'introduis alors dans ma requête une clause HAVING qui exprime une condition sur une opération de regroupement.

```
SELECT Marque, COUNT(*) as "Nombre Avions"
FROM AVION
GROUP BY Marque
HAVING COUNT(*) > 3 -- on ne peut pas utiliser "Nombre Avions"
ORDER BY COUNT(*) DESC
```

	Marque	Nombre Avions
1	AIRBUS	8
2	BOEING	5

Attention à ne pas confondre la clause HAVING et la clause WHERE qui filtre les lignes retenues pour les calculs.

1.9.1 Analyse du HAVING



Les étapes suivantes sont réalisées :


1. Les colonnes WDEPT et SALAIRE sont sélectionnées seulement pour les N° d'employés supérieurs à '00010'.
2. Ces lignes sont triées dans la séquence GROUP BY
3. La moyenne et le Min des salaires sont calculés.
4. Seuls les groupes ayant une moyenne des salaires d'au moins 1600 sont renvoyés.

EXEMPLE 2 : Quelle est la moyenne des salaires, le salaire minimum des employés des départements ayant plus d'un salarié ?

```
SELECT WDEPT, AVG (SALAIRE), MIN (SALAIRE) FROM EMPLOYES
WHERE NOEMP > '00010'
GROUP BY WDEPT
HAVING COUNT (*) > 1
```

WDEPT		
-	15500	1500
B00	21000	2100
C01	16822	1530
E01		

2. L'OPERATEUR UNION

 **L'opérateur UNION est utilisé pour fusionner l'information retrouvée par 2, ou plus, ordres SELECT.**

Syntaxe :

Instruction_select

UNION [ALL]

Instruction_select

Les colonnes des jeux de résultats doivent se correspondre. Il n'est pas nécessaire qu'elles portent le même nom, mais chaque jeu de résultats doit avoir le même nombre de colonnes, et ces colonnes doivent avoir des types de données compatibles et être dans le même ordre. Les structures des tables temporaires issues du Select doivent être équivalentes.

La liste des lignes issues des tables citées dans les SELECT de l'UNION ne comporte aucun doublon, sauf si l'option ALL est spécifiée.

Les noms de colonne du jeu de résultats sont ceux de la première instruction SELECT. Les alias de colonne s'ils sont à définir, seront donc définis dans la première instruction SELECT.

Si la clause ORDER BY doit être définie, elle sera définie à la suite de la dernière instruction SELECT (génération d'erreur sinon).

SQL Server et Transact SQL – Requêtes complexes

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

3. LES OPERATEURS EXCEPT ET INTERSECT

Les opérateurs ensemblistes EXCEPT et INTERSECT sont de nouveaux opérateurs permettant à l'utilisateur de retrouver des enregistrements communs à deux tables ou vues, ou des enregistrements appartenant à l'une sans appartenir à la deuxième.

Ces opérateurs respectent les mêmes règles que l'opérateur UNION : les 2 instructions Select disposent du même nombre de colonnes, et ces colonnes doivent être de type compatible (par exemple, int et smallint ou char(10) et varchar(20)).

Syntaxe :

Instruction_select

INTERSECT | EXCEPT

Instruction_select



EXCEPT renvoie toute valeur distincte de la requête de gauche mais non trouvée dans la requête de droite.

INTERSECT renvoie par contre toute valeur distincte renvoyée aussi bien par la requête à gauche que celle à droite de l'opérande INTERSECT.

4. LES SOUS REQUETES



Une sous-requête est une instruction SQL imbriquée dans une instruction SELECT, INSERT, UPDATE ou DELETE.

Elle permet de scinder une requête complexe en une suite d'étapes logiques et de résoudre un problème avec une seule instruction.

Les sous-requêtes peuvent être **imbriquées** c'est-à-dire **s'exécutant une fois** lorsque la requête externe s'exécute, ou en **corrélation** c'est-à-dire **s'exécutant une fois pour chaque ligne** renvoyée lors de l'exécution de la requête externe.

- Elles doivent être encadrées par des parenthèses à droite de l'opérateur de la clause WHERE ;
- Elles peuvent être imbriquées dans d'autres sous-requêtes ;
- Une sous-requête retourne toujours une seule colonne avec une ou plusieurs lignes ;
- La requête qui contient la sous-requête est généralement appelée *requête externe* ; la sous-requête est appelée *requête interne* ou *SubSelect*.

4.1 SOUS REQUETE IMBRIQUEE : RENVOI D'UNE VALEUR UNIQUE

EXEMPLE: Liste des employés ayant un salaire supérieur à la moyenne des salaires.

```
SELECT NOM, PRENOM FROM EMPLOYES
      WHERE SALAIRE > (SELECT AVG (SALAIRE) FROM EMPLOYES)
```

NOM	PRENOM
BOULIN	ANDRE
VIALA	MARIE
DELMAS	JULIEN

La sous requête est tout d'abord évaluée pour donner un résultat unique :

```
SELECT AVG (SALAIRE) FROM EMPLOYES
```

1630

Le salaire de chaque employé de la table EMPLOYES est alors évalué par rapport à cette valeur.

4.2 SOUS REQUETE IMBRIQUEE : RENVOI D'UNE LISTE DE VALEURS

Il est important en utilisant les sous-requêtes de savoir si le jeu de résultats comprendra plus d'une occurrence.

Si plus d'une occurrence est retournée, un des prédicats suivants doit être codé dans la clause externe WHERE.

- **IN** : contrôle si la valeur en cours du SELECT externe appartient à la liste des valeurs résultantes du SubSELECT.
- **ANY** ou **SOME** : contrôle si la valeur en cours du SELECT externe est > , < , = ,>= ou <= à au moins une des valeurs de la liste des valeurs résultantes du SubSELECT.
- **ALL** : contrôle si la valeur en cours du SELECT externe est > , < , = ,>= ou <= à toutes les valeurs de la liste des valeurs résultantes du SubSELECT.

4.2.1 Résultats SubSELECT > 1 ligne

Table EMPLOYES

NOEMP	PRENOM	NOM	WDEPT	SALAIRE
00010	PAUL	BALZAC	A00	1200
00020	ANNE	MARTIN	B00	1500
00030	ANDRE	BOULIN	C01	2100
00040	PIERRE	DURAND	E01	1530
00050	MARIE	VIALA	A01	1650
00120	JULIEN	DELMAS	E01	1834
00110	JEAN	CAZENEUVE	C01	
00160	ARTHUR	ZOLA	B00	1600

SELECT NOM, PRENOM
FROM EMPLOYES
WHERE SALAIRE

(SELECT SALAIRE FROM EMPLOYES
WHERE WDEPT = 'B00')

Résultats SubSELECT

SALAIRE
1500
1600

IN

Résultat final

NOM	PRENOM
MARTIN	ANNE
ZOLA	ARTHUR

> ANY

Résultat final

NOM	PRENOM
BOULIN	ANDRE
DURAND	PIERRE
VIALA	MARIE
DELMAS	JULIEN
ZOLA	ARTHUR

> ALL

Résultat final

NOM	PRENOM
BOULIN	ANDRE
VIALA	MARIE
DELMAS	JULIEN

4.3 SOUS REQUETE EN CORRELATION



Avec les sous-requêtes en corrélation, la requête interne se sert des informations de la requête externe et s'exécute pour chaque ligne de cette dernière.

Le traitement effectif fonctionne de la manière suivante :

- Chaque ligne de la table est utilisée en entrée pour les colonnes du SELECT primaire ;
- La ligne en cours du SELECT externe est utilisée pour fournir les valeurs pour le SubSELECT ;
Le SubSELECT est évalué et un résultat est retourné ;
Le résultat peut être une valeur simple ou plusieurs occurrences d'une seule colonne ;
- La clause WHERE du SELECT primaire peut maintenant être évaluée en utilisant les valeurs retrouvées depuis le SubSELECT.

Exemple: Liste des employés dont le salaire est inférieur à la moyenne des salaires des employés de leur département.

```
SELECT NOM, PRENOM, SALAIRE, WDEPT FROM EMPLOYES A
WHERE SALAIRE < (SELECT AVG (SALAIRE) FROM EMPLOYES B
WHERE A.WDEPT = B.WDEPT)
```

NOM	PRENOM	SALAIRE	WDEPT
-----	-----	-----	-----
MARTIN	ANNE	1500	B00
DURAND	PIERRE	1530	E01

Les alias de table sont obligatoires pour distinguer les noms des tables.

Déroulement de l'évaluation de la requête :

- Une ligne de la table EMPLOYEES est lue la valeur de WDEPT est transmise à la requête interne ;
- La requête interne est exécutée : la moyenne des salaires du département dont la valeur a été transmise est alors calculée et cette valeur est alors renvoyée à la requête externe ;
- La clause WHERE de la requête externe est alors évaluée et la ligne est incluse ou non dans le jeu de résultats ;
- Le traitement passe à la ligne suivante.

4.4 SOUS REQUETE EN CORRELATION : UTILISATION DE EXISTS ET NOT EXISTS



Le mot clé *EXIST* renvoie un indicateur vrai ou faux selon que la requête interne renvoie ou non des lignes.

La ligne de la table primaire sera incluse dans le jeu de résultats si :

- La sous-requête a renvoyé au moins une ligne si EXISTS est codé ;
- La sous-requête n'a pas renvoyé de ligne si NOT EXISTS est codé.

Exemple: Liste des employés qui ne sont pas affectés à un département.

```
SELECT NOM, PRENOM, SALAIRE, WDEPT FROM EMPLOYES
      WHERE NOT EXISTS (SELECT * FROM DEPART
                        WHERE NODEPT = WDEPT)
```

NOM	PRENOM	WDEPT
BOULIN	ANDRE	C01
DURAND	PIERRE	E01
VIALA	MARIE	A01
DELMAS	JULIEN	E01
CAZENEUVE	JEAN	C01

Il faut obligatoirement coder 'SELECT *' dans le SubSELECT, puisque EXISTS ne retourne pas de données.

5. EXTENSION DE LA CLAUSE DE 'PROJECTION'

La clause de 'projection' (terme issu de l'algèbre relationnelle à la base du langage SQL) est constituée de l'énumération des colonnes ou calculs à fournir ; elle suit juste le mot-clé SELECT.

Une extension de cette clause permet d'exprimer des calculs effectués par sous-requêtes dans la projection.

Exemple : calcul à la volée de la somme des valeurs des articles commandés :

```
select O.OrderID,o.OrderDate, (select sum(Quantity * P.UnitPrice)
from [Order Details] OD inner join Products P
on OD.ProductID = P.ProductID
where OD.OrderID = O.OrderID) as MontantCdeActualise
from Orders O
```

expression retournant une valeur
pour l'ensemble des lignes
concernées par chaque commande

requêtes corrélées

6. SOUS-REQUETE EN CLAUSE FROM

Il est possible de construire une table temporaire et de l'utiliser à la volée comme origine des données dans une requête englobante.

Exemple : calcul du maximum de la somme des valeurs des articles commandés :

```
select MAX(MontantCdeActualise)
from (select sum(Quantity * P.UnitPrice) as MontantCdeActualise
from [Order Details] OD inner join Products P
on OD.ProductID = P.ProductID
Group by OrderID) as total |
```

expression retournant
une table

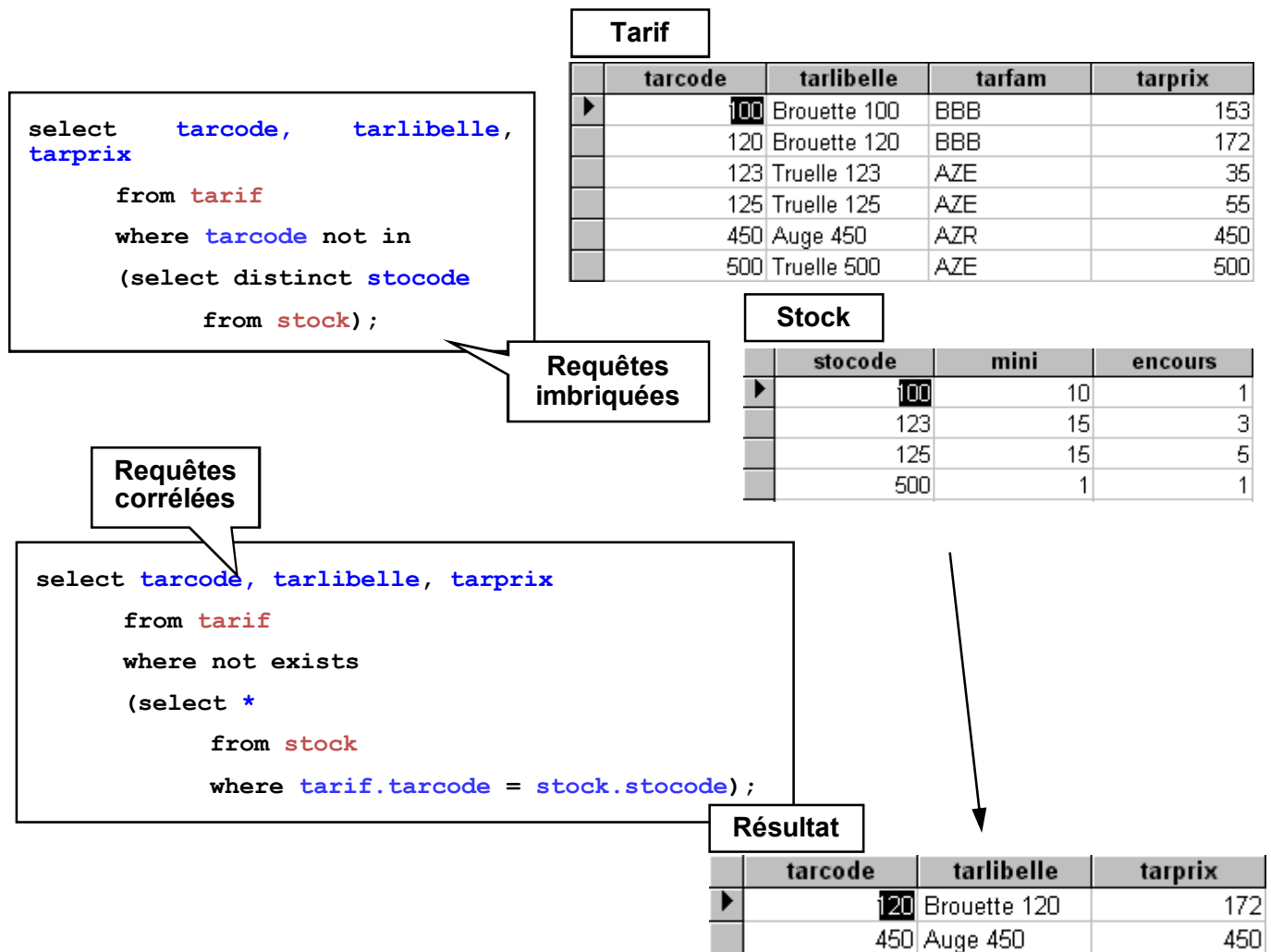
7. EQUIVALENCES ENTRE REQUETES IMBRIQUEES, CORRELEES ET JOINTURES

Il existe souvent plusieurs démarches intellectuelles pour analyser un même besoin de sélection de lignes issues de plusieurs tables. Ces démarches aboutissent à des requêtes SQL de syntaxes fort différentes qui fournissent le même résultat.

Selon la logique du développeur, plutôt mathématique/ensembliste ou plutôt algorithmique, la compréhension de requêtes existantes peut poser des problèmes dans les situations de maintenance d'application.

7.1 REQUETES IMBRIQUEES ET CORRELEES

Exemple : liste des articles du tarif qui ne sont pas en stock :

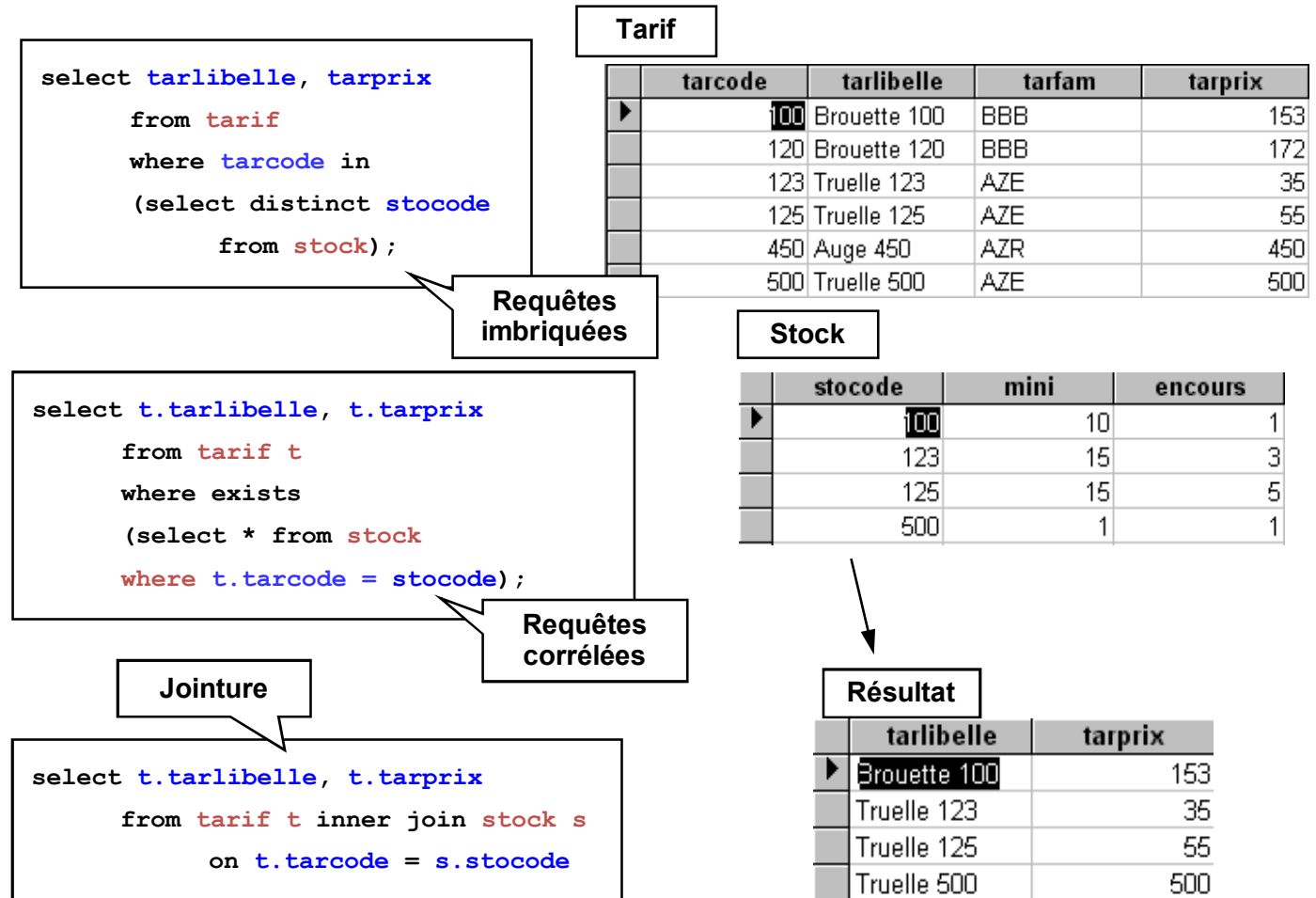


7.2 JOINTURES ET SOUS-REQUETES

La démarche à base de jointures permet de 'mettre à plat' toutes les données nécessaires et de sélectionner les lignes désirées.

La démarche ensembliste utilisant les sous-requêtes part d'une table de base et exprime les sélections nécessaires à l'aide de sous-requêtes (éventuellement en cascade).

Exemple : liste des articles du tarif qui sont en stock :



7.3 REQUETES IMBRIQUEES ET SUCCESSIVES

Des requêtes imbriquées peuvent encore se traduire par une succession de requêtes ; on commence par exécuter la requête la plus imbriquée et on mémorise son résultat dans une table temporaire qui sera réutilisée par la requête suivante, et ainsi de suite.

7.4 CRITERES DE CHOIX ENTRE DIFFERENTES FORMULATIONS EQUIVALENTES

Le seul critère de choix entre les différentes formulations sera le critère de performance lors de l'exécution de la requête ; et cela dépend en grande partie du fonctionnement interne du SGBD.

Alors, à vos chronos !

CREDITS

ŒUVRE COLLECTIVE DE L'AFPA

Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)

V. Bost - formateur

B. Hézard - Formateur

Ch. Perrachon – Ingénieure de formation

Date de mise à jour : 08/02/16

Reproduction interdite

Article L 122-4 du code de la propriété intellectuelle.

« Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droits ou ayants cause est illicite. Il en est de même pour la traduction, l'adaptation ou la reproduction par un art ou un procédé quelconque. »