

Secteur Tertiaire Informatique Filière étude - développement

Activité « Développer la persistance des données »

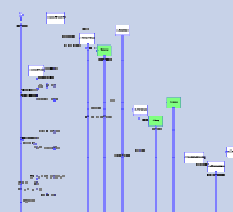
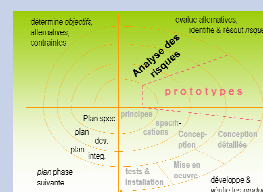
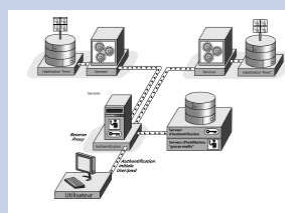
LMD 2 : SELECT

Accueil

Apprentissage

Période en
entreprise

Evaluation



JPF

SOMMAIRE

L'ORDRE SELECT	3
1. Syntaxe.....	3
2. Les expressions suivant le SELECT	4
3. L'option DISTINCT.....	6
4. La clause TOP	7
5. La clause WHERE	8
a) Les opérateurs de comparaison	9
b) Le mot clé BETWEEN	11
c) Le mot clé IN	12
d) Le mot clé LIKE	13
e) Le mot clé IS NULL	14
6. Les fonctions du SELECT	15
a) Les fonctions d'agrégation.....	15
b) Les fonctions scalaires mathématiques.....	17
c) Les fonctions scalaires de chaîne.....	18
d) Les fonctions scalaires de date et heure	20
e) Les fonctions de conversion	22
f) La fonction case	23
g) Les fonctions de classement	24
7. La clause ORDER BY.....	25
8. La clause GROUP BY	27
9. La clause HAVING.....	29
10. Les JOINTURES.....	32
11. Les SOUS-REQUETES.....	37
12. L'opérateur UNION.....	42
13. Les opérateurs EXCEPT et INTERSECT	44

L'ORDRE SELECT

1. SYNTAXE

Cette commande permet la RECHERCHE d'informations en sélectionnant les données selon divers critères.

```
SELECT <NOMS DE COLONNES OU EXPRESSIONS>
FROM   <NOMS DE TABLES>
WHERE  <CONDITIONS DE RECHERCHE>
      GROUP BY <NOMS DE COLONNE DU SELECT>
      HAVING   <CONDITION DE RECHERCHE>
      ORDER BY <NOM OU POSITION DE COLONNE
                DANS L'ORDRE SELECT>
```

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.

Exemple : Lister le contenu de la table EMPLOYES

```
SELECT * FROM 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

L'astérisque (*) insérée dans la liste de sélection permet d'extraire toutes les colonnes d'une table.

2. LES EXPRESSIONS SUIVANT LE SELECT

Dans la liste de sélection, on peut sélectionner certaines colonnes (projection).

Exemple : Lister le nom et le salaire des employés de la table EMPLOYES.

SELECT NOM, SALAIRE FROM EMPLOYES

NOM	SALAIRE
BALZAC	1200
MARTIN	1500
BOULIN	2100
DURAND	1530
VIALA	1650
DELMAS	1834
CAZENEUVE	
ZOLA	1600

On peut utiliser d'autres expressions comme :

- une constante (numérique ou chaîne de caractères),
- l'opérateur de concaténation { + } avec des chaînes de caractères uniquement,
- un des opérateurs arithmétiques { +, -, *, /, % }. (%n reste de la division par n)

Exemple : Lister le nom et le salaire en centimes de chaque employé.

SELECT NOM + ' ' + PRENOM, 'Salaire : ', SALAIRE * 100, ' centimes'
FROM EMPLOYES

BALZAC PAUL	Salaire :	120000	centimes
MARTIN ANNE	Salaire :	150000	centimes
BOULIN ANDRE	Salaire :	210000	centimes
DURAND PIERRE	Salaire :	153000	centimes
VIALA MARIE	Salaire :	165000	centimes
DELMAS JULIEN	Salaire :	183400	centimes
CAZENEUVE JEAN	Salaire :		centimes
ZOLA ARTHUR	Salaire :	160000	centimes

Dans la liste de sélection, les noms de colonne par défaut peuvent être remplacés par des **alias** en utilisant facultativement le mot clé **AS**.

Si l'alias est composé de plusieurs mots, il faut utiliser des quotes.

```
SELECT NOM + ' ' + PRENOM AS 'Nom salarié', SALAIRE * 100 Salaire, ' centimes'
FROM EMPLOYES
```

Nom salarié	Salaire	
-----	-----	-----
BALZAC PAUL	120000	centimes
MARTIN ANNE	150000	centimes
BOULIN ANDRE	210000	centimes
DURAND PIERRE	153000	centimes
VIALA MARIE	165000	centimes
DELMAS JULIEN	183400	centimes
CAZENEUVE JEAN		centimes
ZOLA ARTHUR	160000	centimes

Pour insérer une apostrophe dans un libellé ou un alias, il faut doubler l'apostrophe.

```
SELECT 'aujourd''hui'
```

Donne le résultat :

```
-----
aujourd'hui
```

3. L'OPTION DISTINCT

La syntaxe de base simplifiée du SELECT est :

```
SELECT [ ALL | DISTINCT ] nom_col1  
FROM nom_table
```

L'option **ALL** est prise par défaut, toutes les lignes sélectionnées figurent dans le résultat.

L'option **DISTINCT** permet de ne conserver **qu'un exemplaire de chaque ligne en double**.

SELECT **WDEPT** FROM EMPLOYES

SELECT **DISTINCT WDEPT** FROM EMPLOYES

WDEPT -----	WDEPT -----
A00	A00
B00	B00
C01	C01
E01	E01
A01	A01
E01	
C01	
B00	

4. LA CLAUSE TOP

La clause **TOP** affiche les n premières lignes.

Exemple 1 : Lister les 5 premiers employés de la table **EMPLOYES**.

SELECT TOP 5 NOM, PRENOM FROM **EMPLOYES**

NOM	PRENOM
-----	-----
BALZAC	PAUL
MARTIN	ANNE
BOULIN	ANDRE
DURAND	PIERRE
VIALA	MARIE

Exemple 2 : Lister 20% de la table **EMPLOYES**.

SELECT TOP 20 PERCENT NOM, PRENOM FROM **EMPLOYES**

NOM	PRENOM
-----	-----
BALZAC	PAUL
MARTIN	ANNE

Avec la version 2005, n peut être variable :

En utilisant les compléments de Transact SQL, l'exemple 1 peut être codé :

Declare @n bigint

Set @n=5

SELECT TOP (@n) NOM, PRENOM FROM **EMPLOYES**

L'exemple 2 peut aussi être codé :

Declare @n bigint

Set @n=20

SELECT TOP (@n) PERCENT NOM, PRENOM FROM **EMPLOYES**

La clause TOP peut accepter n'importe quelle expression scalaire ou expression plus élaborée telle que le résultat d'une sous-requête ou une fonction utilisateur retournant une valeur scalaire.

5. LA CLAUSE WHERE

La clause **WHERE** spécifie un ou plusieurs critères permettant de sélectionner les lignes à extraire.

Ces critères peuvent être :

- des comparaisons = <> > >= < <=
- des intervalles **BETWEEN** et **NOT BETWEEN**
- des listes **IN** et **NOT IN**
- des filtres **LIKE** et **NOT LIKE**
- des valeurs inconnues **IS NULL** et **IS NOT NULL**

Remarque : les seuls critères que vous pouvez employer sur les colonnes de type *text* sont les filtres **LIKE** (ressemblance) et **NOT LIKE** (dissemblance).

Ces critères renvoient la valeur **TRUE** (vrai), **FALSE** (faux) ou **UNKNOWN** (inconnu), et peuvent être combinés à l'aide des opérateurs logiques **AND** ou **OR**.

Le mot clé **NOT** permet la négation d'une expression.

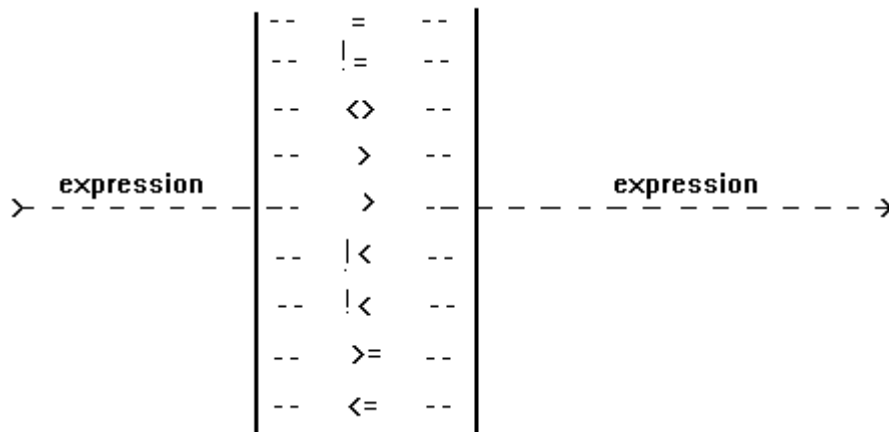
Exemple :

```
WHERE (PRIX < 100.00 OR PRIX > 135.00) AND NOT (AUTEUR = 'DUMAS')
```

Priorité des opérateurs

Quand une expression contient plusieurs opérateurs logiques, **NOT** est évalué en premier, puis **AND**, et finalement **OR**. Les opérateurs arithmétiques sont traités avant les opérateurs logiques.

a) Les opérateurs de comparaison



- Les 2 expressions doivent être de même type
- Les nombres sont comparés selon leur valeur algébrique (conversion)
- Les chaînes de caractères sont comparées de gauche à droite.
- Les données de type *char*, *nchar*, *varchar*, *nvarchar*, *text*, *datetime* et *smalldatetime* doivent être encadrées par des apostrophes.

Quelques exemples :

Rechercher dans la table EMPLOYES, les données concernant les employés qui travaillent dans les départements A00 ou E01

```
SELECT *
FROM EMPLOYES
WHERE WDEPT = 'A00' OR WDEPT = 'E01'
```

NOEMP	PRENOM	NOM	WDEPT	SALAIRE
00010	PAUL	BALZAC	A00	1200
00040	PIERRE	DURAND	E01	1530
00120	JULIEN	DELMAS	E01	1834

Rechercher dans la table EMPLOYES, les données concernant les employés dont le matricule est supérieure à 00110 et dont le salaire est égal à 1834.

```
SELECT NOM, SALAIRE, NOEMP
FROM EMPLOYES
WHERE NOEMP > '00110' AND SALAIRE = 1834
```

NOM	SALAIRE	NOEMP
-----	-----	-----
DELMAS	1834	00120

Rechercher dans la table **EMPLOYES**, les données concernant les employés dont le matricule est supérieur ou égal à 00060 et dont le salaire est égal à 1834 ou 1700

```
SELECT *
FROM EMPLOYES
WHERE (SALAIRE = 1834 OR SALAIRE = 1700) AND NOEMP >= '00060'
```

NOEMP	PRENOM	NOM	WDEPT	SALAIRE
-----	-----	-----	-----	-----
00120	JULIEN	DELMAS	E01	1834

b) Le mot clé BETWEEN

Le mot clé BETWEEN de la clause WHERE permet d'extraire des lignes appartenant à une plage de valeurs donnée.

Exemple : Lister le prénom et le nom des employés dont le salaire est compris entre 1200 et 1600 €.

```
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
```

c) Le mot clé IN

Le mot clé **IN** de la clause WHERE permet d'extraire des lignes correspondant à une liste de valeurs données.

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
```

NB : On peut aussi utiliser NOT IN

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

d) Le mot clé LIKE

Le mot clé **LIKE** de la clause WHERE, conjointement avec les caractères génériques, permet de rechercher un « motif » de caractères dans une zone de type : *char*, *nchar*, *varchar*, *nvarchar* ou *datetime*.

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

Caractères génériques	Description
%	N'importe quelle chaîne comprise entre zéro et plusieurs caractères
_(trait de soulignement)	Un seul caractère
[]	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]
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 positions 3 et 4
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

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

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

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

e) Le mot clé IS NULL

Le mot clé **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

Le prédicat IS (NOT) NULL permet de tester la ou les valeurs NULL contenues dans une colonne.

6. LES FONCTIONS DU SELECT

On peut intégrer des fonctions dans les expressions suivant le SELECT.

a) Les fonctions d'agrégation

Les fonctions d'agrégation effectuent un calcul sur un ensemble de valeurs et renvoient une valeur unique.

Lorsqu'une fonction d'agrégation est exécutée, SQL effectue la synthèse des valeurs d'une colonne spécifique pour la table complète ou pour des groupes de lignes de la table (clause GROUP BY). Une valeur d'agrégation unique est alors générée pour la table complète ou pour chaque groupe de lignes.

A l'exception de la fonction COUNT(*), les fonctions d'agrégation ignorent les valeurs NULL.

Pour les fonctions pour lesquelles ces valeurs sont précisées :

ALL applique la fonction d'agrégation à toutes les valeurs (argument par défaut),

DISTINCT spécifie que la fonction doit seulement être appliquée à chaque instance unique d'une valeur, quel que soit le nombre d'occurrences de la valeur.

AVG ([ALL | DISTINCT] expr)

Renvoie la moyenne d'une collection de valeurs précisées par l'expression entre parenthèses

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

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

SUM ([ALL | DISTINCT] expr)

Renvoie la somme des valeurs des nombres d'une colonne de type numérique

```
SELECT SUM(SALAIRE) FROM EMPLOYES  
WHERE WDEPT = 'E01'
```

MAX (expr) et MIN (expr)

Renvoie la valeur maximum (minimum) d'une collection de valeurs

```
SELECT MAX(SALAIRE) FROM EMPLOYES  
WHERE WDEPT = 'E01'
```

COUNT (*)

Renvoie le nombre de lignes d'une collection

```
SELECT COUNT(*) FROM EMPLOYES
```

renvoie le nombre d'employés.

La fonction COUNT(*) ne requiert aucun paramètre et calcule le nombre de lignes de la table spécifiée sans supprimer les doublons. Elle compte chaque ligne séparément, même les lignes contenant des valeurs NULL.

COUNT ([ALL | DISTINCT] expr)

Renvoie le nombre d'expressions non nulles ou non nulles uniques

```
SELECT COUNT (DISTINCT WDEPT) FROM EMPLOYES
```

renvoie le nombre de départements (non nuls) uniques.

COUNT_BIG ([ALL | DISTINCT] expr)

Idem count, mais la valeur de retour est au format bigint au lieu de int.

VAR (expr)

Renvoie la variance de toutes les valeurs de l'expression numérique donnée.

STDEV (expr)

Renvoie l'écart type de toutes les valeurs de l'expression numérique donnée.

b) Les fonctions scalaires mathématiques

Les fonctions scalaires effectuent une opération sur une valeur ou sur un nombre fixe de valeurs, et non sur une collection de valeurs.

Les fonctions scalaires **mathématiques** effectuent un calcul, généralement basé sur les valeurs d'entrée transmises comme arguments et elles renvoient une valeur numérique.

Mathématiques :

ABS (expr)	renvoie la valeur absolue de <i>expr</i>
CEILING (expr)	renvoie l'entier supérieur ou égal à <i>expr</i>
FLOOR (expr)	renvoie l'entier inférieur ou égal à <i>expr</i>
SIGN (expr)	renvoie 1 si <i>expr</i> positive, -1 si <i>expr</i> négative, 0 si nulle
SQRT (expr)	renvoie la racine carrée de <i>expr</i>
POWER (expr, n)	renvoie <i>expr</i> à la puissance n
SQUARE (expr)	renvoie le carré de <i>expr</i>

Trigonométriques :

PI ()	renvoie la valeur PI : 3.14159265358979
SIN (expr), COS (expr), TAN (expr), COTAN (expr)	
ASIN (expr), ACOS (expr), ATAN (expr)	
DEGREES (expr), RADIANS (expr)	
Conversion de degrés (radians) en radians (degrés)	

Logarithmiques :

LOG (expr)
EXP (expr)
LOG10 (expr)

Diverses :

RAND (expr)	renvoie un nombre aléatoire entre 0 et 1 <i>expr</i> représente la valeur de départ
ROUND (expr, n)	arrondit <i>expr</i> à la précision n
Sur la partie fractionnaire :	Select ROUND(80.49, 1) donne 80.50
Sur la partie entière :	Select ROUND(82, -1) donne 80

c) Les fonctions scalaires de chaîne

Les fonctions scalaires **de chaîne** effectuent une opération sur une valeur d'entrée de type chaîne et **renvoient une valeur** numérique ou de type chaîne.

LEN (expr)

Renvoie le nombre de caractères d'une chaîne :

LEN ('MARTIN') donne 6

LOWER (expr), UPPER (expr)

Renvoie *expr* après avoir transformé les caractères majuscules en caractères minuscules, ou minuscules en majuscules

LOWER ('MARTIN') donne martin

UPPER ('martin') donne MARTIN

LEFT (expr, n) RIGHT (expr, n)

Renvoie les n caractères les plus à gauche (droite) de *expr*

LEFT ('MARTIN', 2) donne MA

RIGHT ('MARTIN', 2) donne IN

LTRIM (expr), RTRIM (expr)

Supprime les espaces à gauche (droite) de *expr*

LTRIM (' MARTIN') donne MARTIN

RTRIM ('MARTIN ') donne MARTIN

SUBSTRING (expr, p, n)

Renvoie n caractères de la chaîne. *expr* à partir de la position spécifiée par p

SUBSTRING ('HUGO', 2, 3) donne 'UGO'

STUFF (expr1, p, n, expr2)

Supprime n caractères de *expr1* à partir d'une position donnée et insère *expr2*

STUFF ('ABCDEF', 2, 3, 'IJKLMNOP') renvoie AIJKLMNOPEF

REPLACE (expr1, expr2, expr3)

Remplace dans *expr1* toutes les occurrences de *expr2* par *expr3*

REPLACE ('ABCDAB', 'AB', 'IJK') renvoie IJKCDIJK

REPLICATE (expr, n)

Duplique une chaîne de caractères

REPLICATE ('b', 2) donne 'bb'

REVERSE (expr)

Inverse une chaîne de caractères

REVERSE ('AGIT') donne 'TIGA'

SPACE (n)

Insère un nombre de blancs : SPACE (2)

CHARINDEX (expr, expr)

Renvoie une valeur donnant la position de départ d'une chaîne dans une chaîne.

CHARINDEX ('njo', 'bonjour') donne 3

PATINDEX (%expr%, expr)

Renvoie une valeur donnant la position de départ d'une chaîne dans une chaîne.

PATINDEX ('%j%', 'bonjour') donne 4

SOUNDEX (expr1)

Associe à une chaîne de caractères un code de 4 caractères

SOUNDEX ('toto') donne T300

SOUNDEX ('tato') donne T300

SOUNDEX ('loto') donne L300

SOUNDEX ('momo') donne M500

SOUNDEX ('ctoto') donne C330

SOUNDEX ('fretfffctoto') donne F361

DIFFERENCE (expr1, expr2)

Soustrait les codes, obtenus par *soundex*, de deux chaînes de caractères en donnant un résultat entre 0 et 4 (4 si les deux chaînes sont proches l'une de l'autre, 0 si elles sont éloignées)

DIFFERENCE ('toto', 'tato') renvoie 4

DIFFERENCE ('ctoto', 'fretfffctoto') renvoie 2

CHAR (expr)

Convertit *expr* en un caractère ASCII (1 octet) :

CHAR(65) donne A

ASCII (expr)

Renvoie le code ASCII de *expr* :

ASCII (A) donne 65

NCHAR (expr)

Convertit *expr* en un caractère UNICODE (codé sur 2 octets)

UNICODE (expr)

Renvoie le code unicode de *expr*

d) Les fonctions scalaires de date et heure

Les fonctions scalaires **de date et heure** effectuent une opération sur une valeur d'entrée de type date et heure et renvoient soit une valeur numérique, de type date ou heure, ou de type chaîne.

CURRENT_TIMESTAMP, GETDATE ()

Renvoie la date et l'heure courante

DAY (expr), MONTH (expr), YEAR (expr)

Renvoie le jour, le mois ou l'année à partir de *expr*

DAY (CURRENT_TIMESTAMP) donne le jour courant

YEAR (CURRENT_TIMESTAMP) donne l'année courante

Les fonctions **DAY**, **MONTH** et **YEAR** sont respectivement synonymes de **DATEPART (dd, date)**, **DATEPART (mm, date)** et **DATEPART (yy, date)**.

Dans les fonctions suivantes, le paramètre *partie_de_date* indique la partie de date à renvoyer suivant les abréviations :

Partie de date	Nom Complet	Abréviation
Année	Year	yy, yyyy
Quart	Quart	qq, q
Mois	Month	mm, m
Quantième	DayofYear	dy
Jour	Day	d, dd
Semaine	Week	wk, ww
Jour de la semaine	WeekDay	dw
Heure	Hour	hh
Minute	Minute	mi
Secondes	Second	ss, s
Millisecondes	Millisecond	ms

DATEPART (partie_de_date, date)

Renvoie un entier représentant l'élément de date précisé dans la date spécifiée.

DATEPART (mm, CURRENT_TIMESTAMP) donne le mois courant

équivalent à :

DATEPART (month, CURRENT_TIMESTAMP)

DATEPART (mm, '21/06/2001') renvoie 6

DATEPART (dw, '21/06/2001') renvoie 4

DATENAME (partie_de_date, date)

Renvoie le nom de l'élément de date précisé dans la date spécifiée.

DATENAME (mm, '21/06/2001') renvoie 'juin'

DATENAME (dw, '21/06/2001') renvoie 'jeudi'

DATEADD (partie_de_date, nombre, date)

Renvoie une nouvelle valeur **datetime** calculée en ajoutant un intervalle à la date spécifiée.

DATEADD (dd, 15, '21/06/2001') renvoie '06/07/2001'

DATEADD (mm, 7, '21/06/2001') renvoie '21/01/2002'

DATEDIFF(partie_de_date, date_début , date_fin)

Renvoie un entier spécifiant le nombre d'intervalles compris entre date_début et date_fin.

DATEDIFF (dd, '21/06/2001', '01/07/2001') renvoie 10

DATEDIFF (mm, '21/06/2001', '01/01/2002') renvoie 7

Il est possible de configurer le premier jour de la semaine par l'intermédiaire de la fonction

SET DATEFIRST (numero_jour)

Les jours sont numérotés de 1 pour le lundi à 7 pour le dimanche ; il est possible de connaître la configuration en interrogeant la fonction @@datefirst.

e) Les fonctions de conversion

Conversion explicite d'une expression d'un type de données en une expression d'un type de données différent. CAST et CONVERT offrent la même fonctionnalité

CAST (expression **as** type_de_donnée)

Permet de convertir une valeur dans le type spécifié.

CAST (DATEPART (hh, DATE) **as** char(2))

renvoie le nombre d'heures sous forme d'une chaîne de caractères de longueur 2, extraite d'un champ DATE de type datetime.

CONVERT (type_de_donnée, expr, style)

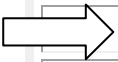
Permet de convertir l'expression dans le type_de_donnée spécifié, avec la possibilité d'utilisation d'un style (facultatif)

CONVERT (char, DATE, 8)

renvoie sous la forme d'une chaîne de caractères (hh :mn :ss) l'heure d'un champ DATE de type datetime avec application du style 8

Exemple : SELECT convert(char,getdate(),103)

donne 09/12/2014

Sans siècle (aa)	Avec siècle (aaaa)	Standard	Affichage
	0 ou 100	défaut	mois jj aaaa hh:mnAM (ou PM)
1	101	États-Unis	mm/jj/aaaa
2	102	ANSI	aa.mm.jj
 3	103	Anglais/Français	jj/mm/aa
4	104	Allemand	jj.mm.aa
5	105	Italien	jj-mm-aa
6	106 ⁽¹⁾		jj mois aa
7	107 ⁽¹⁾		mois jj, aa
8	108		hh:mn:ss
9	109		mois jj aaaa hh:mn:sssAM (ou PM)
10	110	Etat-Unis	mm-jj-aa
11	111	Japonais	aa/mm/jj
12	112	ISO	aaaammjj
13	113	Europe	jj mois aaaa hh:mn:ss:mmm (24h)
14	114		hh:mn:ss:mmmAM (ou PM)

Extrait de styles (Voir aide en ligne pour plus d'informations)

f) La fonction case

La fonction case permet de distinguer différents cas dans la liste de sélection en fonction de critères.

```
SELECT    nom,  
          CASE wdept  
            WHEN 'A00' THEN 'Agence'  
            WHEN 'B00' THEN 'Boutique'  
            WHEN 'C01' THEN 'Commercial'  
            WHEN 'E01' THEN 'Etude'  
            ELSE 'autre'  
          END 'Département'  
  
FROM EMPLOYES
```

NOM	Département
-----	-----
BOULIN	Commercial
DELMAS	Etude
VIALA	autre
ZOLA	Boutique
DURAND	Etude
MARTIN	Boutique
BALZAC	Agence
CAZENEUVE	Commercial

g) Les fonctions de classement

ROW_NUMBER()

La fonction ROW_NUMBER() permet de numéroté séquentiellement chaque ligne rendue par une instruction Select, sur la base d'un critère donné, codé sur une clause Over.

```
SELECT ROW_NUMBER() OVER(ORDER BY SALAIRE DESC) AS  
NUMCOL, PRENOM, NOM, SALAIRE FROM EMPLOYES
```

NUMCOL	PRENOM	NOM	SALAIRE
1	ANDRE	BOULIN	2100
2	JULIEN	DELMAS	1834
3	MARIE	VIALA	1650
4	ARTHUR	ZOLA	1600
5	PIERRE	DURAND	1530
6	ANNE	MARTIN	1500
7	PAUL	BALZAC	1200
8	JEAN	CAZENEUVE	

On ne peut malheureusement pas faire de la sélection sur la colonne obtenue (clause WHERE) pour limiter le nombre de lignes renvoyées, en une seule instruction. Une solution est de passer par l'intermédiaire d'une table temporaire, au moyen d'une procédure stockée.

RANK()

Avec la fonction RANK(), le résultat obtenu est le même qu'avec ROW_NUMBER() à la différence qu'une ligne peut se voir attribuer le même numéro (n) que la ligne qui la précède, la ligne suivante prenant le numéro (n+2). Dans l'exemple précédent si deux salaires sont égaux.

DENSE_RANK()

Le résultat obtenu est le même qu'avec RANK() à la différence que la ligne suivant les 2 lignes dont le numéro de colonne est égal prendra le numéro (n+1).

NTILE(n)

Cette fonction permet de numéroté les lignes par paquet de n éléments.

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.

Exemple 1: Lister le contenu de la table employé, trié par département croissant et nom décroissant

```
SELECT *  
FROM EMPLOYES  
ORDER BY WDEPT ASC, NOM DESC
```

Equivalent à:

```
SELECT *  
FROM EMPLOYES  
ORDER BY WDEPT, NOM DESC
```

NOEMP	PRENOM	NOM	WDEPT	SALAIRE
00140	HUBERT	REEVES	A00	2100
00130	PHILIPPE	LABRO	A00	1520
00010	PAUL	BALZAC	A00	1200
00060	REGINE	WIRTH	A01	1700
00160	ARTHUR	ZOLA	B00	1600
00020	ANNE	MARTIN	B00	1500
00030	ANDRE	BOULIN	C01	2100
00050	MARIE	VIALA	E01	1650
00040	PIERRE	DURAND	E01	1530
00120	JULIEN	DELMAS	E01	1834
00150	MARIE	CURIE	E01	3400

Exemple 2: Lister par salaire décroissant et par département croissant, le salaire en centimes, le prénom, le nom, et le département des employés.

```
SELECT SALAIRE*100, PRENOM, NOM, WDEPT
FROM EMPLOYES
ORDER BY 1 DESC, WDEPT
```

	PRENOM	NOM	WDEPT
-----	-----	-----	-----
	JEAN	CAZENEUVE	A00
300000	MARIE	CURIE	E01
210000	HUBERT	REEVES	A00
210000	REGINE	WIRTH	A01
183450	ANNE	MARTIN	B00
170000	ANDRE	BOULIN	C01
165000	MARIE	VIALA	E01
160000	ARTHUR	ZOLA	B00
153000	PIERRE	DURAND	E01
152000	PHILIPPE	LABRO	A00
150000	JULIEN	DELMAS	E01
120000	PAUL	BALZAC	A00

La valeur « NULL » occupe la position la plus élevée.

8. LA CLAUSE GROUP BY

Cette clause permet de définir et de traiter des groupes.

Un groupe est formé à partir d'un ensemble de lignes d'une table dont une ou plusieurs colonnes ont des valeurs communes.

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

Une seule ligne par groupe apparaît dans la table résultante.

Les colonnes de cette ligne, appelées colonne(s) de regroupement, sont celles citées dans la clause **GROUP BY**.

La liste de colonnes suivant SELECT ne peut comporter que des noms de colonnes de regroupement, ou des fonctions d'agrégation (§ pages 15-16).

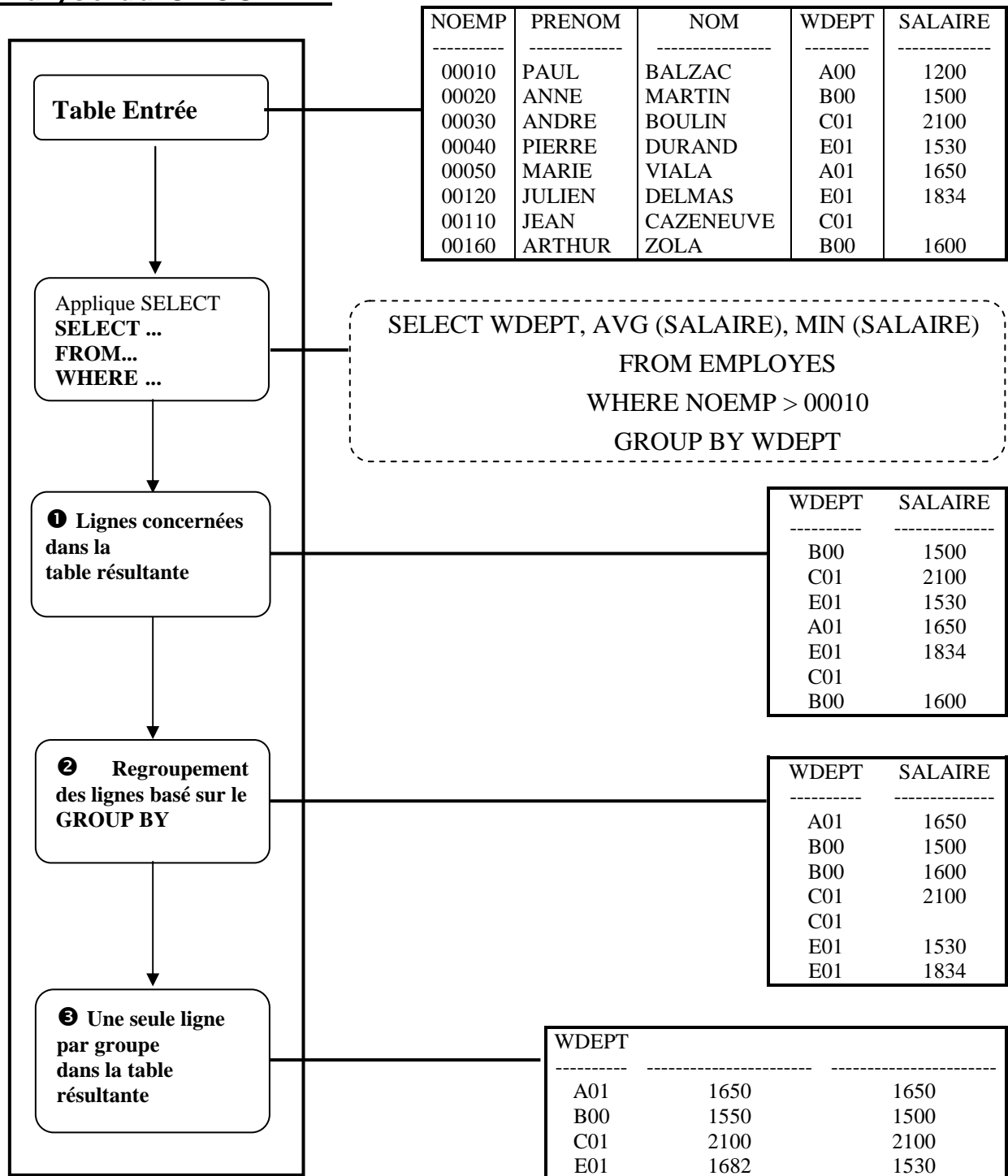
Exemple : Lister le salaire moyen et le salaire minimum des employés à l'intérieur de chaque département pour les n°employés > 00010.

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

WDEPT		
-----	-----	-----
A01	1650	1650
B00	1550	1500
C01	2100	2100
E01	1682	1530

Il y a autant de groupes que de valeurs de WDEPT distinctes.

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 regroupées selon les critères de la clause GROUP BY
- ❸ Une ligne par groupe est générée avec la moyenne des salaires et le salaire mini

9. LA CLAUSE HAVING

La clause HAVING est utilisée en conjonction avec la clause GROUP BY (d'une façon identique au WHERE pour des lignes non groupées).

La clause HAVING agit comme critère de sélection pour les groupes renvoyés avec la clause GROUP BY.

Exemple 1 :

Lister le salaire moyen et le salaire minimum des employés à l'intérieur de chaque département pour les n°employés > 00010.

Lister seulement les groupes pour lesquels la moyenne est supérieure à 1600.

```
SELECT WDEPT, AVG (SALAIRE), MIN (SALAIRE)
FROM EMPLOYES
WHERE NOEMP > 00010
GROUP BY WDEPT
HAVING AVG (SALAIRE) >= 1600
```

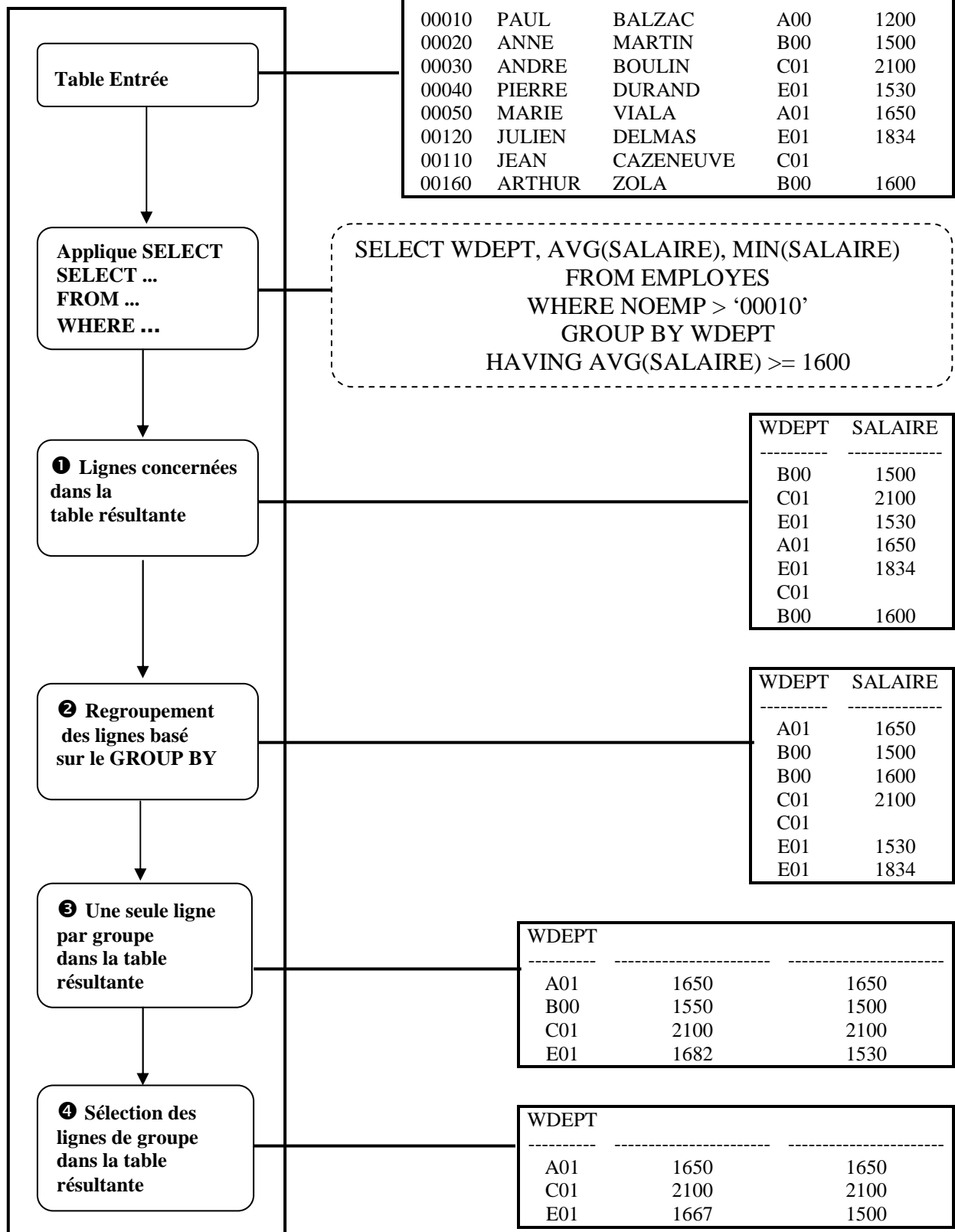
WDEPT		
A01	1650	1650
C01	2100	2100
E01	1682	1530

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**. HAVING porte sur le résultat d'un GROUP BY.

Analyse du HAVING



Les étapes suivantes sont réalisées :

- ❶ Les colonnes WDEPT et SALAIRE sont sélectionnées seulement pour les N° d'employés supérieurs à 00010
- ❷ Ces lignes sont regroupées selon les critères de la clause GROUP BY
- ❸ Le salaire moyen et le salaire minimum sont calculés et une seule ligne par groupe est générée
- ❹ Seuls les groupes ayant une moyenne des salaires supérieure à 1600 sont renvoyés.

Exemple 2 :

Lister la moyenne des salaires et le salaire minimum des employés des départements ayant plus d'un salarié

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

WDEPT		
-----	-----	-----
B00	1550	1500
C01	2100	2100
E01	1682	1530

10. LES JOINTURES

Une jointure est une opération qui permet de **combinaison plusieurs tables pour obtenir un jeu de résultats unique** intégrant des lignes et des colonnes de chaque table.

La plupart des conditions de jointure sont basées sur la clé primaire d'une table et la clé étrangère d'une autre table, mais ce n'est pas une obligation.

Quand la jointure se fait sur des tables ayant des noms de colonne identiques, les noms en double doivent être préfixés par leur nom de table ou un alias associé.

Il existe trois types de jointure : les jointures internes, les jointures externes et les jointures croisées.

Syntaxe partielle :

```
SELECT nom_colonne1, nom_colonne2, ... nom_colonne n
FROM nom_table1
      [ INNER | {LEFT | RIGHT | FULL} [OUTER] ] JOIN nom_table2
      ON conditions _recherche
```

- Le mot clé **JOIN** et ses options spécifient les tables à joindre et la manière de les joindre.
- Le mot clé **ON** spécifie les colonnes communes aux tables.

Jointure interne (par défaut)

Exemple : Liste des employés avec le nom du département où ils sont affectés.

```
SELECT NOM, PRENOM, SALAIRE, NOMDEPT
FROM EMPLOYES INNER JOIN DEPART ON WDEPT = NODEPT
```

NOM	PRENOM	SALAIRE	NOMDEPT
BALZAC	PAUL	1200	SERVICE INFORMATIQUE
MARTIN	ANNE	1500	PLANNING
ZOLA	ARTHUR	1600	PLANNING

La jointure interne renvoie uniquement les lignes en correspondance. Les lignes comportant un NULL dans la condition de recherche (de part ou d'autre) sont ignorées.

Remarque : la clause INNER JOIN peut être abrégée en JOIN.

Analyse de la JOINTURE

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

Table DEPART

NODEPT	NOMDEPT	NOMGR
A00	SERVICE INFORMATIQUE	00010
B00	PLANNING	00020
E21	LOGICIEL	00030

```
SELECT NOM, PRENOM, SALAIRE, NOMDEPT
FROM EMPLOYES
INNER JOIN DEPART
ON WDEPT = NODEPT
```

NOM	PRENOM	SALAIRE	NOMDEPT
BALZAC	PAUL	1200	SERVICE INFORMATIQUE
MARTIN	ANNE	1500	PLANNING
ZOLA	ARTHUR	1600	PLANNING

On aurait pu coder la requête suivant la syntaxe suivante :

```
SELECT NOM, PRENOM, SALAIRE, NOMDEPT
FROM EMPLOYES, DEPART
WHERE WDEPT = NODEPT
```

Jointure externe gauche ou droite

- La jointure **externe gauche** permet d'afficher **toutes les lignes de la première table** nommée (table située à gauche de la clause JOIN).
- La jointure **externe droite** permet d'afficher **toutes les lignes de la seconde table** nommée (table située à droite de la clause JOIN).
- La clause LEFT OUTER JOIN peut être abrégée en LEFT JOIN
La clause RIGHT OUTER JOIN peut être abrégée en RIGHT JOIN
- La jointure **externe complète** (FULL OUTER JOIN) renvoie toutes les lignes, y compris celles sans correspondance, des 2 tables.

Exemple : Liste de tous les employés avec le nom du département où ils sont affectés.

```
SELECT NOM, PRENOM, SALAIRE, NOMDEPT  
FROM EMPLOYES LEFT JOIN DEPART ON WDEPT = NODEPT
```

NOM	PRENOM	SALAIRE	NOMDEPT
BALZAC	PAUL	1200	SERVICE INFORMATIQUE
MARTIN	ANNE	1500	PLANNING
BOULIN	ANDRE	2100	<NULL>
DURAND	PIERRE	1530	<NULL>
VIALA	MARIE	1650	<NULL>
DELMAS	JULIEN	1834	<NULL>
CAZENEUVE	JEAN		<NULL>
ZOLA	ARTHUR	1600	PLANNING

La jointure externe renvoie toutes les lignes de la table EMPLOYES.

La colonne NOMDEPT contient la valeur NULL pour les départements n'existant pas dans la table DEPART. *Ces dernières ne seraient pas apparues avec un simple (INNER) JOIN.*

Jointures croisées

Les jointures croisées affichent toutes les combinaisons de toutes les lignes des tables jointes. Il n'est pas nécessaire de disposer de colonnes communes, et la clause ON n'est pas utilisée pour les jointures croisées.

Les jointures croisées sont très rarement utilisées sur une base de données normalisée. Lors d'une jointure croisée, SQL génère un **produit cartésien** dans lequel le nombre de lignes du jeu de résultats est égal au nombre de lignes de la première table multiplié par le nombre de lignes de la deuxième table.

```
SELECT NOM, NODEPT  
FROM EMPLOYES CROSS JOIN DEPART
```

NOM -----	NODEPT -----
BALZAC	A00
MARTIN	A00
BOULIN	A00
DURAND	A00
VIALA	A00
DELMAS	A00
CAZENEUVE	A00
ZOLA	A00
BALZAC	B00
MARTIN	B00
BOULIN	B00
DURAND	B00
VIALA	B00
DELMAS	B00
CAZENEUVE	B00
ZOLA	B00
BALZAC	E21
MARTIN	E21
BOULIN	E21
DURAND	E21
VIALA	E21
DELMAS	E21
CAZENEUVE	E21
ZOLA	E21

Jointure de plus de deux tables

Il est possible de joindre jusqu'à 256 tables dans une seule requête.

L'utilisation de plusieurs jointures peut être ramenée à une combinaison de jointures indépendantes.

Exemple : Jointure entre les tables A, B et C

1. La première jointure combine les tables A et B pour produire un jeu de résultats
2. Lui-même combiné à la table C dans la deuxième jointure pour produire le jeu de résultats final.

La requête pourra se coder sous la forme :

```
SELECT colonne_1, colonne_2, ..., colonne_n  
FROM Table_1 JOIN Table_2 ON Table_1.colonne_i = Table_2.colonne_j  
      JOIN Table_3 ON Table_2.colonne_x = Table_3.colonne_y
```

La qualification des colonnes (sous forme nomtable.nomcolonne) est nécessaire lorsque deux tables possèdent des colonnes de même nom.

Auto-jointure (cf. réflexive)

Pour trouver des lignes ayant des valeurs en commun avec d'autres lignes de la même table, une table peut être jointe avec une autre instance d'elle-même.

- Des **alias de tables sont nécessaires** pour référencer deux copies de la table : Un alias de table est spécifié dans la clause FROM après le nom de la table.
- Il peut être nécessaire d'utiliser des conditions dans la clause WHERE pour exclure les lignes en double.

Exemple : Liste des employés ayant le même prénom

```
SELECT A.NOM, NODEPT FROM EMPLOYES A ← Noms d'alias  
      JOIN EMPLOYES B ←  
      ON A.PRENOM = B.PRENOM  
      WHERE A.NOM <> B.NOM
```

Ne pas se sélectionner soi-même

11. 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.

Une sous-requête **retourne toujours une seule colonne** avec une ou plusieurs lignes.

Elles doivent être *entre parenthèses à droite de l'opérateur de la clause WHERE*.

Elles peuvent être imbriquées dans d'autres sous-requêtes.

La requête qui contient la sous-requête est appelée *requête externe*.

La sous-requête est appelée *requête interne* ou *SubSelect*.

Les sous-requêtes peuvent être :

- **IMBRIQUEES** : **s'exécutant une fois** lorsque la requête externe s'exécute
- **ou EN CORRELATION** : **s'exécutant une fois pour chaque ligne** renvoyée lors de l'exécution de la requête externe.

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

Déroulement de l'exécution :

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

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

```
-----  
1630
```

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

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.

Résultat 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

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 fonctionne de la manière suivante :

- 1) La ligne en cours du SELECT externe fournit les valeurs au SubSELECT.
- 2) Le SubSELECT est évalué et un résultat est retourné. Le résultat peut être une valeur simple ou plusieurs occurrences d'une seule colonne.
- 3) La clause WHERE du SELECT externe peut maintenant être évaluée en utilisant les valeurs retournées 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 :

- 1) Une ligne de la table EMPLOYEES est lue ; la valeur de WDEPT est transmise à la requête interne.
- 2) 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.
- 3) La clause WHERE de la requête externe est alors évaluée et la ligne est incluse ou non dans le jeu de résultats.
- 4) Le traitement passe à la ligne suivante.

SOUS REQUETE EN CORRELATION : Utilisation de EXISTS et NOT EXISTS

Le mot clé EXISTS 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.

12. L'OPERATEUR UNION

L'opérateur UNION est utilisé pour combiner l'information retrouvée par au moins 2 ordres SELECT.

Syntaxe :

Instruction_select1

UNION [ALL]

Instruction_select2

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.

La liste résultante 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).

Exemple : Liste du département A00

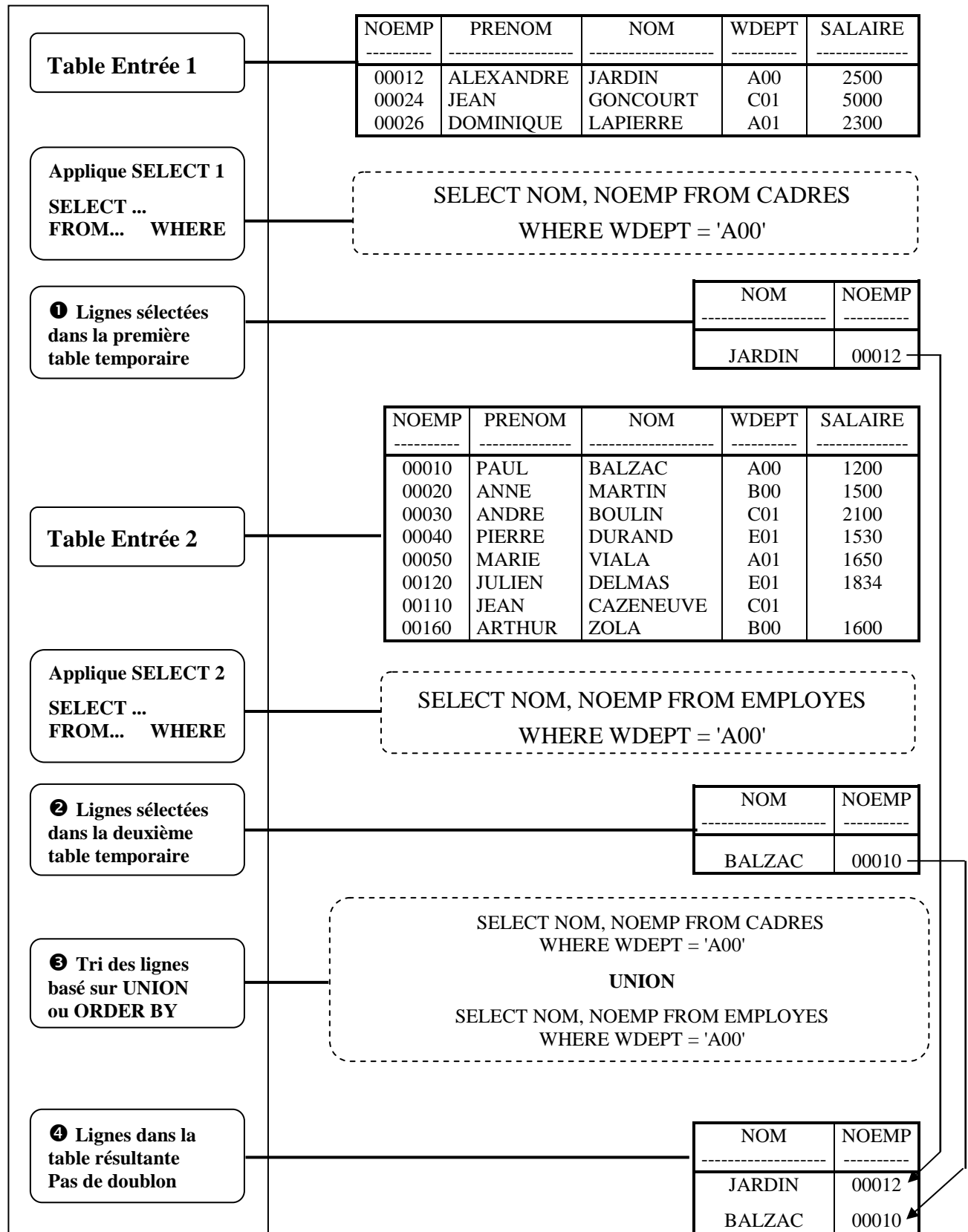
```
SELECT NOM, NOEMP FROM CADRES  
WHERE WDEPT = 'A00'
```

UNION

```
SELECT NOM, NOEMP FROM EMPLOYES  
WHERE WDEPT = 'A00'
```

NOM	NOEMP
-----	-----
JARDIN	00012
BALZAC	00010

Analyse de l'UNION



13. 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.

Etablissement référent
Direction de l'ingénierie Neuilly

Equipe de conception
Groupe d'étude de la filière étude - développement

Remerciements :

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

Date de mise à jour 05/02/2015
afpa © Date de dépôt légal février 15



afpa / Direction de l'Ingénierie 13 place du Général de Gaulle / 93108 Montreuil
Cedex
association nationale pour la formation professionnelle des
adultes
Ministère des Affaires sociales du Travail et de la
Solidarité