

# La commande SELECT

## 1. La commande SELECT

Le SELECT est la commande de base du SQL destinée à **extraire des données** d'une base ou **calculer de nouvelles données à partir d'existantes...**

Voici la syntaxe générale d'une commande SELECT :

```
SELECT [DISTINCT ou ALL] * ou liste de colonnes
FROM nom de table ou de la vue
[WHERE prédicats]
[GROUP BY ordre des groupes]
[HAVING condition]
[ORDER BY ] liste de colonnes
```

**NOTA** : dans cette syntaxe, les mots clef du SQL sont en gras, les paramètres en minuscule et entre crochets on trouve les parties optionnelles

En fait l'ordre SQL SELECT est composé de 6 clauses dont 4 sont optionnelles.

Clauses de l'ordre SELECT :

<b>SELECT</b>	Spécification des colonnes du résultat
<b>FROM</b>	Spécification des tables sur lesquelles porte l'ordre
<b>WHERE</b>	Filtre portant sur les données (conditions à remplir pour que les lignes soient présentes dans le résultat)
<b>GROUP BY</b>	Définition d'un groupe (sous ensemble)
<b>HAVING</b>	Filtre portant sur les résultats (conditions de regroupement des lignes)
<b>ORDER BY</b>	Tri des données du résultat

**NOTA** : La plupart du temps, la difficulté réside dans la compréhension de la différence entre le filtre WHERE et le filtre HAVING. Disons plus pragmatiquement que le filtre WHERE permet de filtrer les données des tables tandis que le filtre HAVING permet de filtrer les données du résultat...

**REMARQUE** : pour spécifier une valeur littérale il faut l'entourer de guillemets simples.

Un premier exemple basique :

SELECT CLI_NOM, CLI_PRENOM FROM T_CLIENT WHERE CLI_CIVILITE = 'M.';	CLI_NOM CLI_PRENOM -----
	DUPONT Alain MARTIN Marc BOUVIER Alain DUBOIS Paul DREYFUS Jean FAURE Alain PAUL Marcel DUVAL Arsène PHILIPPE André CHABAUD Daniel BAILLY Jean-François ...

Permet de trouver les noms et prénoms des clients dont la civilité est 'M.' (monsieur).

**NOTA** : comme tous les paramètres à prendre sous forme de littéraux doivent être exprimés entourés d'apostrophes (simple côtes), il faut dédoubler un tel caractère s'il s'avère présent dans la chaîne utilisé.

## 1.1. L'opérateur \* (étoile)

Le caractère \* (étoile) récupère toutes les colonnes de la table précisée dans la clause FROM de la requête.

Juste après le mot clef SELECT, on précise les colonnes de la table qui doivent être présentées dans la réponse.

L'utilisation du caractère étoile ramène toutes les colonnes de la table dans la réponse. Dans le cas contraire il faut expressément nommer chacune des colonnes et les séparer par des virgules.

<pre>SELECT * FROM T_CLIENT WHERE CLI_CIVILITE = 'M.';</pre>	CLI_ID	CLI_CIVILITE	CLI_NOM	CLI_PRENOM	CLI_ENSEIGNE
	1	M.	DUPONT	Alain	NULL
	2	M.	MARTIN	Marc	Transports MARTIN & fils
	3	M.	BOUVIER	Alain	NULL
	4	M.	DUBOIS	Paul	NULL
	5	M.	DREYFUS	Jean	NULL
	6	M.	FAURE	Alain	Boulangerie du marché
	11	M.	PAUL	Marcel	Cie Internationale des Mach...
	12	M.	DUVAL	Arsène	NULL
	13	M.	PHILIPPE	André	NULL
	16	M.	CHABAUD	Daniel	NULL
	...				

Notons tout de suite la présence à plusieurs reprises du mot clef "NULL" dans la colonne CLI\_ENSEIGNE. Non il ne s'agit pas d'une enseigne particulière, mais simplement de l'absence d'information. Nous verrons que l'absence d'information correspond au marqueur "NULL" qui diffère de la chaîne de caractère vierge ("") ou encore du zéro.

## 1.2. L'opérateur DISTINCT (ou ALL)

Lorsque le moteur construit la réponse, il rapatrie toutes les lignes correspondantes, généralement dans l'ordre où il les trouve, même si ces dernières sont en double, c'est à dire qu'il récupère toutes les lignes (ALL par défaut). C'est pourquoi il est souvent nécessaire d'utiliser le mot clef **DISTINCT** qui permet d'éliminer les doublons dans la réponse.

<pre>SELECT CLI_PRENOM FROM T_CLIENT;</pre>	CLI_PRENOM
	Alain
	Marc
	Alain
	Paul
	Jean
	Alain
	Marcel
	Arsène
	André
	Daniel
	...

<pre>SELECT distinct CLI_PRENOM FROM T_CLIENT;</pre>	CLI_PRENOM
	Alain
	Alexandre
	André
	Arnaud
	Arsène
	Bernard
	Christian
	Christophe
	Daniel
	Denis
	...

### 1.3. L'opérateur AS

Vous pouvez rajouter autant de colonnes que vous le désirez dans la clause de sélection. Lors de l'affichage du résultat, il est possible que certains noms de colonnes soient difficilement compréhensibles. Pour éviter cela, l'opérateur AS sert à donner un alias aux colonnes retournées par la requête.

<pre>SELECT CLI_NOM as NOM, 'homme' as SEXE FROM   T_CLIENT WHERE  TIT_CODE = 'M.' ;</pre>	NOM	SEXE
	-----	----
	DUPONT	homme
	MARTIN	homme
	BOUVIER	homme
	DUBOIS	homme
	DREYFUS	homme
	FAURE	homme
	PAUL	homme
	DUVAL	homme
	PHILIPPE	homme
	CHABAUD	homme
	...	

### 1.4. Opérateur de concaténation

L'opérateur || (double barre verticale) permet de concaténer des champs de type caractères.

<pre>SELECT CLI_CIVILITE    ' '    CLI_PRENOM    ' '    CLI_NOM as NOM FROM   T_CLIENT ;</pre>	NOM
	-----
	M. Alain DUPONT
	M. Marc MARTIN
	M. Alain BOUVIER
	M. Paul DUBOIS
	M. Jean DREYFUS
	M. Alain FAURE
	M. Paul LACOMBE
	Melle. Evelyne DUHAMEL
	Mme. Martine BOYER
	M. Martin MARTIN
	...

Néanmoins on trouve dans certains SGBDR le + comme opérateur de concaténation, comme la fonction CONCAT.

## 1.5. Opérateurs mathématiques de base

On peut utiliser les opérateurs mathématiques de base dans une requête de sélection (+, -, \*, /).

<pre>SELECT PDT_ID, PDT_PUHT * 1.196 AS TARIF_TTC FROM   T_PRODUIT WHERE  PDT_DATE_DEBUT = '2001-01-01';</pre>	PDT_ID	TARIF_TTC
	-----	-----
	1	424,51
	2	482,40
	3	617,47
	4	424,51
	5	463,10
	6	482,40
	7	424,51
	8	540,29
	9	482,40
	10	617,47
	...	

Les opérateurs mathématiques peuvent aussi être appliqués entre colonnes.

<pre>SELECT ID, PUHT, QTE, (PUHT * QTE) AS TOTALHT FROM PRODUIT WHERE PUHT &gt; 200;</pre>	ID	PUHT	QTE	TOTALHT
	-----	-----	-----	-----
	1	225,52	2	451,04
	2	345,67	3	1037,01
	3	123,43	4	493,72
	4	545,45	2	1090,90
	5	456,36	1	456,36
	...			

## 1.6. Particularité du "FROM"

Il est obligatoire de désigner la ou les tables dans la clause FROM. Il est possible également, lors de l'utilisation de plusieurs tables de « surnommer » les tables. Dans ce cas, la syntaxe de la partie FROM de la commande SELECT est la suivante :

FROM *nom\_de\_table surnom*

## 2. La clause ORDER BY

ORDER BY *colonne1* | 1 [ASC ou DESC] [, *colonne2* | 2 [ASC ou DESC] ...

Cette clause permet de définir le tri des colonnes de la réponse, soit en précisant le nom littéral de la colonne, soit en précisant son n° d'ordre dans l'énumération qui suit le mot clef SELECT. ASC spécifie l'ordre ascendant et DESC l'ordre descendant du tri.

ASC ou DESC peut être omis, dans ce cas c'est l'ordre ascendant qui est utilisé par défaut.

Bien que la clause ORDER BY ne soit pas nécessaire, il est souvent utile de trier la réponse en fonction des colonnes. En revanche le temps de réponse s'en ressent souvent. Souvent, le fait de placer DISTINCT suffit, en général, à établir un tri puisque le moteur doit se livrer à une comparaison des lignes mais ce mécanisme n'est pas garanti car ce tri s'effectue dans un ordre non contrôlable qui peut varier d'un serveur à l'autre.

**Attention** : le tri est un tri interne, il ne faut donc placer dans cette clause que les noms des colonnes présentées dans la clause SELECT.

<pre>SELECT CLI_NOM, CLI_PRENOM FROM   T_CLIENT ORDER BY CLI_NOM, CLI_PRENOM;</pre>	CLI_NOM	CLI_PRENOM
	-----	-----
	AIACH	Alexandre
	ALBERT	Christian
	AUZENAT	Michel
	BACQUE	Michel
	BAILLY	Jean-François
	...	

**NOTA** : Un problème, qui n'est pas résolu, est de pouvoir choisir l'ordre des colonnes de la réponse. Sur certains serveurs cela peut être obtenu en plaçant les noms des colonnes à obtenir dans l'ordre où l'on veut les voir apparaître dans la clause SELECT, mais cette possibilité n'est jamais garantie...

**ATTENTION** : la clause ORDER BY est la dernière clause de tout ordre SQL et ne doit figurer qu'une seule fois dans le SELECT, même s'il existe des requêtes imbriquées ou un jeu de requêtes ensemblistes.

### 3. La clause WHERE

WHERE *condition*

La condition doit contenir n'importe quelle expression logique renvoyant une valeur vrai. Si la valeur renvoyée par la condition est fausse, le jeu de résultats ne contiendra pas les données correspondantes.

Voici un exemple simple :

<pre>SELECT CLI_NOM FROM T_CLIENT WHERE CLI_CIVILITE = 'M.';</pre>	CLI_NOM
	-----
	DUPONT
	MARTIN
	BOUVIER
	DUBOIS
	DREYFUS
	FAURE
	LACOMBE
	DUHAMEL
	BOYER
	MARTIN
	...

Le résultat ne contient donc que les clients dont la civilité est 'M.'.

Cependant, certains SGBD ne comprennent pas le type booléen, ce qui fait qu'une requête comme la suivante ne pourra pas toujours fonctionner :

```
SELECT *
FROM T_FACTURE
WHERE FA_ACQUITTEE ;
```

Pour palier au manque de booléen, on utilise soit un littéral (True/False, Vrai/Faux, Oui/Non), soit un numérique avec les valeurs 0 (Faux) et 1 (Vrai). L'avantage des valeurs numériques est que le calcul logique est comparable aux divisions et additions...

Il faudra donc revoir la requête ci-dessus :

```
SELECT *
FROM T_FACTURE
WHERE FA_ACQUITTEE = 1 ;
```

#### 3.1. Opérateurs de comparaison

Dans la clause WHERE, différents opérateurs de comparaisons logiques peuvent être utilisés:

WHERE *valeur1* [NOT et] = ou < ou <= ou > ou >= ou <> *valeur2* [OR ou AND ...]

<pre>SELECT CLI_NOM, CLI_PRENOM FROM T_CLIENT WHERE CLI_NOM &gt;= 'A' AND CLI_NOM &lt; 'E';</pre>	CLI_NOM	CLI_PRENOM
	-----	-----
	DUPONT	Alain
	BOUVIER	Alain
	DUBOIS	Paul
	DREYFUS	Jean
	DUHAMEL	Evelyne
	BOYER	Martine
	DUVAL	Arsène
	DAUMIER	Amélie
	...	

Ici on obtient tous les noms et prénoms des clients dont le nom commence par les lettres A, B, C ou D.

**Attention** : dans certains moteurs de requête SQL l'opérateur « différent de » (<>) s'écrit !=

### 3.2. Opérateur IN

L'opérateur **IN** permet de rechercher si une valeur se trouve dans un ensemble donné, quel que soit le type des valeurs de référence spécifiées (alpha, numérique, date...). Bien entendu, il est possible d'inverser le fonctionnement de l'opérateur IN en lui adjoignant l'opérateur NOT.

<pre>SELECT CLI_CIVILITE, CLI_NOM, CLI_PRENOM FROM   T_CLIENT WHERE  CLI_CIVILITE IN ('Mme.', 'Mlle.');</pre>	CLI_CIVILITE	CLI_NOM	CLI_PRENOM
	-----	-----	-----
	Mme.	BOYER	Martine
	Mme.	GALLACIER	Noëlle
	Mme.	HESS	Lucette
	Mme.	LETERRIER	Monique
	Mme.	MARTINET	Carmen
	Mme.	DAVID	Jacqueline
	Mme.	MOURGUES	Jacqueline
	Mme.	ZAMPIERO	Annick
	Mme.	ROURE	Marie-Louise
	Mme.	DE CONINCK	Patricia
	...		

Le contenu de la parenthèse peut être remplacé par le résultat d'une requête possédant une colonne unique. Dans ce cas on parle de requêtes imbriquées, ce que nous verrons plus loin.

### 3.3. Opérateur BETWEEN

L'opérateur BETWEEN permet de rechercher si une valeur se trouve dans un intervalle donné, quel que soit le type des valeurs de référence spécifiées (alpha, numérique, date...).

Ainsi, la requête vue dans l'exemple d'utilisation des opérateurs de comparaison peut s'écrire :

<pre>SELECT CLI_NOM, CLI_PRENOM FROM   T_CLIENT WHERE  CLI_NOM BETWEEN 'A' AND 'E';</pre>	CLI_NOM	CLI_PRENOM
	-----	-----
	DUPONT	Alain
	BOUVIER	Alain
	DUBOIS	Paul
	DREYFUS	Jean
	DUHAMEL	Evelyne
	BOYER	Martine
	DUVAL	Arsène
	DAUMIER	Amélie
	...	

**NOTA** : les opérateurs IN et BETWEEN sont très pratiques dans le cas où l'on désire effectuer des requêtes où l'utilisateur peut saisir une liste de choix multiples (IN) ou une plage de valeur (BETWEEN).

### 3.4. Opérateur LIKE

L'opérateur LIKE permet d'effectuer une comparaison partielle. Il est surtout employé avec les colonnes contenant des données de type alpha. Il utilise les jokers % et \_ ('pour cent' et 'blanc souligné'). Le joker % remplace n'importe quelle chaîne de caractères, y compris la chaîne vide. Le blanc souligné remplace un et un seul caractère.

<pre>SELECT CLI_NOM, CLI_PRENOM FROM   T_CLIENT WHERE  CLI_NOM LIKE 'B%';</pre>	CLI_NOM	CLI_PRENOM
	-----	-----
	BOUVIER	Alain
	BOYER	Martine
	BAILLY	Jean-François
	BOUCHET	Michel
	BEAUNEE	Pierre
	BERGER	Jean-Pierre
	BOURA	André
	...	

NOTA : l'opérateur LIKE effectue une recherche en tenant compte de la différence entre lettres majuscules et minuscules. Si vous voulez effectuer une recherche en ne tenant aucunement compte de la différence entre majuscules et minuscules, il convient d'utiliser les opérateurs LOWER et UPPER. Mais la plupart du temps, l'utilisation du like dans un SGBDR donné ignore la casse.

### 3.5. Résumé des opérateurs pour les conditions de la clause WHERE

Voici un tableau résumant les principaux opérateurs utilisés pour la construction des prédicats :

opérateurs de comparaisons	= < <= > >=
connecteurs logiques	{OR   AND}
opérateur de négation	NOT
parenthèses	( ... )
opérateurs mathématiques	+ - * /
comparaison logique	IS [NOT] {TRUE   FALSE   UNKNOWN}
comparaison avec valeur	IS [NOT] NULL
intervalle	valeur BETWEEN borne_basse AND borne_haute
comparaison partielle de chaîne de caractères	valeur LIKE motif [ESCAPE échappement]
comparaison à une liste de valeur	valeur [NOT] IN (liste)

## 4. Fonctions diverses

### 4.1. Trantypage à l'aide de la fonction CAST

La fonction CAST permet de changer le type de données d'une colonne afin d'effectuer une comparaison de données de type hétérogène par exemple entre un champ contenant des données numériques et un champ contenant des données de type chaîne de caractères...

Bien entendu il faut qu'un type de donnée puisse être converti dans un autre type (compatibilité de types) afin que la réponse ne soit pas entaché d'erreurs ou d'omissions.

Sa syntaxe est : CAST(*colonne AS nouveau type*).

	FA_CODE	FA_MONTANTHT
	-----	-----
SELECT FA_CODE, FA_MONTANTHT FROM T_FACTURE WHERE CAST(RIGHT(FA_CODE,8) AS INTEGER) > 45000;	R41045001	1024,43
	R41045002	2048,45
	R41045003	4032,34
	R41045004	123,56
	R41045005	45322,45

Exemple 18

	CLI_VILLE	CLI_CP	CLI_CP + 1
	-----	-----	-----
SELECT CLI_VILLE, CLI_CP, CAST(CLI_CP AS INTEGER) + 1 FROM T_CLIENT ;	VERSAILLES	78000	78001
	MONTMAIZIN	11254	11255
	PARIS	75015	75016
	VERGNOLLES	84524	84525
	MARSEILLE	13002	13003
	PARIS	75012	75013
	BONNEUIL CEDEX	94152	94153
	PARIS	75012	75013
	PARIS	75014	75015
	PARIS	75017	75018
	...		

### 4.2. Mise en majuscule / Minuscule

Les opérateurs **LOWER** et **UPPER** permettent de mettre en majuscule ou en minuscule des chaînes de caractères dans les requêtes.

	CLI_NOM	CLI_PRENOM
	-----	-----
SELECT UPPER(CLI_NOM), UPPER(LEFT(CLI_PRENOM, 1))    LOWER(RIGHT(CLI_PRENOM, LEN(CLI_PRENOM) - 1)) FROM T_CLIENT;	DUPONT	Alain
	MARTIN	Marc
	BOUVIERS	Alain
	DUBOIS	Paul
	DREYFUS	Jean
	FAURE	Alain
	...	

**NOTA** : pour effectuer une recherche en ne tenant aucunement compte de la différence entre majuscules et minuscules, il faut utiliser l'opérateur UPPER (ou lower mais attention à la transformation des accents !) :

	CLI_ID	CLI_NOM	CLI_PRENOM	CLI_ENSEIGNE
	-----	-----	-----	-----
SELECT * FROM T_CLIENT where UPPER(CLI_PRENOM) = UPPER(CLI_NOM);	10	MARTIN	Martin	HERMAREX

**NOTA** : certains SGBD permettent de paramétrer l'activation de la recherche systématique des chaînes de caractères sans tenir compte de la casse. Sur d'autres, le paramétrage permet de confondre les lettres accentuées ou non...



### 4.3. Supprimer les blancs (ou tout autre caractères)

La fonction **TRIM** permet de supprimer en tête ou en queue (ou les deux) le blanc ou tout autre caractère spécifié.

TRIM ([LEADING ou TRAILING ou BOTH] [caractère] FROM nom de colonne)

LEADING : suppression en tête

TRAILING : suppression en queue

BOTH : suppression en tête et en queue

**NOTA** : certains serveurs SQL proposent différentes fonctions comme LTRIM et RTRIM pour une suppression des blancs en tête ou en queue.

Dans notre table téléphone, nous voulons supprimer le zéro de tête des n° afin de pouvoir les communiquer aux étrangers qui n'ont pas besoin de composer ce chiffre (ils doivent simplement composer +33 suivi du numéro à 9 chiffres).

	TEL_NUMERO	TEL_INTERNATIONAL
	-----	-----
SELECT TEL_NUMERO, '+33 '    TRIM(LEADING '0' FROM TEL_NUMERO) AS TEL_INTERNATIONAL FROM   T_TELEPHONE;	01-45-42-56-63	+33 1-45-42-56-63
	01-44-28-52-52	+33 1-44-28-52-52
	01-44-28-52-50	+33 1-44-28-52-50
	06-11-86-78-89	+33 6-11-86-78-89
	02-41-58-89-52	+33 2-41-58-89-52
	01-51-58-52-50	+33 1-51-58-52-50
	01-54-11-43-21	+33 1-54-11-43-21
	06-55-41-42-95	+33 6-55-41-42-95
	01-48-98-92-21	+33 1-48-98-92-21
	01-44-22-56-21	+33 1-44-22-56-21
	...	

### 4.4. Extraire une sous chaîne

La fonction **SUBSTRING** permet d'extraire une sous chaîne d'une chaîne de caractère. Elle a besoin de l'indice du premier caractère et du nombre de caractères sur lequel elle doit opérer.

SUBSTRING ( *nom de colonne* FROM n TO m)

Extrait la sous chaîne de nom de colonne en commençant à n sur m caractères.

	CLI_NOM	CLI_PRENOM	INITIALES
	-----	-----	-----
SELECT CLI_NOM, CLI_PRENOM, SUBSTRING(CLI_PRENOM FROM 1 FOR 1)    SUBSTRING(CLI_NOM FROM 1 FOR 1) AS INITIALES FROM   T_CLIENT ;	DUPONT	Alain	AD
	MARTIN	Marc	MM
	BOUVIER	Alain	AB
	DUBOIS	Paul	PD
	DREYFUS	Jean	JD
	FAURE	Alain	AF
	LACOMBE	Paul	PL
	DUHAMEL	Evelyne	ED
	BOYER	Martine	MB
	MARTIN	Martin	MM
	...		

Cet exemple construit les initiales des clients à partir des colonnes CLI\_NOM et CLI\_PRENOM\_CLI.

**Attention**, certains SGBD utilisent la fonction SUBSTR.

## 4.5. Opérateur de traitement des dates

### 4.5.1. Extraire un paramètre temporel d'une date

L'opérateur **EXTRACT** permet d'extraire depuis une date, le jour le mois ou l'année...

**EXTRACT** ( YEAR ou MONTH ou DAY FROM nom de colonne )

Dans la table des réservation on recherche l'identifiant des chambres ayant été réservées au cours du mois de mai de n'importe quelle année et pour 3 personnes.

	CHB_ID
	-----
	1
	5
	6
	8
	11
	12
	16
	17
	18
	20

```
SELECT distinct CHB_ID
FROM TJ_CHB_PLN_CLI
WHERE EXTRACT(MONTH FROM PLN_JOUR) = 5
AND CHB_PLN_CLI_RESERVE = 1
AND CHB_PLN_CLI_NB_PERS = 3;
```

**NOTA** : il est dommage de constater que la fonction EXTRACT du standard SQL, souvent fort utile, est rarement présente dans les moteurs de bases de données. Ni Access, ni Oracle, ni Sybase, ni SQL Server en sont dotés. Seul le middleware BDE de Borland Inprise Corel permet d'exploiter pleinement cette fonction avec les SGBDR Paradox, dBase, FoxPro, InterBase, MSSQL, Sybase, Informix, DB2, Oracle.

Cependant il est courant de trouver des fonctions s'en approchant : Exemple DATEPART dans SQL Server, ou simplement YEAR(), MONTH() ou DAY() comme dans MySQL.

### 4.5.2. Heure et date courante

L'heure courante, la date courante et le combiné date/heure courant peuvent être obtenu à l'aide des fonctions CURRENT\_DATE, CURRENT\_TIME et CURRENT\_TIMESTAMP.

	CHB_ID
	-----
	1
	5
	6
	8
	11
	12
	16
	17
	18
	20

```
SELECT distinct CHB_ID
FROM TJ_CHB_PLN_CLI
WHERE (CHB_PLN_CLI_RESERVE = 1)
AND PLN_JOUR BETWEEN CURRENT_DATE and CURRENT_DATE + 14
AND CHB_PLN_CLI_NB_PERS = 3;
```

Cette requête renvoie les chambres réservées pour 3 personnes entre la date du jour et pour les deux semaines à venir.

Attention : la plupart des SGBDR n'acceptent pas encore cette version normalisée des fonctions de recherche de temps courant. Voici les fonctions spécifiques aux différents serveurs SQL :

Oracle	SYSDATE()
Sybase	GETDATE()
SQL Server	GETDATE()
Access	NOW()
MySQL	NOW()
Paradox (QBE)	TODAY

## 4.6. Opérateurs statistiques

Il est possible de réaliser des comptages statistiques sur les colonnes, à l'aide des opérateurs **AVG** (moyenne), **MAX** (maximum), **MIN** (minimum), **SUM** (total), **COUNT** (nombre). On les appelle aussi **fonctions d'aggrégations**.

SELECT AVG(TRF_CHB_PRIX) as MOYENNE, MAX(TRF_CHB_PRIX) as MAXI, MIN(TRF_CHB_PRIX) as MINI, SUM(TRF_CHB_PRIX) as TOTAL, COUNT(TRF_CHB_PRIX) as NOMBRE FROM TJ_TRF_CHB WHERE TRF_DATE_DEBUT = '2001-01-01';	MOYENNE	MAXI	MINI	TOTAL	NOMBRE
	-----	-----	-----	-----	-----
	406,74	512,00	352,00	7728,00	19

Cette requête calcule la moyenne, le montant maximum, minimum, la totalisation et le nombre des tarifs de chambre pour la date de debut du premier janvier 2001.

**ATTENTION** : nous verrons que l'utilisation des fonctions statistiques nécessite la plupart du temps la mise en place d'une clause de groupage, afin de déterminer quel est le sous ensemble cible d'agrégation pour les calculs.

## 4.7. Autres opérateurs mathématiques (non normalisés)

ABS	valeur absolue	LN	logarithme népérien
MOD	modulo	POWER	logarithme décimal
SIGN	signe	LOG	puissance
SQRT	racine carrée	COS	cosinus
CEIL	plus petit entier	COSH	cosinus hyperbolique
FLOOR	plus grand entier	SIN	sinus
ROUND	arrondi	SINH	sinus hyperbolique
TRUNC	tronqué	TAN	tangente
EXP	exponentielle	TANH	tangente hyperbolique
PI	constante Pi		

Certains sont rarement implémentés du fait que les SGBD sont axés sur l'informatique de gestion, la collecte et le traitement d'informations et non le calcul mathématique.

**Attention** : le nom de certains de ces opérateurs peut différer d'un SGBDR à l'autre.

## 4.9. Autres opérateurs de traitement des chaînes de caractères (non normalisés)

CONCAT	concaténation : équivalent du    - Nota : utiliser de préférence    plus standard. Le + entre colonne alphanumérique peut aussi souvent être utilisé comme opérateur de concaténation, préférez de toutes façons
INITCAP	initiales en lettres capitales
LPAD	complément ou troncature à n position à gauche
LTRIM / RTRIM	suppression en tête/queue d'une chaîne
REPLACE	remplacement
RPAD	complément ou troncature à n position à droite
SOUNDEX	code de consonnance – Attention : phonétique souvent anglaise
INSTR	Position d'une chaîne dans une sous chaîne
LENGTH	longueur de la chaîne
TO_CHAR	numérique sous forme littérale – Attention : souvent en anglais
ASCII	code ASCII d'un caractère
CHR	caractère dont le code ASCII est donné
REVERSE	Inverse l'ordre des caractères d'une chaîne

FLIP	Pivote les parties droite et gauche d'une chaîne par rapport au n° du caractère servant de pivot.
------	---

**Attention** : le nom de certains de ces opérateurs peut différer d'un SGBDR à l'autre.

## 5. Traitement des "valeurs" nulles

**NOTA** : le NULL n'est pas à proprement parler une valeur, mais bien l'absence de valeur, c'est pourquoi nous parlerons de *marqueur* NULL et non de valeur NULL.

Le marqueur NULL pose une quantité de problèmes et nous allons dans ce paragraphe soulever un coin du voile.

### 5.1. Le null n'est ni la chaîne vide, ni le zéro

NULL n'est pas une valeur. C'est un marqueur. Par conséquent le marqueur NULL ne peut jamais être comparé à une valeur.

Recherchons les clients qui n'ont pas d'enseigne.

<pre>SELECT CLI_ID, CLI_NOM FROM T_CLIANT WHERE CLI_ENSEIGNE = " ;</pre>	<pre>CLI_ID  CLI_NOM -----  ----- ...</pre>
--	---

La réponse doit produire une table vide !

Pour contourner ce problème il faut :

- soit penser à enregistrer une chaîne de caractère vide lors de l'insertion des données dans la table
- soit la clause WHERE avec un opérateur spécialisé dans le traitement des valeurs nulles

Voici un extrait de la table T\_LIGNE\_FACTURE

LIF_ID	FAC_ID	LIF_QTE	LIF_REMISE_POURCENT	LIF_REMISE_MONTANT	LIF_MONTANT	LIF_TAUX_TVA
1	1	1,00	15,00	NULL	320,00	18,60
2	3	1,00	NULL	50,00	250,00	18,60
3	3	1,00	NULL	50,00	320,00	18,60
4	3	1,00	NULL	NULL	240,00	18,60
5	5	1,00	NULL	NULL	320,00	18,60
6	5	1,00	NULL	NULL	220,00	18,60
7	7	1,00	NULL	NULL	220,00	18,60
8	7	1,00	NULL	NULL	250,00	18,60
9	7	1,00	NULL	NULL	320,00	18,60
10	7	1,00	NULL	NULL	270,00	18,60

...

Nous voulons calculer le montant total de chacune des lignes de cette table, pour une facture donnée.

La requête pour FAC\_ID = 3 est la suivante :

```
SELECT FAC_ID, SUM(LIF_REMISE_MONTANT) AS TOTAL_REMISE_MONTANT, SUM(LIF_MONTANT) AS
TOTAL_MONTANT
FROM T_LIGNE_FACTURE
WHERE FAC_ID = 3
GROUP BY FAC_ID;
```

Le résultat donne :

FAC_ID	TOTAL_REMISE_MONTANT	TOTAL_MONTANT
3	NULL	810,00

On constate que pour les lignes qui n'ont pas de valeurs renseignées dans la colonne LIF\_REMISE\_MONTANT, le résultat du calcul donne la valeur « null » qui se traduit à l'affichage par... rien !

**NOTA** : en général, pour se sortir de ce mauvais pas, on peut, lors de la création de la base de données, obliger tous les champs de type numérique (réels ou entiers) à ne pas accepter la valeur nulle et prendre par défaut la valeur zéro...

**Attention** : l'arithmétique des nuls est assez particulière... Souvenez vous toujours que les NULL se propagent. Cela est vrai pour les numériques, les dates mais aussi pour les chaînes de caractères. Ainsi SQL opère une distinction entre une chaîne de caractère vide et un champ non renseigné. Dans le cas de la concaténation d'une colonne NULL et d'une colonne proprement renseigné, la valeur renvoyée sera NULL !!!



## 5.2. Opérateurs de traitement des marqueurs NULL

La norme SQL 2 (1992) spécifie une comparaison et différents opérateurs sur les marqueurs NULL :

**IS NULL** / **IS NOT NULL** : teste si la colonne est vide ou non vide.

**COALESCE()** qui recherche la première valeur non vide dans un ensemble

**NULLIF** NULLifie une colonne en fonction d'une valeur donnée

COALESCE ( valeur1, valeur2 [, valeur3] ... )

NULLIF ( nom\_de\_colonne, valeur)

expression IS [NOT] NULL

**NOTA** : **ISNULL** (en un seul mot) est une autre fonction de branchement que l'on rencontre parfois (renvoi une valeur si la valeur est nulle). Dans la même veine, **NVL** ou **VALUE** sont des expressions équivalentes à COALESCE que l'on rencontre sur certains SGBDR.

La requête précédente s'exprime, à l'aide de l'opérateur ISNULL :

```
SELECT FAC_ID, SUM(ISNULL(LIF_REMISE_MONTANT, 0)) AS TOTAL_REMISE_MONTANT,  
       SUM(ISNULL(LIF_MONTANT, 0)) AS TOTAL_MONTANT  
FROM T_LIGNE_FACTURE  
WHERE FAC_ID = 3  
GROUP BY FAC_ID;
```

Le résultat donne :

FAC_ID	TOTAL_REMISE_MONTANT	TOTAL_MONTANT
3	100,00	810,00

**NOTA** : En règle générale, dès que l'on traite des colonnes contenant des valeurs monétaires ou numériques, il est bon de faire en sorte que la colonne soit obligatoire et que par défaut elle soit renseignée à zéro.

Sinon, il faudra faire un usage systématique des fonctions NULLIF ou COALESCE dans tous les calculs et cela grèvera les performances d'exécution !

## 6. Négation de valeurs

C'est l'opérateur **NOT** qui réalise la négation de valeurs et inverse la valeur logique d'un prédicat.

L'opérateur NOT peut être combiné avec la plupart des opérateurs de comparaison. Mais il devient très intéressant lorsqu'il est combiné aux opérateurs IN, BETWEEN, LIKE et NULL

Recherchons par exemple toutes les chambres permettant de recevoir au moins 3 personnes, ne comportant pas le chiffre 4 ni les chambres portant les n° 7 et 13 pour un client particulièrement superstitieux...

	CHB_ID	CHB_NUMERO	CHB_COUCHAGE
	-----	-----	-----
SELECT CHB_ID, CHB_NUMERO, CHB_COUCHAGE FROM T_CHAMBRE WHERE NOT (CAST(CHB_NUMERO AS VARCHAR(10)) LIKE '%4%') AND CHB_NUMERO NOT IN ('7', '13') AND CHB_COUCHAGE >= 3 ;	1	1	3
	5	5	3
	6	6	5
	8	8	3
	11	11	3
	12	12	3
	15	16	3
	16	17	5
	17	18	3
	19	20	3

Nous verrons que le NOT IN est particulièrement précieux dans les requêtes imbriquées.

Nous voulons maintenant le nom des clients qui ne commence pas par 'DU' :

<pre>SELECT CLI_NOM FROM T_CLIENT WHERE CLI_NOM NOT LIKE 'DU%';</pre>	CLI_NOM
	-----
	MARTIN
	BOUVIER
	DREYFUS
	FAURE
	LACOMBE
	BOYER
	MARTIN
	PAUL
	PHILIPPE
	PIERRELAYE
	...

## 7. Conclusion

Curieusement Paradox n'a pas implémenté les opérateurs de récupération des valeurs temporelles courantes alors qu'ils existent en QBE ! Il est facile de s'en passer en passant la date ou l'heure courante en paramètre de la requête, mais tout de même...

En ce qui concerne MS Access, on ne peut qu'être frappé par le fait que la plupart des fonctions de base des requêtes sont incompatible avec la norme. Par exemple le LIKE utilise des jokers différent : \* remplace le % et ? remplace le \_. Cela oblige à utiliser une syntaxe propriétaire qui rend la portabilité des requêtes très difficile d'un SGBD à l'autre.

Plus curieux la plupart des SGBD n'accepte pas l'opérateur de concaténation ||!

Dans Sybase comme SQL Server la fonction modulo s'exprime sous la forme d'un caractère '%' d'ou d'énormes possibilité de confusions entre les caractères joker du like, comme le calcul de pourcentage... A quand une release sur ce sujet ???

Le SGBD le plus proche de la norme est celui de Sybase, suivi de SQL Server. Le plus complet par son jeu de fonction est sans doute Oracle.