La commande SELECT

Syntaxe générale

SELECT [DISTINCT] * ou liste de colonnes
FROMnom de table ou de la vue
[WHERE prédicats]
[GROUPBY ordre des groupes]
[HAVING condition]
[ORDERBY] liste de colonnes

- Le caractère * permet l'affichage de toutes les colonnes de la ou des tables mis en jeux
- Les sections entre crochets[] sont optionnelles

La requête d’interrogation de données

Ou se trouvent les réponses aux questions que je dois me poser pour construire ma requête ?

SELECT "les informations dont j’ai besoin "
FROM "Ou se trouve ces informations et critères dans la base de données "
WHERE " Quelle est le(s) critère(s) de recherche "

Les autres sections nous permettrons d’ajouter une dimension statistique et d’ordre
[GROUP BY ordre des groupes]
[HAVING condition]
[ORDER BY] liste de colonnes

La requête d'interrogation de données

Reprenons notre exemple précédent : j'aimerai connaitre le nom et le prénom des clientes dont le prénom contient le nom Marie, qui se traduit en SQL :

```
1 select nom, prenom
2 from client
3 where (tit_code ='Melle' OR tit_code='Mme.') AND prenom like '%Marie%'
```

CLI_ID	TIT_CODE	NOM	PRENOM	ENSEIGNE
95	M.	AIACH	Alexandre	*NULL*
38	M.	ALBERT	Christian	*NULL*
23	M.	AUZENAT	Michel	*NULL*
88	M.	BACQUE	Michel	GARAGE DU CENTRE
17	M.	BAILLY	Jean-François	Entreprise DUPONT CHAUFFAGE
84	M.	BAVEREL	Frédéric	*NULL*
36	M.	BEAUNEE	Pierre	*NULL*
94	M.	BENATTAR	Bernard	*NULL*

La table originale

La requête d'interrogation de données

les informations dont j'ai besoin

Ou se trouve ces informations et la base de données

```
1 select nom, prenom
2 from client
3 where (tit_code ='Melle' OR tit_code='Mme.') AND prenom like '%Marie%'
```

Quelle est le(s) critère(s) de recherche

CLI_ID	TIT_CODE	NOM	PRENOM	ENSEIGNE
95	M.	AIACH	Alexandre	*NULL*
38	M.	ALBERT	Christian	*NULL*
23	M.	AUZENAT	Michel	*NULL*
88	M.	BACQUE	Michel	GARAGE DU CENTRE
17	M.	BAILLY	Jean-François	Entreprise DUPONT CHAUFFAGE
84	M.	BAVEREL	Frédéric	*NULL*
36	M.	BEAUNEE	Pierre	*NULL*
94	M.	BENATTAR	Bernard	*NULL*

La requête d'interrogation de données

```
1 select nom, prenom
2 from client
3 where (tit_code ='Melle' OR tit_code='Mme.') AND prenom like '%Marie%'
```

CLI_ID	TIT_CODE	NOM	PRENOM	ENSEIGNE
95	M.	AIACH	Alexandre	*NULL*
38	M.	ALBERT	Christian	*NULL*
23	M.	AUZENAT	Michel	*NULL*
88	M.	BACQUE	Michel	GARAGE DU CENTRE
17	M.	BAILLY	Jean-François	Entreprise DUPONT CHAUFFAGE
84	M.	BAVEREL	Frédéric	*NULL*
36	M.	BEAUNEE	Pierre	*NULL*
94	M.	BENATTAR	Bernard	*NULL*

La table originale

nom	prenom
ROURE	Marie-Louise

Le résultat

Voyons maintenant ce qu’il est possible de faire avec le SQL

La commande SELECT

Exemple 1 :

```
SELECT   CLI_NOM, CLI_PRENOM
FROM     CLIENT
WHERE    TIT_CODE = 'M.' ;
```

La table originale

CLI_ID	TIT_CODE	CLI_NOM	CLI_PRENOM	CLI_ENSEIGNE
1	M.	DUPONT	Alain	NULL
2	M.	MARTIN	Marc	Transports MARTIN ...
3	M.	BOUVIER	Alain	NULL
4	M.	DUBOIS	Paul	NULL
5	M.	DREYFUS	Jean	NULL
6	M.	FAURE	Alain	Boulangerie du marc...
7	M.	LACOMBE	Paul	NULL

Le résultat

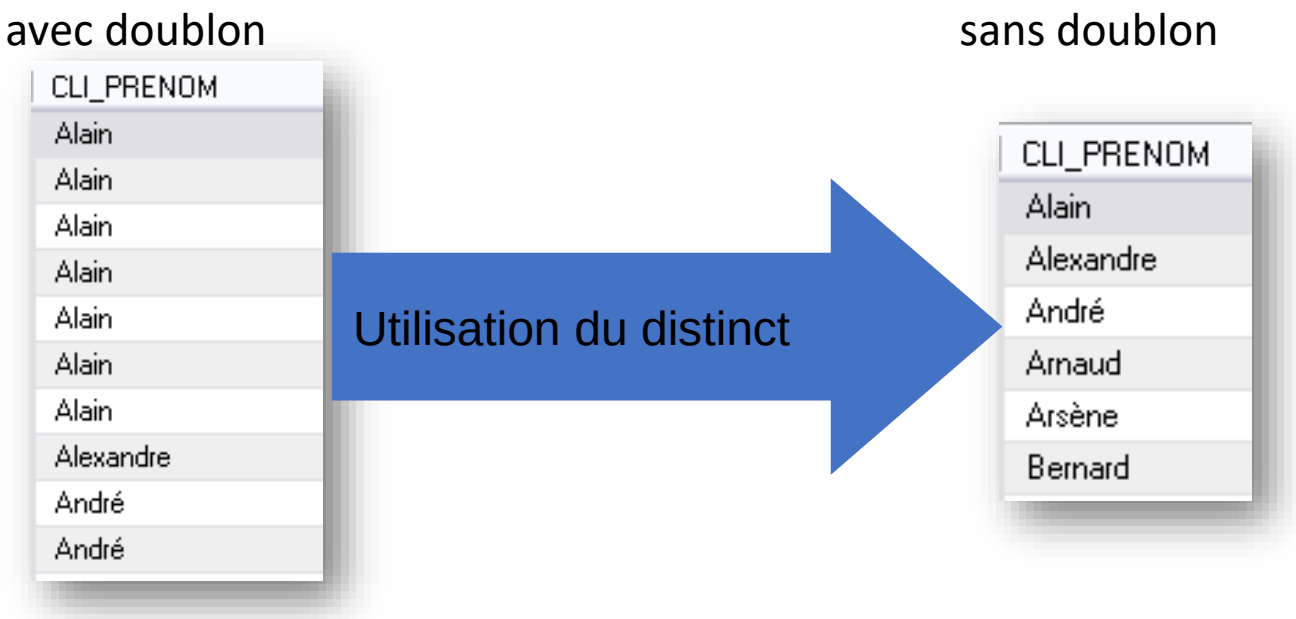
CLI_NOM	CLI_PRENOM
DUPONT	Alain
MARTIN	Marc
BOUVIER	Alain
DUBOIS	Paul
DREYFUS	Jean

La commande SELECT -Distinct-

Résultat sans doublon

```
SELECT CLI_PRENOM
FROM CLIENT
WHERE TIT_CODE = 'M.'
```

```
SELECT distinct CLI_PRENOM
FROM  CLIENT
WHERE TIT_CODE = 'M.'
```



La commande SELECT -AS-

- Dénomination des colonnes résultant d'une requête (l'opérateur AS)

```
SELECT CLI_NOM, CLI_PRENOM
FROM CLIENT
WHERE TIT_CODE = 'M.' ;
```

CLI_NOM	CLI_PRENOM
DUPONT	Alain
MARTIN	Marc
BOUVIER	Alain

```
SELECT CLI_NOM AS"NOM" , CLI_PRENOM AS"PRENOM"
FROM CLIENT
WHERE TIT_CODE = 'M.';
```

NOM	PRENOM
DUPONT	Alain
MARTIN	Marc
BOUVIER	Alain
DUBOIS	Paul
DREYFUS	Jean
FAURE	Alain

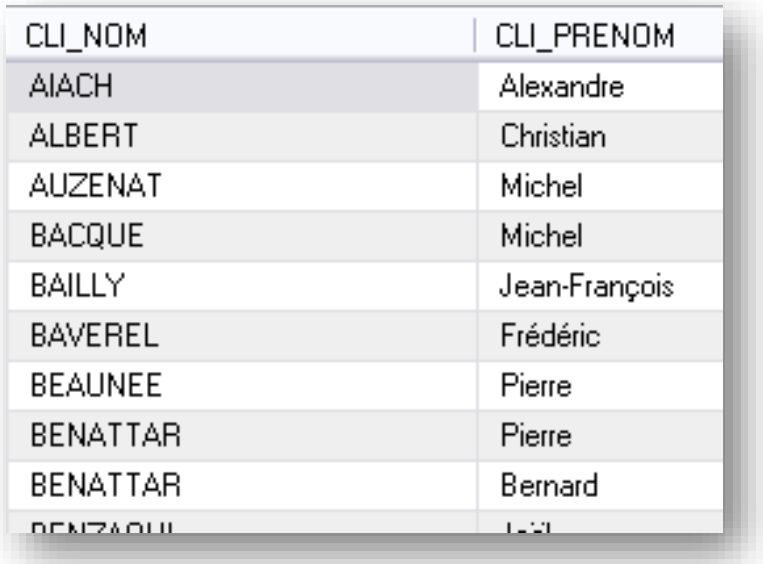
La commande SELECT -Order By-

- Tri du résultat (la clause ORDER BY)

```
SELECT     CLI_NOM, CLI_PRENOM
FROM       CLIENT
ORDER BY   CLI_NOM, CLI_PRENOM DESC;
ou
SELECT     CLI_NOM, CLI_PRENOM
FROM       CLIENT
ORDER BY   1, 2 DESC ;
```

Tri croissant sur le Nom

Tri décroissant sur le prénom



CLI_NOM	CLI_PRENOM
AIACH	Alexandre
ALBERT	Christian
AUZENAT	Michel
BACQUE	Michel
BAILLY	Jean-François
BAVEREL	Frédéric
BEAUNEE	Pierre
BENATTAR	Pierre
BENATTAR	Bernard

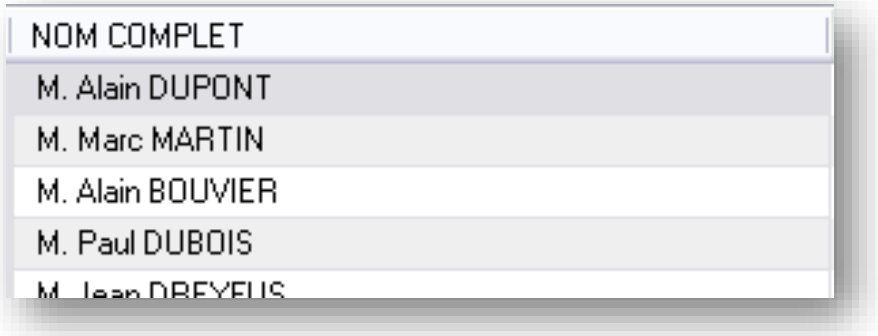
La commande SELECT -concat-

- Concaténer un résultat
  - Opérateur de concaténation ||

```
SELECT ( TIT_CODE||' '||CLI_PRENOM||' '||CLI_NOM ) AS "NOM COMPLET"
FROM CLIENT ;
```

Ou avec la fonction concat()

```
SELECT concat( TIT_CODE,' ',CLI_PRENOM,' ',CLI_NOM ) AS "NOM COMPLET"
FROM CLIENT ;
```



La commande SELECT -calculs-

- Opérateurs Mathématiques

  Il est possible d'effectuer des calculs sur les valeurs de la table lors de l'affichage en utilisant les opérateurs
  +,-,*,/,%

```
SELECT TRF_CHB_PRIX as "HT",
       TRF_CHB_PRIX * 1.206 as "PRIX TTC",
      (TRF_CHB_PRIX * 1.206)-TRF_CHB_PRIX as "Montant tva",
      TRF_CHB_PRIX % 3 as "Modulo 3"
FROM CHB;
```

HT	PRIX TTC	Montant tva	Modulo 3
220.00	265.32000	45.32000	1.00
242.00	291.85200	49.85200	2.00
264.00	318.38400	54.38400	0.00
330.00	397.98000	67.98000	0.00
352.00	424.51200	72.51200	1.00

La clause WHERE

La clause WHERE

Elle permet de restreindre les lignes à afficher en fonction d'un ou plusieurs critères, chaque critère doit être séparé par un opérateur logique :

AND, OR , NOT

Exemple : les informations du client qui se nomme DUPOND Pierre

```
SELECT    CLI_NOM, CLI_PRENOM
FROM      CLIENT
WHERE     CLI_NOM   = 'DUPOND'
      AND CLI_PRENOM = 'Pierre'
```

La clause WHERE

Opérateurs de comparaisons

OPÉRATEUR(S)	RENVOI "TRUE" SI...
<> ou !=	les deux valeurs ne sont pas égales
<	la valeur de gauche est strictement inférieure à celle de droite
>	la valeur de gauche est strictement supérieure à celle de droite
<=	la valeur de gauche est strictement inférieure ou égale à celle de droite
>=	la valeur de gauche est strictement supérieure ou égale à celle de droite
Val BETWEEN x AND y	la valeur testée est située entre deux valeurs données (val >= x AND val=<y)
IN	la valeur testée se situe dans une liste valeurs données
NOT IN	la valeur testée ne se situe pas dans une liste de valeurs données
LIKE	la valeur de gauche correspond à celle de droite (celle de droite peux utiliser le caractère % pour simuler n'importe quel nombre de caractères, et _ pour un seul caractère
NOT LIKE	les deux valeurs ne correspondent pas

La clause WHERE

Comment le moteur SQL traite les critères dans la clause WHERE ?

- Il est à noter que le moteur SQL vérifie les critères ligne par ligne, si l'on reprend la requête suivante :

```
1 select nom, prenom
2 from client
3 where (tit_code ='Melle' OR tit_code='Mme.') AND prenom like '%Marie%'
```

Les critères :

(tit_code='Melle' OR tit_code='Mme') AND prenom like '%Marie%'

- seront vérifiés ligne par ligne

La clause WHERE

Comment le moteur SQL traite les critères dans la clause WHERE ?

CLI_ID	TIT_CODE	NOM	PRENOM	ENSEIGNE
93	Mme.	LEAL	Jany	*NULL*
43	M.	MONTEIL	Jean	*NULL*
5	M.	DREYFUS	Jean	*NULL*
52	M.	FORGEOT	Jean-Bernard	*NULL*
50	M.	SAVY	Jean-Claude	Etude et conseil SAVY frères
79	M.	LEPERCQ	Jean-Claude	*NULL*

(tit_code='Melle' OR tit_code='Mme') => VRAI
AND
prenom like '%Marie%' => FAUX
Donc VRAI and FAUX = FAUX
la ligne ne sera pas sélectionnée

La clause WHERE

Comment le moteur SQL traite les critères dans la clause WHERE ?

CLI_ID	TIT_CODE	NOM	PRENOM	ENSEIGNE
93	Mme.	LEAL	Jany	*NULL*
43	M.	MONTEIL	Jean	*NULL*
5	M.	DREYFUS	Jean	*NULL*
52	M.	FORGEOT	Jean-Bernard	*NULL*
50	M.	SAVY	Jean-Claude	Etude et conseil SAVY frères
79	M.	LEPERCQ	Jean-Claude	*NULL*

(tit_code='Melle' OR tit_code='Mme') => FAUX
AND
prenom like '%Marie%' => FAUX
Donc VRAI and FAUX = FAUX
la ligne ne sera pas sélectionnée

La clause WHERE

Comment le moteur SQL traite les critères dans la clause WHERE ?

CLI_ID	TIT_CODE	NOM	PRENOM	ENSEIGNE
93	Mme.	LEAL	Jany	*NULL*
43	M.	MONTEIL	Jean	*NULL*
5	M.	DREYFUS	Jean	*NULL*
52	M.	FORGEOT	Jean-Bernard	*NULL*
50	M.	SAVY	Jean-Claude	Etude et conseil SAVY frères
79	M.	LEPERCQ	Jean-Claude	*NULL*

…….

Et ainsi de suite sur toutes les lignes de la table

…….

(tit_code='Melle' OR tit_code='Mme') => FAUX
AND
prenom like '%Marie%' => FAUX
Donc FAUX and FAUX = FAUX
la ligne ne sera pas sélectionnée

La clause WHERE : l'opérateur IN / NOT IN

Opérateur IN (comparaison avec une liste)

```
SELECT    TIT_CODE, CLI_NOM, CLI_PRENOM
FROM      CLIENT
WHERE     TIT_CODE IN ('Mme.', 'Melle.');
```

TIT_CODE	CLI_NOM	CLI_PRENOM
Melle.	DUHAMEL	Evelyne
Melle.	DAUMIER	Amélie
Mme.	BOYER	Martine
Mme.	GALLACIER	Noëlle
Mme.	HESS	Lucette
Mme.	LETERRIER	Monique
Mme.	MARTINET	Carmen

L'opérateur NOT IN aura l'effet inverse, dans l'exemple précèdent avec NOT IN, la requête affichera tous les hommes

La clause WHERE intervalle de valeur -Between .. and

Opérateur BETWEEN AND (dans l'intervalle)

```
SELECT CLI_ID , PMT_CODE, FAC_DATE
FROM   FACTURE
WHERE FAC_DATE BETWEEN '1999-01-31' AND '2000-12-31'
order by  FAC_DATE;
```

CLI_ID	PMT_CODE	FAC_DATE
1	CB	1999-01-31
2	CB	1999-01-31
3	CHQ	1999-01-31
5	CB	1999-01-31
6	CHQ	1999-01-31

La clause WHERE utiliser des jokers -LIKE

Comparer des chaînes de caractères (opérateur LIKE)

utilisation des jokers % et _

% remplace n'importe quelle chaîne de caractères, y compris la chaîne vide

_ remplace un et un seul caractère

Par exemple , 't_ _ _ i _ _ %' permet de trouver les mots qui commence par t suivi de 3 caractères, un i en 5eme position suivi de 2 Caractères et quelque soit le nombre de caractères pour finir

La clause WHERE utiliser des jokers -LIKE

Comparer des chaînes de caractères (opérateur LIKE)

utilisation des jokers % et _

- EXEMPLE

```
SELECT CLI_NOM, CLI_PRENOM
FROM CLIENT
WHERE (CLI_NOM LIKE 'A%' ) OR  (CLI_NOM LIKE 'D%')
ORDER BY 1;
```

CLI_NOM	CLI_PRENOM
AIACH	Alexandre
ALBERT	Christian
AUZENAT	Michel
DAUMIER	Amélie
DAVID	Jacqueline

Les chaînes de caractères sont bornées par des SIMPLES quottes
Le LIKE est sensible à la casse(minuscule/majuscule) sous postgres

La clause WHERE -LIKE-

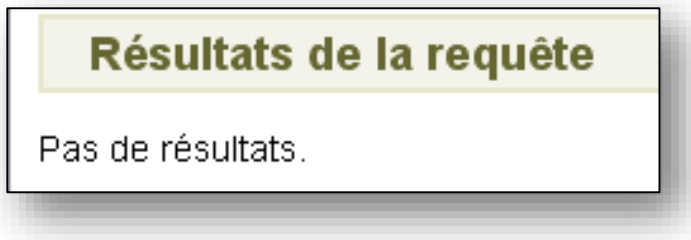
ATTENTION

Dés que vous voulez utiliser un joker (% ou _) vous devez obligatoirement utiliser l'opérateur LIKE et surtout pas le =

L'opérateur LIKE permet de prendre en compte les caractères % et _ comme des jokers alors que le = considérera le % et le _ comme de simples caractères

- EXEMPLE

```
SELECT    CLI_NOM, CLI_PRENOM
FROM      CLIENT
WHERE     CLI_NOM   = 'A%' OR CLI_NOM = 'D%';
```



La clause WHERE traitement des champs vide – IS NULL

Traitement des "*valeurs*" NULLES

Le NULL n'est ni une chaîne vide, ni le zéro

CLI_ID	TIT_CODE	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
7	M.	LACOMBE	Paul	NULL
8	Melle.	DUHAMEL	Evelyne	NULL

Le champs prend la valeur NULL si :

- Il n'est pas renseigné lors de l'insertion d'un nouvel enregistrement dans la table

ET

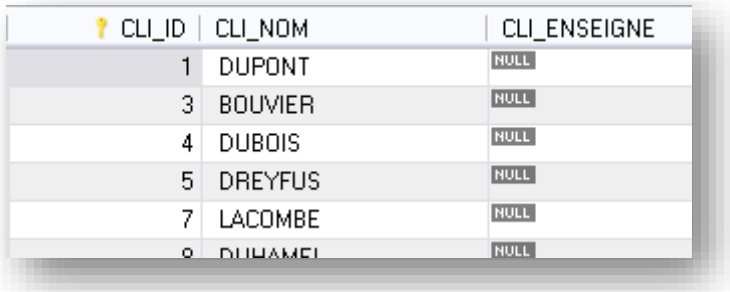
- qu'il a été défini comme tel lors de la création de la table (…..cli_enseigne varchar(45) default NULL,….)

La clause WHERE traitement des champs vide – IS NULL

Traitement des "*valeurs*" NULLES(suite)

- EXEMPLE

```
SELECT  CLI_ID, CLI_NOM, CLI_ENSEIGNE
FROM CLIENT
WHERE  CLI_ENSEIGNE IS NULL;
```



CLI_ID	CLI_NOM	CLI_ENSEIGNE
1	DUPONT	NULL
3	BOUVIER	NULL
4	DUBOIS	NULL
5	DREYFUS	NULL
7	LACOMBE	NULL
8	DUHAMEL	NULL

L'utilisation de IS NOT NULL dans la requête précédente, permettra à la requête de retourner les clients qui ont une enseigne.

Les fonctions

Fonction d'agrégations

Fonction	Description
AVG(col)	Moyenne
COUNT(col)	Nombre d'occurence
MAX(col)	Maximum
MIN(col)	Minimum
SUM(col)	Somme

```
SELECT AVG(TRF_CHB_PRIX) as "MOYENNE",
       MAX(TRF_CHB_PRIX) as "MAXI",
       MIN(TRF_CHB_PRIX) as "MINI"
FROM   TRF_CHB ;
```

MOYENNE	MAXI	MINI
326.370000	512.00	220.00

Il en existe d'autres, voir la documentation : http://docs.postgresql.fr/9.1/functions-aggregate.html

Les fonctions

Fonctions d'agrégations (ou calculs verticaux)

Autre exemple:

Compter le nombre de prix différents

```
SELECT COUNT(TRF_CHB_PRIX) as "Nb prix"
FROM TRF_CHB ;
```

Nb prix
100

Attention il se peut que plusieurs chambres aient le même prix dans ce cas il faut supprimer les doublons sur le prix avant le comptage :

```
SELECT COUNT(distinct TRF_CHB_PRIX) as "Nb prix"
FROM TRF_CHB ;
```

Nb prix
29

Les fonctions

Fonctions d'agrégations (ou calculs verticaux)

ATTENTION, les colonnes vides (NULL) ne sont pas pris en compte dans les fonctions d'agrégat

```
select sum(val1)
from compter ;
```

sum
35

```
select count(val1)
from compter;
```

count
2

```
select count(val1)
from compter
where val1 is NULL;
```

count
0

```
select count(*)
from compter;
```

count
5

id	val1	val2
1	12	Laurent
5	*NULL*	Lisa
4	*NULL*	Valentin
3	*NULL*	Cléo
2	23	*NULL*

Les fonctions

Fonctions d'agrégations (ou calculs verticaux)

Autre exemple:

On peut combiner fonction d'agrégat et calcul horizontaux comme par :

```
SELECT SUM(TRF_CHB_PRIX*qte/1.206) as "Nb prix"
FROM TRF_CHB ;
```

Mais ATTENTION on ne peut pas placer une fonction d'agrégat en paramètre d'une fonction d'agrégat

```
SELECT SUM(MAX (TRF_CHB_PRIX)) as "Nb prix"
FROM TRF_CHB ;
```

Les fonctions

Quelques fonctions de chaînes de caractères

FONCTION	EXEMPLE
CHARACTER_LENGTH OU LENGTH	SELECT CLI_NOM, CHARACTER_LENGTH(CLI_NOM) AS longueur FROM t_client;
LOWER (minuscule) UPPER (MAJUSCULE)	SELECT UPPER(CLI_PRENOM), LOWER(CLI_NOM) FROM T_CLIENT;
REPLACE(str,from_str,to_str)	*Remplace tous le D par % dans le nom des clients* SELECT REPLACE (cli_nom,'D','%') FROMt_client;
TO_CHAR(numeric, text)	Convertit un champ de type numeric en une chaîne to_char(-125.8, '000 999.99') => -000 125.80
TRIM([LEADING|TRAILING|BOTH] [caractère] FROM nom de colonne)	SELECT CHB_NUMERO, TRIM(TRAILING 'er' from CHB_ETAGE) AS "Etage N°" FROM CHAMBRE WHERE CHB_ETAGE like '%er';

Pour en savoir plus sur les fonctions de chaînes caractères RDV http://docs.postgresqlfr.org/9.1/functions-string.html

Les fonctions

Quelques fonctions de dates

FONCTION	EXEMPLE
EXTRACT(type* FROM date)	`SELECT extract(YEAR from fac_date) FROM t_facture;` `SELECT extract(YEAR from date '2006-10-10');`
Date1 - Date2	`SELECT fac_pmt_date- fac_date FROM t_facture ;`
Date1 + INTERVAL 'n type*'	`SELECT date '1998-01-02'+ INTERVAL '31 DAY';`
Date1 - INTERVAL 'n type*'	`SELECT date '1998-01-02'- INTERVAL '31 DAY';`
TO_CHAR(timestamp, text)**	`SELECT to_char(now(), 'DD-MM-YYYY'); =>20-03-2007` `SELECT to_char(now(), 'DD/MM/YYYY'); =>20/03/2007`
NOW()	`SELECT NOW(); =>'2006-10-16 22:41:58'`
CURRENT_DATE()	`SELECT CURRENT_DATE ; =>'2006-10-16'`
age(timestamp)	`SELECT AGE(date '1963-01-15'); =>44 years 2 mons 5 days`

*SECOND, MINUTE, HOUR, MONTH, YEAR, WEEK,DAY, DOY (Le N^{ème} jour de l'année)…

** pour en savoir plus sur les fonctions de formatage : http://docs.postgresqlfr.org/9.1/functions-formatting.html

Pour en savoir plus sur les fonctions de dates : http://docs.postgresqlfr.org/9.1/functions-datetime.html

Les fonctions

Autres exemples de fonctions de dates :

```
SELECT date '2001-10-01' - date '2001-09-28';
```

Cet exemple renvoi 3

```
SELECT (now() - date_embauche)
FROM employe;
```

Renvoi le nombre de jour entre le date courante et la date d'embauche de chaque employé
(sous la forme d'un intervalle : 294 days 10:43:31.974)

```
SELECT extract (day from ( (now() - date_embauche) )
FROM employe;
```

Idem que précédement mais sous la forme d'un nombre de jours (exemple : 294)

Les expressions conditionnelles

CASE

similaire aux instructions if/else des autres langages

- Syntaxe

```
CASE WHEN condition THEN résultat
          [WHEN ...] [ELSE résultat]
END
```

- Exemple

```
SELECT  typ_code,
    CASE WHEN typ_code='TEL'
         THEN 'Téléphone fixe'
              ELSE CASE WHEN typ_code='FAX'
                        THEN 'Télécopie'
                        ELSE 'Téléphone portable'
                   END
    END as "tel"
FROM TELEPHONE
```

tel_id	cli_id	typ_code	numero	localisation
1	1	tel	01-45-42-56-63	domicile
2	1	tel	01-44-28-52-52	bureau
3	1	fax	01-44-28-52-50	bureau
4	3	gsm	06-11-86-78-89	portable

typ_code	tel
tel	Téléphone portable
tel	Téléphone portable
fax	Téléphone portable
gsm	Téléphone portable
tel	Téléphone portable
tel	Téléphone portable

Les expressions conditionnelles

COALESCE

renvoie le premier de ces arguments qui n'est pas NULL.

- Syntaxe

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

- Exemple

```
SELECT cli_nom,cli_enseigne
FROM CLIENT;
```

cli_nom	prenom	cli_enseigne
DUPONT	ALAIN	*NULL*
DURANT	CYRIL	*NULL*
AUZENAT	MICHEL	*NULL*
CHTCHEPINE	DOMINIQUE	HOTEL *** DE LA GARE

```
SELECT cli_nom, COALESCE (cli_enseigne, 'Particulier')
FROM CLIENT
```

cli_nom	coalesce
DUPONT	Particulier
DURANT	Particulier
AUZENAT	Particulier
CHTCHEPINE	HOTEL *** DE LA GARE

Le transtypage

Il est possible de convertir à la volée une valeur avec l'opérateur *:: type*

Exemple 1 :

```
SELECT    CLI_NOM, CLI_PRENOM
FROM      CLIENT
WHERE extract (day  from (CLI_DATENAIS - '1987-11-23' :: date ))
```

Indique au moteur SQL que ce qu'il y dans la chaine de caractére est de type date

- Exemple 2 :

```
SELECT    sum(tariff_chb/1.2) :: numeric(5,2)
FROM      tarif
```

Permet de formater le résultat à 2 chiffres après la virgule et 3 avant : NNN , DD

La Clause GROUP BY

Présentation

Le GROUP BY permet de réaliser des calculs statistiques sur des ensembles qui ont une ou des valeurs communes.

Exemples :

```
SELECT COUNT(*) AS "NOMBRE", CHB_ETAGE
FROM   CHAMBRE
GROUP BY CHB_ETAGE
ORDER BY 2 DESC;
```

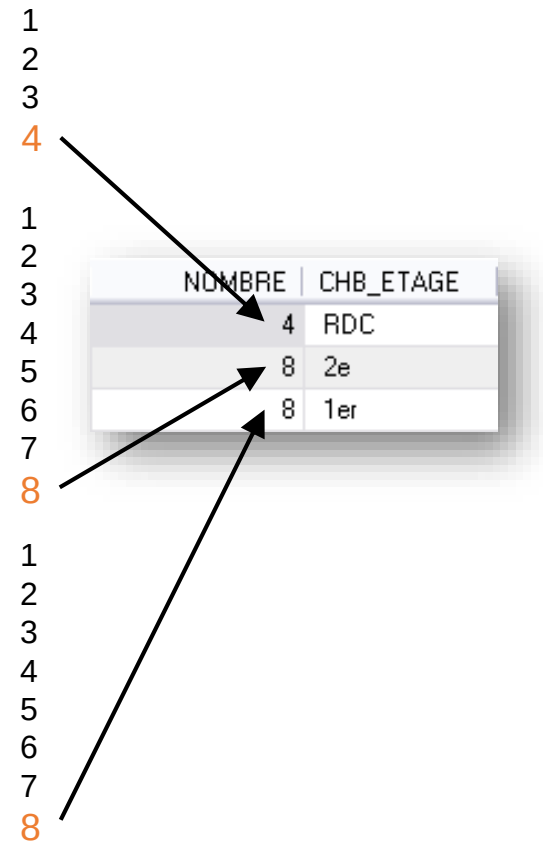
NOMBRE	CHB_ETAGE
4	RDC
8	2e
8	1er

La Clause GROUP BY

Fonctionnement

```
SELECT COUNT(*) AS "NOMBRE",CHB_ETAGE
FROM CHAMBRE
GROUP BY CHB_ETAGE;
```

CHB_NUMERO	CHB_ETAGE
1	RDC
2	RDC
3	RDC
4	RDC
15	2e
16	2e
17	2e
18	2e
19	2e
20	2e
14	2e
21	2e
12	1er
10	1er
9	1er
8	1er
7	1er
6	1er
5	1er
11	1er



La Clause GROUP BY

Remarque importante

Le plupart des SGBDs attendent dans le GROUP BY les mêmes champs que ceux placés dans le SELECT, hormis les fonctions d'agrégat (les calculs).

```
SELECT COUNT(*) AS "NOMBRE",CHB_ETAGE, CHB_NUMERO
FROM CHAMBRE
GROUPBY CHB_ETAGE, CHB_NUMERO ;
```

Mais l'inverse n'est pas vrai

```
SELECT COUNT(*) AS "NOMBRE", CHB_NUMERO
FROM CHAMBRE
GROUPBY CHB_ETAGE, CHB_NUMERO ;
```

La Clause HAVING

Présentation

La clause HAVING permet de filtrer le résultat issu du GROUP BY, de la même manière que le fait le WHERE sur le SELECT

```
SELECT COUNT(*) AS "NOMBRE ", CHB_ETAGE
FROM CHAMBRE
GROUP BY CHB_ETAGE
HAVING  COUNT(*) > 5 ;
```

NOMBRE	CHB_ETAGE
8	1er
8	2e