

FORMATION:

# INTRODUCTION AUX BASES DE DONNEES ET SQL

# Propriété intellectuelle



Paris   Brest   Nantes   Bordeaux  
Toulouse   Montpellier   Lyon   Paris

# Initiation au langage SQL

- 1. Introduction
  - Rappels sur le modèle relationnel
  - Les caractéristiques du langage SQL
- 2. Le Langage d'Interrogation de Données (LID)
  - La sélection de données
  - La gestion des valeurs NULL
  - Les restrictions ou conditions
  - Les tris
  - Les jointures

- 3- Utilisation des fonctions
- 4- Utilisation des opérateurs ensemblistes
  - Union
  - Intersect
  - Minus (ou Except)
- 5. Utilisation des sous interrogations
- 6. Le Langage de Manipulation de Données (LMD)
  - L'Insert
  - L'Update
  - Le Delete

- 7. Notions sur le langage de définition de données (LDD)
  - Création de tables : syntaxe
  - Types de données
  - Types de contraintes
  - Modifier la définition d'une table
  - Supprimer une table
  - Notions sur : Vue et index

- **Rappel sur le modèle relationnel**
- **Les caractéristiques du langage SQL**

Une base de données est un ensemble cohérent d'informations mémorisées sur support informatique.

Ces informations sont accessibles à l'aide d'une application appelée système de gestion de base de données (SGBD). Si ce SGBD est basé sur le modèle relationnel de CODD, on dit qu'il s'agit d'un système de gestion de base de données relationnel (SGBDR).

Pour dialoguer avec un SGBDR on utilise le langage SQL. Ce langage permet de soumettre des requêtes (des questions) au SGBDR.

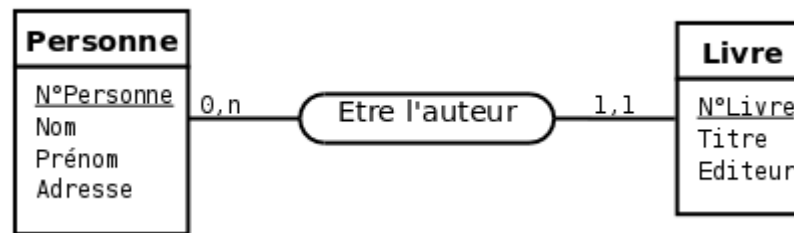


La modélisation relationnelle permet de représenter les relations à l'aide de tables (à deux dimensions)

Un attribut est le nom des colonnes qui constitue la définition d'une table. Il comporte un typage de données.

On appelle tuple (ou n-uplet) une ligne de la table.

La cardinalité d'une relation est le nombre de tuples qui la composent.



La clé principale (ou primaire) d'une relation est l'attribut, ou l'ensemble d'attributs, permettant de désigner de façon unique un tuple.

Une clé étrangère, par contre, est une clé faisant référence à une clé appartenant à une autre table.

- Rappel sur le modèle relationnel
- **Les caractéristiques du langage SQL**

Une requête est un ordre adressé à un SGBD.

Cet ordre peut consister à extraire, à ajouter, à modifier, à administrer les données de la base. De façon générale, l'utilisateur, comme l'administrateur, dialogue avec le SGBD en lui soumettant des requêtes (des questions) et en récupérant en retour des résultats (les réponses).

Le langage SQL est devenu le standard en matière d'interface relationnelle, ceci probablement à cause des raisons suivantes :

- Issu de SEQUEL (interface de System-R), SQL a été développé chez IBM à San José.
- Basé sur des mots clefs anglais explicites, il est relativement simple et facile à apprendre pour des utilisateurs non-informaticiens. Il illustre bien la tendance des langages formels à s'orienter vers un certain "langage naturel".
- SQL est un langage normalisé.

Le standard ANSI a valeur nominative, en principe seulement aux Etats-Unis.

L'équivalent français est la norme AFNOR.

La norme internationale de SQL est la norme ISO (International Standards Organisation) numéro 9075.

Les normes sont accompagnées de niveau qui indiquent le degré d'évolution de SQL. Ainsi l'ISO a défini les normes et les niveaux suivants :

- SQL-86
- SQL-89
- SQL-92 (ou SQL2)
  - Entry
  - Transitional
  - Intermediate
  - Full
- SQL:2003
- SQL:2008
- SQL:2011

La norme définit deux langages SQL:

- un Langage de Manipulation de Données et de modules, (en anglais *SQLDML*), pour déclarer les procédures d'exploitation et les appels à utiliser dans les programmes. On peut également rajouter une composante pour l'interrogation de la base : Langage d'Interrogation de Données.
- un Langage de Définition de Données (en anglais *SQL-DDL*), à utiliser pour déclarer les structures logiques de données et leurs contraintes d'intégrité. On peut également rajouter une composante pour la gestion des accès aux données : Langage de Contrôle de Données : (en anglais *SQL-DCL*)

Les instructions incontournables de SQL sont les suivantes :

Langage SQL			
LMD		LDD	
LID		LCD	
SELECT	INSERT UPDATE DELETE	GRANT REVOKE	CREATE ALTER TRUNCATE DROP RENAME



- **La Sélection de données**
- Gestion des valeurs NULL
- CASE WHEN
- Les restrictions ou conditions
- Les tris
- Les jointures

Le SELECT est la commande de base du SQL.

Elle permet d'extraire des données d'une base ou d'en calculer de nouvelles à partir de données existantes.

*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]*

**SELECT** : Spécification des colonnes du résultat

**FROM** : Spécification des tables sur lesquelles porte le select

**WHERE** : Filtre portant sur les données (conditions à remplir pour faire afficher le résultat)

**GROUP BY** : Définition du sous ensemble

**HAVING** : conditions de regroupement des lignes

**ORDER BY** : Tri des données du résultat

Requête:

```
SELECT CLI_NOM, CLI_PRENOM FROM T_CLIENT
```

Résultat :

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
...	

Il existe plusieurs opérateurs de sélection:

- Le caractère \* récupère toutes les colonnes
- DISTINCT qui permet d'éliminer les doublons dans la réponse
- AS sert à donner un nom à de nouvelles colonnes créées par la requête
- L'opérateur || (double barre verticale) permet de concaténer des champs de type caractères (parfois c'est +)
- On peut utiliser les opérateurs mathématiques de base pour combiner différentes colonnes (+, -, \*, /)

- La Sélection de données
- **Gestion des valeurs NULL**
- CASE WHEN
- Les restrictions ou conditions
- Les tris
- Les jointures

## La gestion du NULL sous Mysql

IFNULL(expr1, expr2)

Si l'argument expr1 n'est pas NULL, la fonction IFNULL() retournera l'argument expr1, sinon elle retournera l'argument expr2. La fonction IFNULL() retourne une valeur numérique ou une chaîne de caractères.



# La gestion des valeurs NULL

```
mysql> SELECT IFNULL(1, 0);
```

```
-> 1
```

```
mysql> SELECT IFNULL(NULL, 10);
```

```
-> 10
```

```
mysql> SELECT IFNULL(1/0, 10);
```

```
-> 10
```

```
mysql> SELECT IFNULL(1/0, 'oui');
```

```
-> 'oui'
```

# La gestion des valeurs NULL

La gestion du NULL sous SqlServer

ISNULL(check\_expr, replacement\_value)

ISNULL remplace NULL avec la valeur spécifiée *replacement\_value*. La fonction retourne la valeur *check\_expr* si celle-ci n'est pas NULL, sinon elle retourne *replacement\_value*.

# La gestion des valeurs NULL

Exemple:

Ici, si le prix est NULL on affiche 0.00

```
SELECT title, ISNULL(price, 0.00) AS price
```

# La gestion des valeurs NULL

La gestion du NULL sous Oracle.

`NVL(val1, replace_with)`

val1 est la valeur testée pour le NULL.

replace\_with est la valeur qui est retournée si val1 est NULL.

Cela s'applique à:

Oracle 8i, Oracle 9i, Oracle 10g, Oracle 11g

# La gestion des valeurs NULL

Exemple:

Ici, si la ville du fournisseur est NULL on affiche 'n/a'

```
select NVL(supplier_city, 'n/a')  
from suppliers;
```

## Exercices:

1. Affichez tous les employés de la société
2. Affichez toutes les catégories de produits
3. Affichez les noms, prénoms et date de naissance et la commission (à 0 si pas de commission) de tous les employés de la société
4. Affichez la liste des fonctions des employés de la société
5. Affichez la liste des pays de nos clients
6. Affichez la liste des localités dans lesquelles il existe au moins un client.

## Exercices:

1. Affichez les produits commercialisés et la valeur de stock par produit ( prix unitaire \*quantité en stock)
2. Affichez le nom, le prénom, l'âge et l'ancienneté des employés, dans la société.
3. Écrivez la requête qui permet d'afficher:

Employé	a un	gain annuel	sur 12 mois
-----	-----	-----	-----
Fuller	gagne	120000	par an.
.....			

- La Sélection de données
- Gestion des valeurs NULL
- CASE WHEN
- **Les restrictions ou conditions**
- Les tris
- Les jointures



Affichage des lignes vérifiant une condition.

Syntaxe :

**SELECT** ....

**WHERE** *condition*

Ex: Select \* from client where numclient=123;

Dans la clause WHERE ou de condition, il est possible d'utiliser tous les opérateurs logiques.

Expression \*op\* Expression

\*op\* peut être:

- =
- <, <=
- >, >=
- !=, <>, ^=

Il existe d'autres opérateurs:

- Opérateur IN
  - WHERE TIT\_CODE IN ('Mme.', 'Melle.')
- Opérateur BETWEEN
  - WHERE CODE\_POSTAL BETWEEN 91000 AND 92000
- Opérateur LIKE
  - WHERE CLI\_NOM LIKE 'B%'
- Comparaison logique
  - IS [NOT] {TRUE | FALSE | UNKNOWN} / IS [NOT] NULL
- Connecteurs logiques
  - {OR | AND}

## Exercices:

1. Affichez le nom de la société et le pays des clients qui habitent à Toulouse.
2. Affichez le nom, prénom et fonction des employés dirigés par l'employé numéro 2.
3. Affichez le nom, prénom et fonction des employés qui ne sont pas des représentants.
4. Affichez le nom, prénom et fonction des employés qui ont un salaire inférieur à 3500.
5. Affichez le nom de la société, la ville et le pays des clients qui n'ont pas de fax.
6. Affichez le nom, prénom et la fonction des employés qui n'ont pas de supérieur.

- La Sélection de données
- Gestion des valeurs NULL
- CASE WHEN
- Les restrictions ou conditions
- Les tris
- Les jointures

# La clause ORDER BY

Cette clause permet de définir le tri des colonnes.

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.

```
SELECT CLI_NOM, CLI_PRENOM  
FROM T_CLIENT  
ORDER BY CLI_NOM, CLI_PRENOM
```

ou

```
SELECT CLI_NOM, CLI_PRENOM  
FROM T_CLIENT  
ORDER BY 1, 2 ASC
```

## Exercices:

1. Trier les employés par nom de salarié en ordre décroissant.
2. Trier les clients par pays.
3. Trier les clients par pays et par ville.



- La Sélection de données
- Gestion des valeurs NULL
- CASE WHEN
- Les restrictions ou conditions
- Les tris
- Les jointures

La jointure entre deux tables.

Soient deux tables *table1* et *table2*. *table1* a les colonnes *col1* et *col2* et *table2* les colonnes *cola*, *colb*.

Supposons que le contenu des tables soit le suivant :

<b>table1</b>	<b>col1</b>	<b>col2</b>	<b>table2</b>	<b>cola</b>	<b>colb</b>
	x	3		a	7
	y	4		b	4

Soit la commande :

```
SELECT col1, cola FROM table1, table2
```

```
WHERE table1.col2=table2.colb
```

Cette requête extrait des données de deux tables : *table1* et *table2* avec une condition entre deux colonnes de tables différentes. On appelle ce type de requête, une jointure.

Comment fonctionne-t-elle ?

Une nouvelle table est construite avec pour colonnes, l'ensemble des colonnes des deux tables et pour lignes le produit cartésien des deux tables :

col1	col2	cola	colb
x	3	a	7
x	3	b	4
y	4	a	7
y	4	b	4

La condition *WHERE col2=colb* est appliquée à cette nouvelle table. On obtient donc la nouvelle table suivante :

col1	col2	cola	colb
y	4	b	4

# Jointures

Il y a ensuite affichage des colonnes demandées (select):

col1	cola
y	b

Syntaxe d'une requête multi-tables.

SELECT *colonne1, colonne2, ...*

FROM *table1, table2, ..., tablep*

WHERE *condition*

ORDER BY ...

La nouveauté ici vient du fait que les colonnes *colonne1, colonne2, ...* proviennent de plusieurs tables *table1, table2, ...*

Si deux tables ont des colonnes de même nom, on lève l'ambiguïté par la notation *tablei.colonnej*. La *condition* peut porter sur les colonnes des différentes tables.

## Fonctionnement:

1. La table produit cartésien de *table1*, *table2*, ..., *tablep* est réalisée. Si  $n_i$  est le nombre de lignes de *tablei*, la table construite a donc  $n_1 * n_2 * \dots * n_p$  lignes comportant l'ensemble des colonnes des différentes tables.
2. La *condition* du WHERE est appliquée à cette table. Une nouvelle table est ainsi produite.
3. Celle-ci est ordonnée selon le mode indiqué dans ORDER.
4. Les colonnes demandées derrière SELECT sont affichées.

## Sans condition (produit cartésien)

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT, T_TELEPHONE
```

## Avec condition

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT, T_TELEPHONE  
WHERE T_CLIENT.CLI_ID = T_TELEPHONE.CLI_ID
```

```
SELECT CLI_NOM, TEL_NUMERO  
FROM T_CLIENT C, T_TELEPHONE T  
WHERE C.CLI_ID = T.CLI_ID AND TYP_CODE = 'FAX'
```

# Jointures

Jointure naturelle	<pre>SELECT ... FROM &lt;table gauche&gt; NATURAL JOIN &lt;table droite&gt; [USING &lt;noms de colonnes&gt;]</pre>
--------------------	--

```
SELECT CLI_NOM, TEL_NUMERO  
  
FROM T_CLIENT NATURAL JOIN T_TELEPHONE
```

La jointure se fait sur la colonne qui porte le même nom dans les deux tables.



# Jointures

Jointure interne	<pre>SELECT ... FROM &lt;table gauche&gt; [INNER]JOIN &lt;table droite&gt; ON &lt;condition de jointure&gt;</pre>
------------------	---

```
SELECT CLI_NOM, TEL_NUMERO  
  
FROM T_CLIENT INNER JOIN T_TELEPHONE  
  
ON T_CLIENT.CLI_ID = T_TELEPHONE.CLI_ID
```

Ici on spécifie le nom de la colonne sur lequel faire la jointure

# Jointures

Jointure externe	<pre>SELECT ... FROM &lt;table gauche&gt; LEFT   RIGHT   FULL OUTER JOIN &lt;table droite&gt; ON condition de jointure</pre>
------------------	--

```
SELECT CLI_NOM, TEL_NUMERO  
  
FROM T_CLIENT C LEFT OUTER JOIN T_TELEPHONE T  
  
ON C.CLI_ID = T.CLI_ID AND TYP_CODE IS NULL
```

Les jointures externes rapatrient les informations disponibles, même si des lignes de table ne sont pas renseignées entre les différentes tables jointes.

SELECT colonnes

FROM TGauche LEFT OUTER JOIN TDroite

ON condition\_de\_jointure

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table TGauche qui n'ont pas été prises en compte au titre de la satisfaction du critère.

SELECT colonnes

FROM TGauche RIGHT OUTER JOIN TDroite

ON condition\_de\_jointure

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table TDroite qui n'ont pas été prises en compte au titre de la satisfaction du critère.

SELECT colonnes

FROM TGauche FULL OUTER JOIN TDroite

ON condition\_de\_jointure

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table TGauche et TDroite qui n'ont pas été prises en compte au titre de la satisfaction du critère.

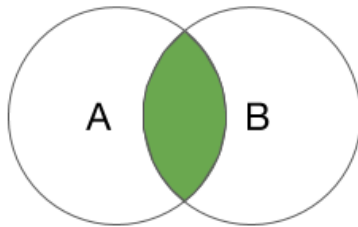
# Jointures

Jointure croisée	<pre>SELECT ... FROM &lt;table gauche&gt; CROSS JOIN &lt;table droite&gt;</pre>
------------------	---

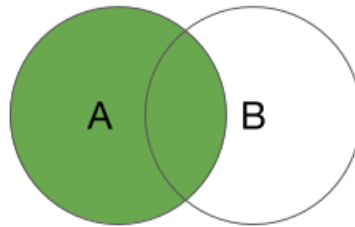
```
SELECT CHB_ID, TRF_DATE_DEBUT, 0 AS TRF_CHB_PRIX  
FROM T_TARIF CROSS JOIN T_CHAMBRE  
ORDER BY CHB_ID, TRF_DATE_DEBUT
```

On fait sciemment le produit cartésien.

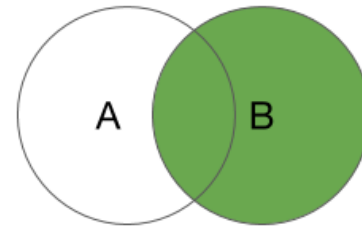
# Jointures



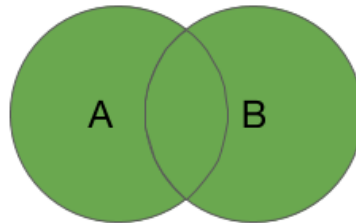
INNER JOIN



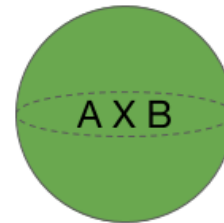
LEFT OUTER JOIN



RIGHT OUTER JOIN



FULL OUTER JOIN



CARTESIAN (CROSS) JOIN

# INNER JOIN

ID_com	Nom
1	Alexia
2	Bastien
3	Jean

ID_Vente	CA	ID_com
001	30,00 €	1
002	50,00 €	3

Nom	CA
Alexia	30,00 €
Jean	50,00 €



# LEFT JOIN

ID_com	Nom
1	Alexia
2	Bastien
3	Jean

ID_Vente	CA	ID_com
001	30,00 €	1
002	50,00 €	3

Nom	CA
Alexia	30,00 €
Jean	50,00 €
Bastien	

# RIGHT JOIN

ID_com	Nom
1	Alexia
2	Bastien
3	Jean

ID_Vente	CA	ID_com
001	30,00 €	1
002	50,00 €	3
004	100,00 €	4

Nom	CA
Alexia	30,00 €
Jean	50,00 €
	100,00 €

# FULL OUTER JOIN

ID_com	Nom
1	Alexia
2	Bastien
3	Jean

ID_Vente	CA	ID_com
001	30,00 €	1
002	50,00 €	3
004	100,00 €	4

Nom	CA
Alexia	30,00 €
Jean	50,00 €
Bastien	
	100,00 €

## Exercices:

1. Affichez le nom, prénom, fonction et salaire des employés qui ont un salaire compris entre 2500 et 3500
2. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités de produits qui ne sont pas d'une des catégories 1, 3, 5 et 7.
3. Affichez le nom du produit, le nom du fournisseur, le nom de la catégorie et les quantités des produits qui ont le numéro fournisseur entre 1 et 3 ou un code catégorie entre 1 et 3, et pour lesquelles les quantités sont données en boîtes ou en cartons.

## Exercices:

4. Écrivez la requête qui permet d'afficher le nom des employés qui ont effectué au moins une vente pour un client parisien.
5. Affichez le nom des produits et le nom des fournisseurs pour les produits des catégories 1, 4 et 7.
6. Affichez la liste des employés ainsi que le nom de leur supérieur hiérarchique.

# Utilisation de fonctions

## Transtypage:

Il permet de changer le type de données d'une colonne afin d'effectuer une comparaison de données de type hétérogène.

```
SELECT CHB_ID, CHB_NUMERO, CHB_POSTE_TEL  
  
FROM T_CHAMBRE  
  
WHERE CAST(CHB_POSTE_TEL AS INTEGER) / 10 >  
CHB_NUMERO
```

## Mise en majuscule / Minuscule:

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

```
SELECT upper(CLI_PRENOM), lower(CLI_NOM)
FROM T_CLIENT
```

CLI_NOM	CLI_PRENOM
-----	-----
ALAIN	dupont
MARC	martin
ALAIN	bouvier
.....	



## Extraire une sous chaîne:

La fonction SUBSTRING permet d'extraire une sous-chaîne d'une chaîne de caractères.


```
SELECT CLI_NOM, CLI_PRENOM,  
CONCAT(SUBSTRING(CLI_PRENOM,1,1) , SUBSTRING  
(CLI_NOM,1,1)) AS INITIALES  
  
FROM T_CLIENT
```

CLI_NOM	CLI_PRENOM	INITIALES
-----	-----	-----
DUPONT	Alain	AD
MARTIN	Marc	MM

# Fonctions

Heure et date courante:

```
SELECT distinct CHB_ID  
FROM CHB_PLN_CLI  
WHERE PLN_JOUR  
BETWEEN CURRENT_DATE  
AND CURRENT_DATE + 14
```

Oracle	SYSDATE CURRENT_DATE
Sybase	GETDATE()
SQL Server	GETDATE()
Access	NOW() Date()
MySQL	NOW() CURRENT_DATE
PostgreSQL	NOW() CURRENT_DATE
Paradox (QBE)	TODAY  <b>diginamic</b> FORMATION <small>Dynamisez votre carrière digitale !</small>

# Fonctions

## Fonctions statistiques:

```
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'
```

ABS	Valeur absolue
MOD	Modulo
SIGN	Signe
SQRT	Racine carrée
CEIL (SQL Server)	Plus grand entier
FLOOR	Plus petit entier
ROUND	Arrondi
TRUNC (SQL Server convert)	Tronqué
EXP	Exponentielle
LN	Logarithme népérien
LOG	Logarithme décimal
POWER	Puissance
COS	Cosinus
COSH	Cosinus hyperbolique
SIN	Sinus
SINH	Sinus Hyperbolique
TAN	tangente
TANH	Tangente hyperbolique
PI	Constante pi

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

CONCAT (Mysql, Oracle, PostgreSQL)	Concaténation : utiliser de préférence    chez Oracle, Sql Server et PostgreSQL
INITCAP (Oracle, PostgreSQL)	Initiales en lettres capitales
LPAD (Mysql, Oracle, PostgreSQL)	Complément ou troncature à n position à gauche
RPAD (Mysql, Oracle, PostgreSQL)	Complément ou troncature à n position à droite
LTRIM / RTRIM	Suppression des espaces en tête/queue d'une chaîne.
REPLACE	Remplacement
SOUNDEX	Code de consonance - Attention : phonétique souvent anglaise.
INSTR (Mysql, Oracle)	Position d'une chaîne dans une sous chaîne
PATINDEX (Sql Server)	Position d'une chaîne dans une sous chaîne

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

LENGTH (Mysql, Oracle, PostgreSQL)	Longueur de la chaîne.
LEN (Sql Server)	Longueur de la chaîne.
ASCII	Code ASCII d'un caractère.
CHR (Oracle, PostgreSQL)	Caractère dont le code ASCII est donné.
CHAR (Mysql et Sql Server)	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.

## Autres opérateurs sur les valeurs temporelles (non normalisés):

+ INTERVAL '1' day (Oracle, Mysql)	Ajoute des jours, des mois, des années à une date
DATEADD (day,1,champ) (Sql Server)	Ajoute des jours, des mois, des années à une date
ADD_MONTHS (Oracle)	Ajoute des mois à une date
NEXT_DAY (Oracle)	Date du prochain jour d'un nom donné
LAST_DAY (Oracle, Mysql)	Renvoie le n° du dernier jour d'un mois d'une date
MONTHS_BETWEEN (Oracle)	Nombre de mois entre deux dates
DATEDIFF (Sql Server)	Différence entre deux dates
DATEPART (Sql Server)	Renvoie la valeur du jour, mois ou année
EXTRACT (Oracle, Mysql)	Renvoie la valeur du jour, mois ou année

# Fonctions

Autres opérateurs sur les valeurs temporelles (non normalisés):

TO_CHAR (Oracle)	Date sous forme littérale - Attention : souvent en anglais
TO_DATE (Oracle)	Convertit une chaîne de caractère en date
Cast ( ... as datetime) (Sql Server)	Convertit une chaîne de caractère en date



## Exercices:

1. Affichez la somme des salaires et des commissions des employés.
2. Affichez la moyenne des salaires et des commissions des employés.
3. Affichez le salaire maximum et la plus petite commission des employés.
4. Affichez le nombre distinct de fonction.

# La clause GROUP BY

La clause GROUP BY est nécessaire dès que l'on utilise des fonctions de calculs statistiques avec des données brutes. Cette clause groupe les lignes sélectionnées en se basant sur la valeur de colonnes spécifiées pour chaque ligne et renvoie une seule ligne par groupe

Sans Group By:

```
SELECT COUNT(CHB_ID) AS NOMBRE, CHB_ETAGE  
FROM T_CHAMBRE
```

NOMBRE CHB\_ETAGE

-----

1 RDC

1 RDC

1 RDC

1 RDC

1 1er

1 1er

1 1er

.....

Avec Group By:

```
SELECT COUNT(CHB_ID) AS NOMBRE, CHB_ETAGE  
FROM T_CHAMBRE  
GROUP BY CHB_ETAGE
```

NOMBRE CHB\_ETAGE

-----

8	1er
8	2e
4	RDC

La clause HAVING remplace le WHERE sur les opérations résultant des regroupements.

```
SELECT SUM(CHB_COUCHAGE) AS NOMBRE, CHB_ETAGE  
FROM T_CHAMBRE  
GROUP BY CHB_ETAGE  
HAVING SUM(CHB_COUCHAGE) >= 20
```

NOMBRE CHB\_ETAGE

-----

23          1er

22          2e

## Exercices:

1. Écrivez la requête qui permet d'afficher la masse salariale des employés par fonction. (4 lignes)

FONCTION	somme
Assistante commerciale	2000.00
Chef des ventes	8000.00
Représentant(e)	20711.00
Vice-Président	10000.00

2. Affichez le numéro de commande de celles qui comportent plus de 5 références de produit. (4 lignes)

NO_COMMANDE	nombre
10657	6
10847	6
10979	6
11077	25

3. Afficher la valeur des produits en stock et la valeur des produits commandés par fournisseur, pour les fournisseurs qui ont un numéro compris entre 3 et 6. (4 lignes)

NO_FOURNISSEUR	stock	commandé
3	18450.00	0.00
4	5005.00	1000.00
5	18650.00	3150.00
6	7822.00	0.00

## Exercices:

4 - Afficher le nom et prénom des employés ayant fait plus de 100 commandes (4 lignes)

NOM	PRENOM	nb_commande
Callahan	Laura	104
Davolio	Nancy	123
Leverling	Janet	127
Peacock	Margaret	156

4 bis - Afficher la société qui a fait plus de 30 commandes (1 ligne)

SOCIETE	nb_commande
Save-a-lot Markets	31



# Opérateurs ensemblistes

Les opérations ensemblistes en SQL sont celles définies dans l'algèbre relationnelle. Elles sont réalisées grâce aux opérateurs:

- UNION
- INTERSECT (ne fait pas partie de la norme SQL et n'est donc pas implémenté dans tous les SGBD)
- EXCEPT (ne fait pas partie de la norme SQL et n'est donc pas implémenté dans tous les SGBD)

Ces opérateurs s'utilisent entre deux clauses *SELECT*.

## L'opérateur UNION :

Cet opérateur permet d'effectuer une UNION des tuples sélectionnés par deux clauses *SELECT* (les deux tables sur lesquelles on travaille devant avoir le même schéma).

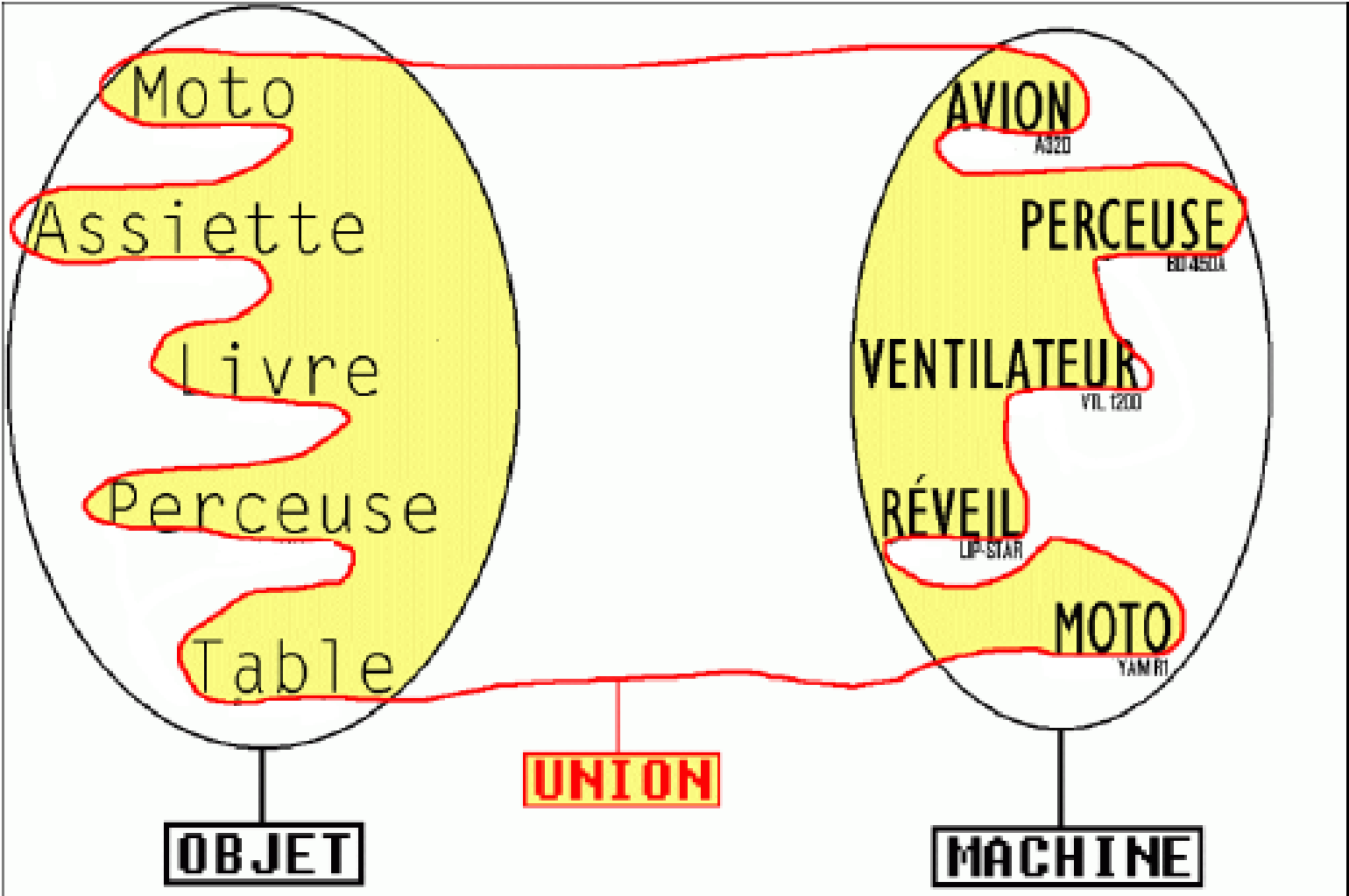
*SELECT ---- FROM ---- WHERE -----*

*UNION*

*SELECT ---- FROM ---- WHERE -----*

Par défaut les doublons sont automatiquement éliminés. Pour conserver les doublons, il est **possible d'utiliser une clause *UNION ALL***.

# Opérateur union



## L'opérateur INTERSECT :

Cet opérateur permet d'effectuer une INTERSECTION des enregistrements sélectionnés par deux clauses *SELECT* (les deux tables sur lesquelles on travaille devant avoir le même schéma).

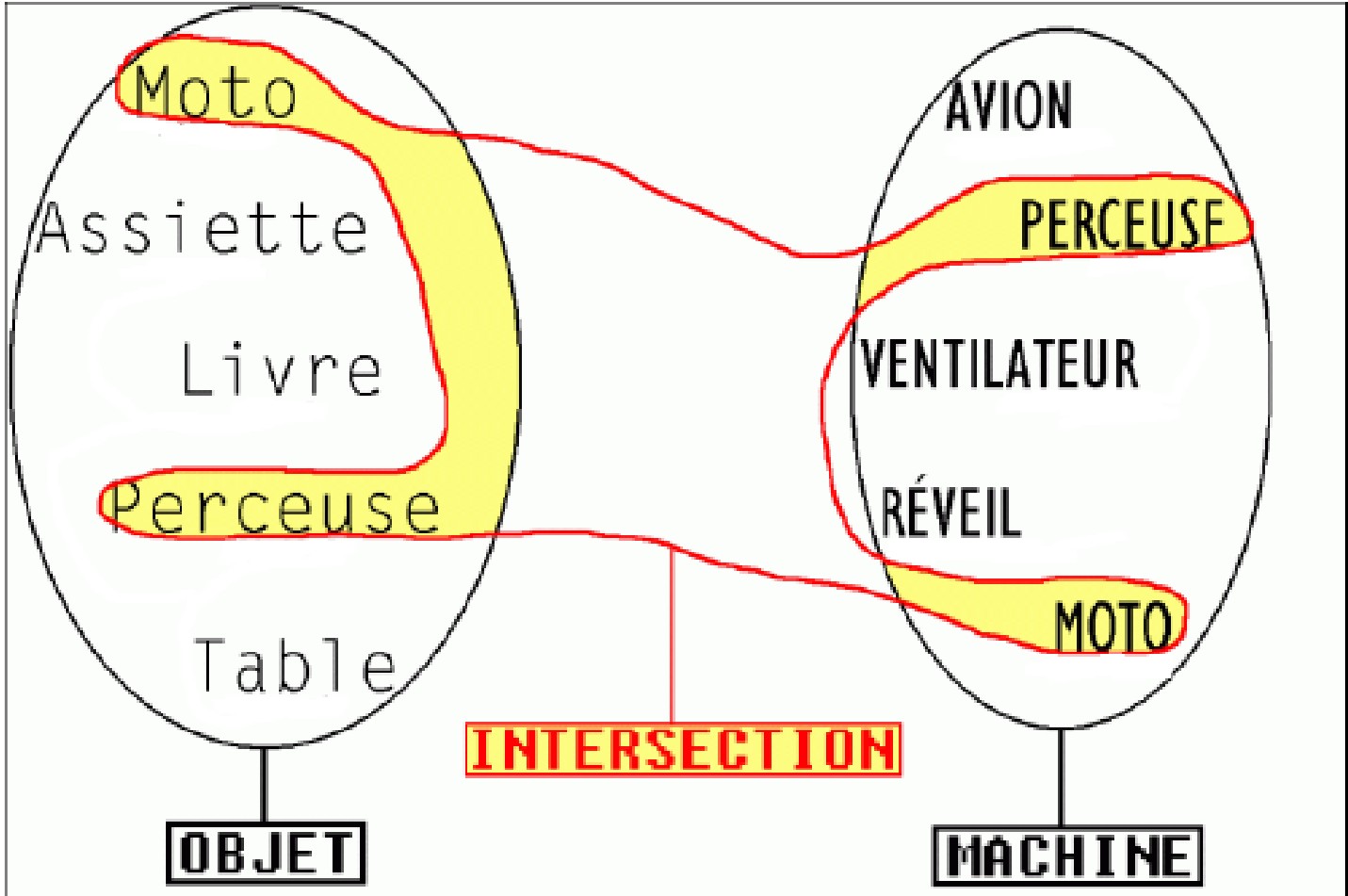
*SELECT ---- FROM ---- WHERE ----- INTERSECT*

*SELECT ---- FROM ---- WHERE -----*

L'opérateur *INTERSECT* n'étant pas implémenté dans tous les SGBD, il est possible de le remplacer par des commandes usuelles:

*SELECT a,b FROM table1 WHERE EXISTS ( SELECT c,d FROM table2  
WHERE a=c AND b=d )*

# Opérateur Intersect



## L'opérateur MINUS (Oracle) ou EXCEPT (PostgreSQL) :

Cet opérateur permet d'effectuer une DIFFERENCE entre les enregistrements sélectionnés par deux clauses SELECT, c'est-à-dire sélectionner les enregistrements de la première table n'appartenant pas à la seconde.

*SELECT a,b FROM table1 WHERE -----*

*MINUS*

*SELECT c,d FROM table2 WHERE -----*

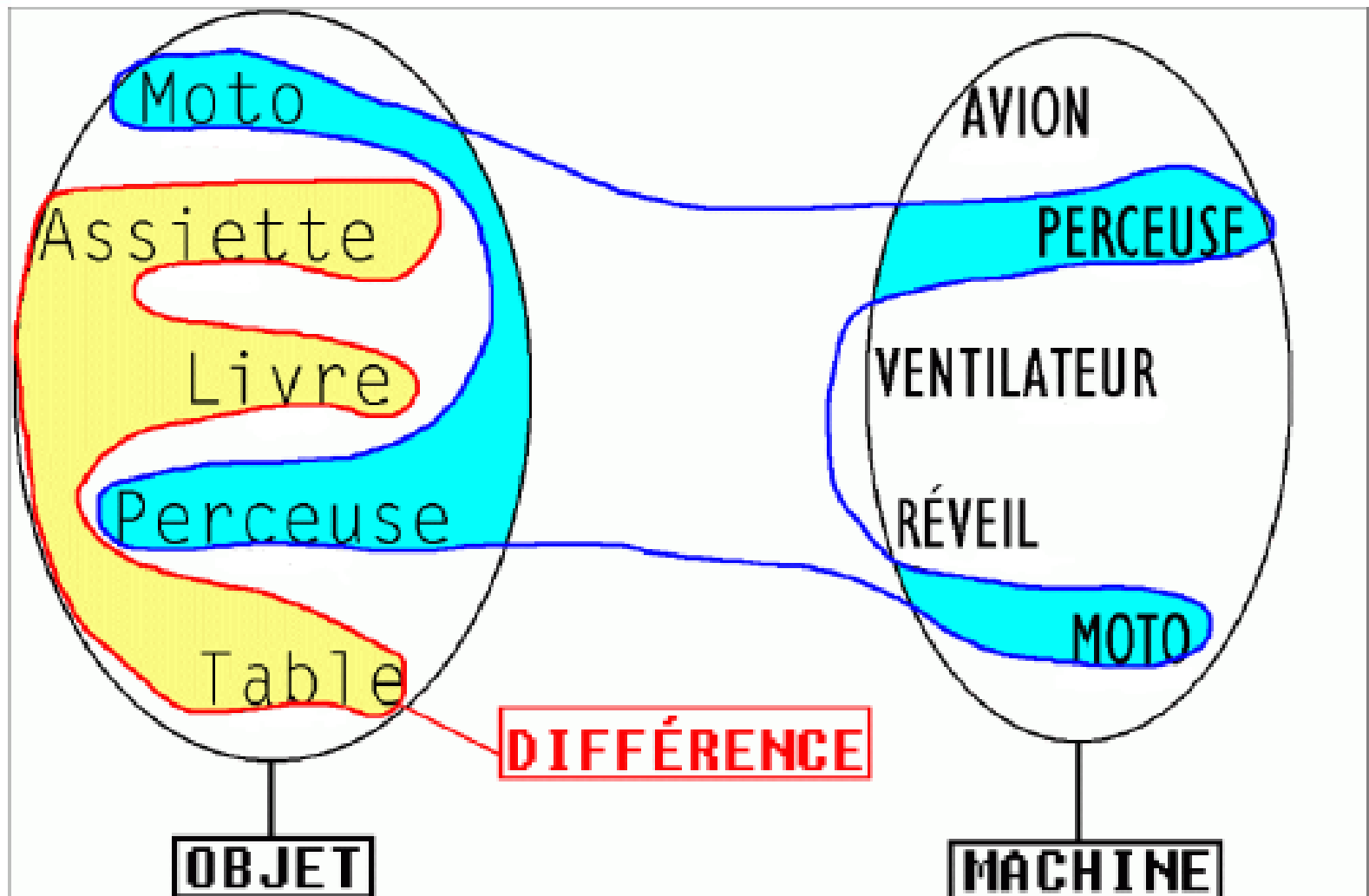
L'opérateur MINUS n'étant pas implémenté dans tous les SGBD, il est possible de le remplacer par des commandes usuelles:

*SELECT a,b FROM table1*

*WHERE NOT EXISTS ( SELECT c,d FROM table2*

87 *WHERE a=c AND b=d )*

# Opérateur Minus (ou Except)





## Exercices:

1. Affichez la société, adresse et ville de résidence pour tous les tiers de l'entreprise. (120 lignes)

SOCIETE	adresse	ville
Exotic Liquids	49 Gilbert St.	London
Nouvelle-Orléans Caiun Delights	P.O. Box 78934	Nouvelle-Orléans
Grandma Kelly's Homestead	707 Oxford Rd.	Ann Arbor
Tokvo Traders	9-8 SekimaiMusashino-shi	Tokvo
Cooperativa de Ouesos 'Las Cabras'	Calle del Rosal 4	Oviedo
Mavumi's	92 SetsukoChuo-ku	Osaka
Pavlova, Ltd.	74 Rose St.Moonie Ponds	Melbourne
Specialty Biscuits, Ltd.	29 Kind's Way	Manchester
PB Knäckebröd AB	Kaloadaanatan 13	Göteborg

2. Affichez toutes les commandes qui comportent en même temps des produits de catégorie 1 du fournisseur 1 et des produits de catégorie 2 du fournisseur 2. (5 lignes)

NO_COMMANDE
11047
10258
10806
11030
11077

3. Affichez la liste des produits que les clients parisiens ne commandent pas. (71 lignes)

ref_produit	nom_produit
1	Chai
2	Chang
3	Aniseed Syrup
4	Chef Anton's Cajun Seasoning
5	Chef Anton's Gumbo Mix
6	Grandma's Boysenberry Spread

# Sous-interrogations

- Dans la clause WHERE
- Dans la clause FROM
- Sous-interrogations synchronisées

Une caractéristique puissante de SQL est la possibilité qu'un prédicat employé dans une clause WHERE (expression à droite d'un opérateur de comparaison) comporte un SELECT emboîté.

### Sous-interrogation à une ligne et une colonne

Dans ce cas, le SELECT imbriqué équivaut à une valeur.

WHERE exp \*op\* (SELECT ...)

Où \*op\* est un des opérateurs : =, !=, <>, <, >, <=, >=

# Syntaxe

Exemple:

Liste des employés travaillant dans le même département que  
MERCIER :

```
SELECT NAME
```

```
FROM EMP
```

```
WHERE DEPT = (SELECT DEPT
```

```
FROM EMP
```

```
WHERE NAME = 'MERCIER')
```

## Sous-interrogation ramenant plusieurs lignes

Une sous-interrogation peut ramener plusieurs lignes à condition que l'opérateur de comparaison admette à sa droite un ensemble de valeurs.

Les opérateurs permettant de comparer une valeur à un ensemble de valeurs sont :

- l'opérateur IN
- les opérateurs obtenus en ajoutant ANY ou ALL à la suite des opérateurs de comparaison classique =, <>, <, >, <=, >=
  - ANY : la comparaison sera vraie si elle est vraie pour au moins un élément de l'ensemble (elle est donc fausse si l'ensemble est vide).
  - ALL : la comparaison sera vraie si elle est vraie pour tous les éléments de l'ensemble (elle est vraie si l'ensemble est vide).

# Syntaxe

## Exemples:

- WHERE exp \*op\* ANY (SELECT ...)
- WHERE exp \*op\* ALL (SELECT ...)
- WHERE exp IN (SELECT ...)
- WHERE exp NOT IN (SELECT ...)

Où \*op\* est un des opérateurs =, !=, <>, <, >, <=, >=

Liste des employés gagnant plus que tous les employés du département 30 :

```
SELECT NOME, SAL
```

```
FROM EMP
```

```
WHERE SAL > ALL (SELECT SAL
```

```
FROM EMP
```

```
WHERE DEPT=30)
```



- Dans la clause WHERE
- Dans la clause FROM
- Sous-interrogations synchronisées

# Dans la clause From

- Syntaxe :

SELECT A.x , A.y, B.z

FROM table A , (SELECT x, z  
FROM table) B

WHERE A.x = B.x

# Dans la clause From

## Exemple:

Part du salaire de chaque employé par rapport à la masse salariale.

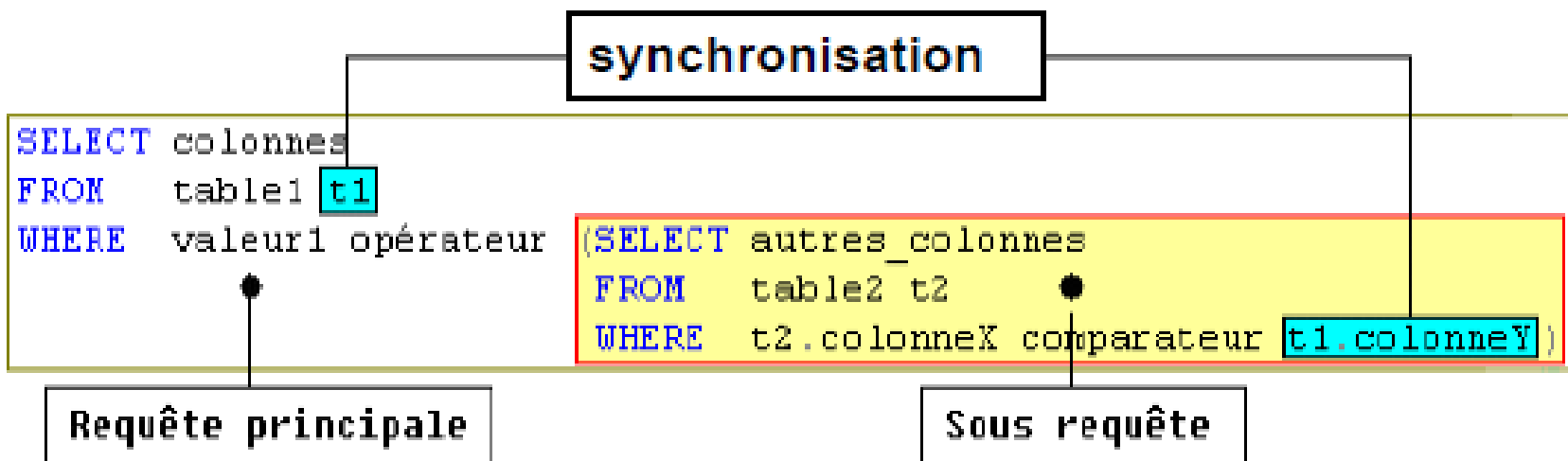
```
SELECT nom, salaire, round(salaire/b.masse*100)
FROM employes, (SELECT sum(salaire) AS masse
                 FROM employes) b;
```

- Dans la clause WHERE
- Dans la clause FROM
- **Sous-interrogations synchronisées**

# Sous-requêtes synchronisées

- Une sous-requête synchronisée est une sous-requête qui s'exécute pour chaque ligne de la requête principale et non une fois pour toute.
- Pour arriver à ce résultat, il suffit de faire varier une condition en rappelant dans la sous requête la valeur de la ou des colonnes de la requête principale qui doit servir de condition.

# Sous-requêtes synchronisées



Exercices:

- 1. Affichez tous les produits pour lesquels la quantité en stock est inférieur à la moyenne des quantités en stock. (46 lignes)

Nom_produit	UNITES_STOCK
Chai	39
Chano	17
Aniseed Svrup	13
Chef Anton's Gumbo Mix	0
Unde Bob's Organic Dried Pears	15
Northwoods Cranberry Sauce	6
Ikura	31
Oueso Cabrales	22
Konbu	24
Tofu	35
...	...

Moyenne : 43,11

2. Affichez toutes les commandes pour lesquelles les frais de ports dépassent la moyenne des frais de ports pour ce client. (304 lignes)  
(bis : afficher les moyenne de frais de port du client sur chaque ligne)

NO_COMMANDE	CODE_CLIENT	NO_EMPLOYE	DATE_COMMANDE	DATE_ENVOI	PORT	Code_client	moy
10248	VINET	5	2017-03-01	2017-03-09	161.90	VINET	58.410000
10250	HANAR	4	2017-03-01	2017-03-09	329.15	HANAR	253.260000
10253	HANAR	3	2017-03-01	2017-03-09	290.85	HANAR	253.260000
10255	RICSU	9	2017-03-01	2017-03-09	741.65	RICSU	493.913000
10257	HILAA	4	2017-03-01	2017-03-09	409.55	HILAA	349.766667

3. Affichez les produits pour lesquels la quantité en stock est supérieure à la quantité en stock de chacun des produits de catégorie 3. (14 lignes)  
(bis: ajouter le max de la catégorie 3, pour vérifier)

NOM_PRODUIT	UNITES_STOCK	MAX
Grandma's Bovsenberrv Soread	120	76
Oueso Mancheco La Pastora	86	76
Gustaf's Knäckebröd	104	76
Geitost	112	76
Sasquatch Ale	111	76
Inlaod Sill	112	76



Exercices:

4. Affichez les produits, fournisseurs et unités en stock pour les produits qui ont un stock inférieur à la moyenne des stocks des produits pour le même fournisseur. (34 lignes)

NOM_PRODUIT	SOCIETE	UNITES_STOCK
Chano	Exotic Liquids	17
Aniseed Svrup	Exotic Liquids	13
Chef Anton's Gumbo Mix	Nouvelle-Orléans Caiun Delights	0
Louisiana Hot Spiced Okra	Nouvelle-Orléans Caiun Delights	4
Uncle Bob's Organic Dried Pears	Grandma Kellv's Homestead	15
Northwoods Cranberry Sauce	Grandma Kellv's Homestead	6

5. Affichez les employés avec leur salaire et le % par rapport au total de la masse salariale par fonction. (9 lignes)

NOM	pourcentage	fonction
Fuller	100 %	Vice-Président
Buchanan	100 %	Chef des ventes
Callahan	100 %	Assistante commerciale
Dodsworth	13 %	Représentant(e)
Kino	14 %	Représentant(e)
Suvama	15 %	Représentant(e)
Peacock	17 %	Représentant(e)
Davolio	19 %	Représentant(e)
Leverlino	21 %	Représentant(e)

# Le Langage de Manipulation de Données (LMD)

# LMD

Le langage de manipulation de données (LMD) est le langage permettant de modifier les informations contenues dans la base.

Il existe trois commandes SQL permettant d'effectuer les trois types de modification des données :

- INSERT = ajout de lignes
- UPDATE = mise à jour de lignes
- DELETE = suppression de lignes

Ces trois commandes travaillent sur la base telle qu'elle était au début de l'exécution de la commande. Les modifications effectuées par les autres utilisateurs entre le début et la fin de l'exécution ne sont pas prises en compte (même pour les transactions validées).

# Insertion

```
INSERT INTO table (col1, ..., coln )
```

```
VALUES (val1, ..., valn )
```

ou

```
INSERT INTO table (col1, ..., coln )
```

```
SELECT ...
```

table est le nom de la table sur laquelle porte l'insertion.

col1, ..., coln est la liste des noms des colonnes pour lesquelles on donne une valeur. Cette liste est optionnelle. Si elle est omise, le SGBD prendra par défaut l'ensemble des colonnes de la table dans l'ordre où elles ont été données lors de la création de la table. Si une liste de colonnes est spécifiée, les colonnes ne figurant pas dans la liste auront la valeur NULL.

# Insertion

```
INSERT INTO dept
```

```
VALUES (10, 'FINANCES', 'PARIS')
```

```
INSERT INTO dept (lieu, nomd, dept)
```

```
VALUES ('GRENOBLE', 'RECHERCHE', 20)
```

```
INSERT INTO PARTICIPATION (MATR, CODEP)
```

```
SELECT MATR, 10
```

```
FROM EMP
```

```
WHERE NOM = 'MARTIN'
```

# Modification

La commande UPDATE permet de modifier les valeurs d'un ou plusieurs champs, dans une ou plusieurs lignes existantes d'une table.

```
UPDATE table
SET col1 = exp1, col2 = exp2, ...
WHERE prédicat
ou
UPDATE table
SET (col1, col2,...) = (SELECT ...)
WHERE prédicat
```

*table* est le nom de la table mise à jour; *col1*, *col2*, ... sont les noms des colonnes qui seront modifiées; *exp1*, *exp2*,... sont des expressions.

Elles peuvent aussi être un ordre SELECT renvoyant les valeurs attribuées aux colonnes (deuxième variante de la syntaxe).

Les valeurs de *col1*, *col2*... sont mises à jour dans toutes les lignes satisfaisant le prédicat.

La clause WHERE est facultative. Si elle est absente, toutes les lignes sont mises à jour.

# Modification

Faire passer MARTIN dans le département 10:

```
UPDATE EMP
```

```
SET DEPT = 10
```

```
WHERE NOM = 'MARTIN'
```

Donner à CLEMENT un salaire 10 % au dessus de la moyenne des salaires des secrétaires:

```
UPDATE EMP
```

```
SET SAL = (SELECT AVG(SAL) * 1.10
```

```
FROM EMP
```

```
WHERE POSTE = 'SECRETAIRE')
```

```
WHERE NOM = 'CLEMENT'
```



# Suppression

L'ordre DELETE permet de supprimer des lignes d'une table.

DELETE FROM table

WHERE prédicat

La clause WHERE indique quelles lignes doivent être supprimées.

ATTENTION: cette clause est facultative; si elle n'est pas précisée, **TOUTES LES LIGNES DE LA TABLE SONT SUPPRIMEES !** (heureusement qu'il existe ROLLBACK !)

# Suppression

DELETE FROM dept

WHERE dept = 10

DELETE FROM dept <-- *j'efface toutes les lignes*

## Exercices:

1. Insérez une nouvelle catégorie de produits nommée «fruits et légumes », en respectant les contraintes.
2. Créez un nouveau fournisseur « Grandma » (no\_fournisseur = 30) avec les mêmes coordonnées que le fournisseur « Grandma Kelly's Homestead ».
3. Attribuer les produits de « Grandma Kelly's Homestead » au nouveau fournisseur créé (« Grandma »).
4. Supprimez l'ancien fournisseur «Grandma Kelly's Homestead» .

# Notions sur le Langage de Définition de Données (LDD)

# Types de données

**CHAR, CHARACTER, VARCHAR, NCHAR, NVARCHAR**

**DECIMAL, NUMERIC, FLOAT, REAL, DOUBLE, SMALLINT, INTEGER**

**BIT, NBIT, BLOB, IMAGE**

**TIMESTAMP, DATE, TIME, INTERVAL**

# Création de table

```
CREATE TABLE table  
  
(colonne1 type1 contraintes,  
colonne2 type2 contraintes,  
  
.....  
  
.....  
  
, contraintes de table)
```

table est le nom que l'on donne à la table;

colonne1, colonne2, ... sont les noms des colonnes ;

type1, type2, ... sont les types des données qui seront contenues dans les colonnes.

# Contraintes

- Contrainte de colonne:

[NOT] NULL | UNIQUE | PRIMARY KEY | CHECK ( prédicat\_de\_colonne ) |  
FOREIGN KEY [colonne] REFERENCES table (colonne)

- Contrainte de table:

CONSTRAINT nom\_contrainte UNIQUE | PRIMARY KEY ( liste\_colonne ) |  
CHECK ( prédicat\_de\_table ) | FOREIGN KEY liste\_colonne REFERENCES  
nom\_table (liste\_colonne)

# Contraintes

Une colonne peut donc recevoir les contraintes suivantes :

NULL / NOT NULL : précise si la colonne est obligatoire.

DEFAULT : valeur par défaut qui est placée dans la colonne lors des insertions et de certaines opérations particulières, lorsque l'on a pas donné de valeur explicite à la colonne.

PRIMARY KEY : précise si la colonne est la clef de la table.

ATTENTION : nécessite que la colonne soit NOT NULL

UNIQUE : les valeurs de la colonne doivent être uniques (pas de doublon).

CHECK : permet de préciser un prédicat qui acceptera la valeur s'il est vérifié.

FOREIGN KEY : permet, pour les valeurs de la colonne, de faire référence à des valeurs existantes dans une colonne d'une autre table. Ce mécanisme s'appelle intégrité référentielle.



# Création de table

Exemple:

```
CREATE TABLE article (  
    ref CHAR(10) NOT NULL,  
    prix DECIMAL(9,2),  
    datemaj DATE  
)
```

# Création de table

```
CREATE TABLE T_CLIENT (  
    CLI_ID INTEGER NOT NULL PRIMARY KEY,  
    CLI_NOM CHAR(32) NOT NULL,  
    CLI_PRENOM VARCHAR(32)  
)
```

# Création de table

```
CREATE TABLE T_VOITURE (  
    VTR_ID INTEGER NOT NULL PRIMARY KEY,  
    VTR_MARQUE CHAR(32) NOT NULL,  
    VTR_MODELE VARCHAR(16),  
    VTR_IMMATRICULATION CHAR(10) NOT NULL UNIQUE,  
    VTR_COULEUR CHAR(5) CHECK (VALUE IN ('BLANC', 'NOIR', 'ROUGE', 'VERT', 'BLEU'))  
)
```

```
CREATE TABLE T_CLIENT (  
    CLI_ID INTEGER NOT NULL PRIMARY KEY,  
    CLI_NOM CHAR(32) NOT NULL CHECK (SUBSTRING(VALUE, 1, 1) <> ' '),  
    CLI_PRENOM VARCHAR(32) REFERENCES TR_PRENOM (PRN_PRENOM)  
)
```

# Modification de table

ALTER TABLE nom\_table

Ajout d'une colonne - ADD:

ALTER TABLE table

ADD (col1 type1, col2 type2, ...)

Modification d'une colonne - MODIFY:

ALTER TABLE table

MODIFY (col1 type1, col2 type2, ...)

Suppression d'une colonne - DROP COLUMN:

ALTER TABLE table

DROP COLUMN col

La colonne supprimée ne doit pas être référencée par une clé étrangère ou être un index.

# Modification de table

Ajouter, supprimer ou renommer une contrainte:

Des contraintes d'intégrité peuvent être ajoutées ou supprimées par la commande ALTER TABLE. On ne peut ajouter que des contraintes de table.

```
ALTER TABLE EMP
```

```
DROP CONSTRAINT NOM_UNIQUE
```

```
ALTER TABLE EMP
```

```
ADD CONSTRAINT SAL_MIN CHECK(SAL + NVL(COMM,0) > 1000)
```

```
ALTER TABLE EMP
```

```
MODIFY CONSTRAINT SAL_MIN DISABLE
```

```
ALTER TABLE EMP
```

```
RENAME CONSTRAINT NOM1 TO NOM2
```

# Modification de table

```
ALTER TABLE T_CLIENT  
ADD CLI_PRENOM VARCHAR(25)
```

```
ALTER TABLE T_CLIENT  
ADD CLI_DATE_NAISSANCE DATE
```

```
ALTER TABLE T_CLIENT  
ADD CONSTRAINT CHK_DATE_NAISSANCE  
CHECK (CLI_DATE_NAISSANCE BETWEEN '1880-01-01' AND '2020-01-01')
```

# Suppression de table

```
DROP TABLE nom_objet
```

```
DROP TABLE TMP_IMP_DATE
```

Dans PostgreSQL :

DROP TABLE supprime tout index, règle, déclencheur ou contrainte qui existe sur la table cible. Néanmoins, pour supprimer une table référencée par une vue ou par une contrainte de clé étrangère d'une autre table, CASCADE doit être ajouté. (CASCADE supprime complètement une vue dépendante mais dans le cas de la clé étrangère, il ne supprime que la contrainte, pas l'autre table.)

```
CREATE DATABASE database_name
```

```
DROP DATABASE database_name
```



## Exercices:

1. Créez une table pays avec 2 champs : code pays (4 caractères, clé primaire), nom pays (40 caractères maximum)
2. Ajoutez une colonne courriel (75 caractères possibles) à la table CLIENTS. Puis modifiez la pour passer à 60 caractères. Pour finir, supprimez cette colonne.
3. Créez une vue qui affiche le nom de la société, l'adresse, le téléphone et la ville des clients qui habitent à Toulouse, Strasbourg, Nantes ou Marseille.

MERCI POUR VOTRE ATTENTION