UNIVERSITÉ POLYTECHNIQUE DE BINGERVILLE

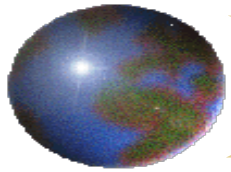
BASE DE DONNEES

Chapitre 4.

Le langage SQL (Structured Query Language)

Partie 2

Dr. Coulibaly Tiékoura



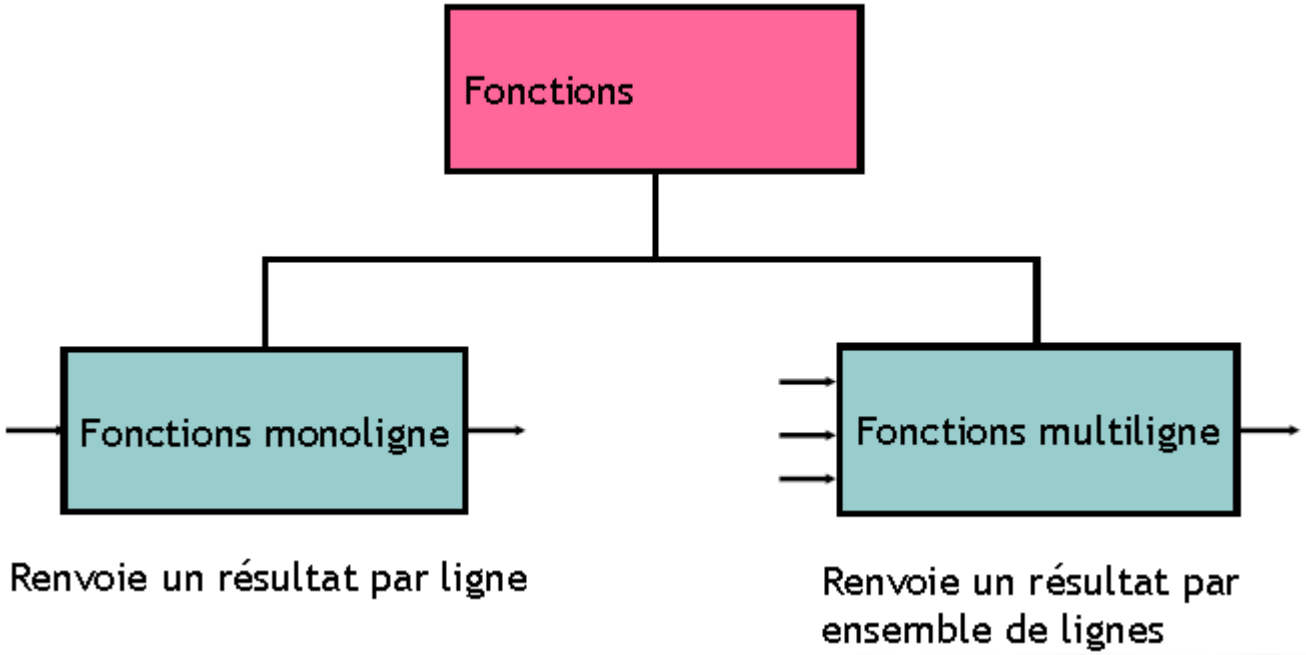
Plan du Chapitre 4

V-
Les fonctions (monoligne et de groupe)

V- Les fonctions (monoligne et de groupe)

5.1. LES FONCTIONS MONOLIGNES

On distingue 2 types de fonctions SQL:



Les fonctions monoligne :

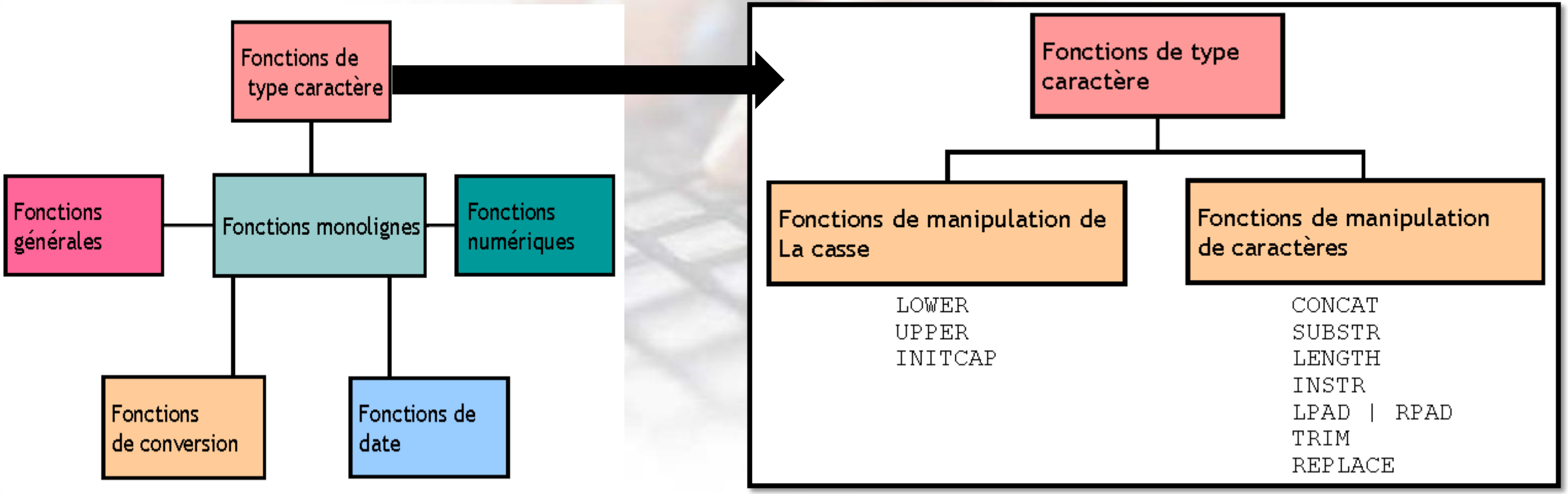
- Manipulent des données
- Acceptent des arguments et renvoient une seule valeur.
- Opèrent sur chaque ligne renvoyée.
- Renvoient un seul résultat par ligne.
- Peuvent modifier le type de données
- Peuvent être imbriquées
- Acceptent des arguments pouvant être une colonne ou une expression.

```
function_name [(arg1, arg2,...)]
```

V- Les fonctions (monoligne et de groupe)

5.1. LES FONCTIONS MONOLIGNES

On distingue plusieurs catégories de fonctions monolignes:



5.1. LES FONCTIONS MONOLIGNES

5.1.1. Les fonctions de type caractère

a) les fonctions de manipulation de la casse

- Ces fonctions convertissent la casse de chaînes de caractères :

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

Exemple:
Afficher le numéro, le nom et le numéro de département de l'employé Higgins.

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

TAF: Convertir le nom du client d'identifiant 5 de votre base de donnée, en majuscule.

V- Les fonctions (monoligne et de groupe)

5.1. LES FONCTIONS MONOLIGNES

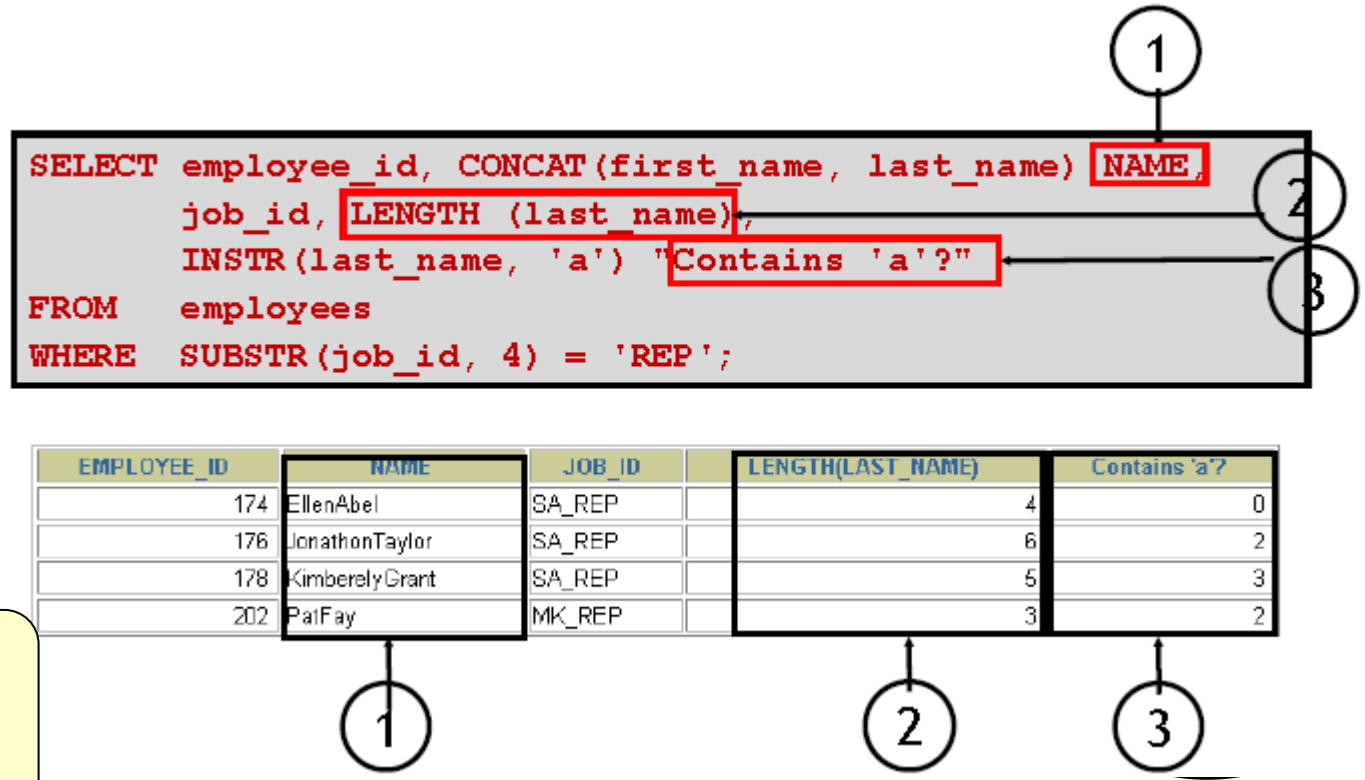
5.1.1. Les fonctions de type caractère

b) les fonctions de manipulation de caractères

- Ces fonctions manipulent des caractères:

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ('JACK and JUE','J','BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

TAF: donner le numero client, la concaténation du nom et prenom (rénommée en surnom) et la taille de ce Surnom pour l'étudiant d'identifiant 3.



5.1. LES FONCTIONS MONOLIGNES

5.1.2. Les fonctions numériques

- **ROUND** : arrondit la valeur à une décimale donnée.
- **TRUNC** : tronque la valeur à une décimale donnée.
- **MOD** : renvoie le reste de la division.

Function	Result
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100

TAF: Tester chacune de ces fonctions numérique.
Donner le modulo de 1000000 par 156 et renommer en « Modulo »

5.1. LES FONCTIONS MONOLIGNES

5.1.3. Les fonctions de date

a) fonction SYSDATE

- La fonction **SYSDATE()** dans MySQL est utilisée pour renvoyer la date et l'heure actuelles au format *AAAA-MM-JJ HH:MM:SS* ou *AAAAMMJJHHMMSS.uuuuuu* selon le contexte de la fonction.

TAF: Obtenir la date et l'heure actuelles à l'aide de la Fonction SYSDATE.

```
SELECT SYSDATE() as date_et_heure_actuelle ;
```

b) Calcul arithmétique sur des dates:

- Ajoutez un nombre à une date ou soustrayez un nombre d'une date afin d'obtenir une date résultante.
- Soustrayez une date d'une autre afin de déterminer le nombre de jours entre ces dates.

Function	Result
MONTHS_BETWEEN	Nombre de mois entre deux dates
ADD_MONTHS	Ajout d'un mois à une date
NEXT_DAY	Jour qui suit la date indiquée
LAST_DAY	Dernier jour du mois
ROUND	Date arrondie
TRUNC	Date tronquée

5.2. LES FONCTIONS DE GROUPE

- Les fonctions de groupe opèrent sur des ensembles de lignes afin de renvoyer un seul résultat par groupe.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400
...	

Salaire maximum de la table EMPLOYEES

MAX(SALARY)
24000

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

- **Syntaxe:**

```
SELECT      [column,] group_function(column), ...
FROM        table
[WHERE      condition]
[GROUP BY   column]
[ORDER BY   column];
```

5.2. LES FONCTIONS DE GROUPE

Les fonctions AVG et SUM

Les fonctions AVG et SUM sont généralement utilisées pour les données numériques

```
SELECT  AVG(salary), MAX(salary),
        MIN(salary), SUM(salary)
FROM    employees
WHERE   job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

Les fonctions MIN et MAX:

Les fonctions MIN et MAX pour les valeurs sont généralement utilisées pour les données numériques, les valeurs de type caractère et les valeurs de type date.

```
SELECT  MIN(hire date), MAX(hire date)
FROM        employees;
```

MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

5.2. LES FONCTIONS DE GROUPE

La fonction COUNT

- **COUNT(*)** renvoie le nombre de lignes d'une table:

```
SELECT  COUNT(*)
FROM    employees
WHERE   department_id = 50;
```

COUNT(*)
5

- **COUNT(expr)** renvoie le nombre de lignes avec des valeurs non NULL pour expr:r

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

COUNT(COMMISSION_PCT)
3

TAF: Appliquez les fonctions MIN, MAX, AVG, SUM et COUNT aux données de votre BD

1) Donnez le salaire moyen des employés
2) Donnez la masse salariale des employés
3) Quel est le plus petit prix de produit pratiqué?
4) Quel est le nombre de clients? d'employés?

5.2. LES FONCTIONS DE GROUPE

Créez des groupes de données avec la clause GROUP BY

- La clause GROUP BY sert à diviser les lignes d'une table en groupes plus petits.

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY department_id ;
```

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333

```
SELECT    department_id dept_id, job_id, SUM(salary)
FROM      employees
GROUP BY department_id, job_id ;
```

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200

TAF: Répondez à la question suivante en utilisant les données de votre BD
Donnez le nombre d'entreprises par ville d'appartenance

5.2. LES FONCTIONS DE GROUPE

Restreindre les résultats des groupes avec la clause HAVING

- permet donc de SÉLECTIONNER les colonnes d'une table t1 en GROUPANT les lignes qui ont des valeurs identiques sur une colonne donnée cx et que la condition de HAVING soit respectée.
- Les groupes qui correspondent à la clause HAVING s'affiche.

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

Exemple 1:on souhaite récupérer la liste des clients qui ont commandé plus de 40€, toute commandes confondu

id	client	tarif	date_achat
1	Pierre	102	2012-10-23
2	Simon	47	2012-10-27
3	Marie	18	2012-11-05
4	Marie	20	2012-11-14
5	Pierre	160	2012-12-03

```
SELECT client, SUM(tarif)
FROM achat
GROUP BY client
HAVING SUM(tarif) > 40
```

client	SUM(tarif)
Pierre	262
Simon	47

5.2. LES FONCTIONS DE GROUPE

- **Restreindre les résultats des groupes avec la clause HAVING**

Exemple 2:

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING    SUM(salary) > 13000
ORDER BY SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

Exemple 3: Imbriquer des fonctions de groupe

- Afficher le salaire moyen maximal

```
SELECT    MAX(AVG(salary))
FROM      employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

TAF: donner l'identifiant de l'entreprise et le salaire moyen des employés selon la catégorie en prenant en compte que les salaires moyens supérieur à 60 000F.

VI-
Jointures et produits cartésien

VI- Jointures et produits cartésien

6.1. Objectifs et types de jointure

- Ecrire des instructions SELECT afin d'accéder aux données de plusieurs tables à l'aide

d'équijointures et de non-équijointures.

- Joindre une table à elle-même à l'aide de l'auto-jointure.
- Générer un produit cartésien de toutes les lignes de plusieurs tables.

- Les jointures conformes à la norme SQL

sont les suivantes:

  - Jointures croisées
  - Jointures naturelles
  - Clause USING

EMPLOYEES

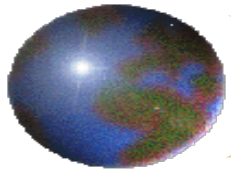
EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

```
SELECT   table1.column, table2.column
FROM     table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)]|
[CROSS JOIN table2];
```



VI- Jointures et produits cartésien

6.2. Jointure naturelle

- La clause **NATURAL JOIN** est basée sur toutes les colonnes des deux tables portant le même nom.
- Elle sélectionne les lignes des deux tables dont les valeurs sont identiques dans toutes les colonnes qui correspondent.
- Si les colonnes portant le même nom présentent des types de données différents, une erreur est renvoyée.

```
SELECT department_id, department_name,
       location_id, city
FROM   departments
NATURAL JOIN locations ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle
110	Accounting	1700	Seattle
190	Contracting	1700	Seattle
20	Marketing	1800	Toronto
80	Sales	2500	Oxford

TAF: Donner la liste comportant les noms et prénoms des employés, leur salaire respectif ainsi que le nom du département dans lequel chacun travaille.

6.2. Jointure naturelle

- Utilisez la clause **ON** pour indiquer des conditions arbitraires ou pour désigner les colonnes à joindre. Elle facilite la compréhension du code

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

TAF: Donner la liste des produits avec les données suivantes: le libellé du produit, le prix HT, la marque, le nom et l'identifiant du département dans lequel ils ont été produits.

6.3. Auto Jointure

- Utilisez la clause **ON** pour indiquer des conditions arbitraires ou pour désigner les colonnes à joindre. Elle facilite la compréhension du code

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping

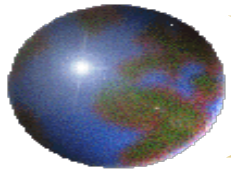
6.4. Produit cartésien

- Un produit cartésien est généré dans les cas suivants :
  - ✓ Une condition de jointure est omise
  - ✓ Une condition de jointure n'est pas valide
  - ✓ Toutes les lignes de la première table sont jointes à toutes les lignes de la seconde.
- Pour éviter un produit cartésien, incluez toujours une condition de jointure valide.
- La clause **CROSS JOIN** génère le produit cartésien de deux tables

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration

TAF: On souhaite avoir les différentes combinaisons possibles entre les tables employés et départements

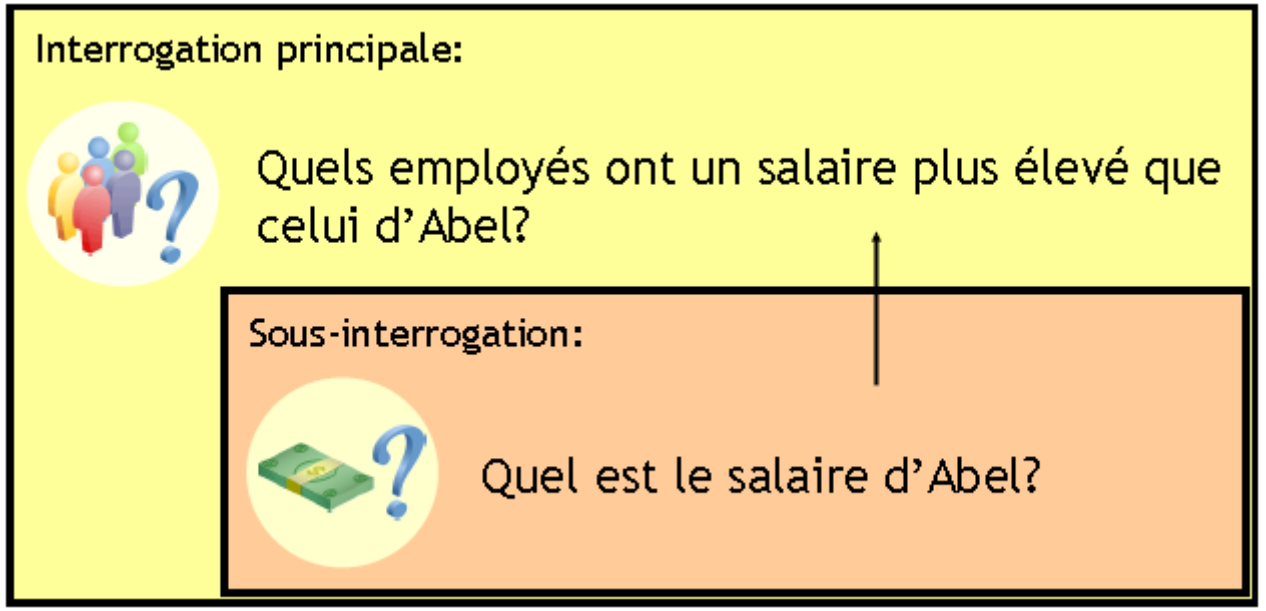


VII-
Sous-interrogations et opérateurs ensemblistes

VII- Sous-interrogations et opérateurs ensemblistes

7.1. Syntaxe

Qui a un salaire plus élevé que celui d' Abel?




```
SELECT    select_list
FROM      table
WHERE     expr operator
                    (SELECT      select_list
                    FROM         table);
```

- La sous-interrogation (interrogation interne) s'exécute une fois avant l'interrogation principale (interrogation externe).
- Le résultat de la sous-interrogation est utilisé par l'interrogation principale.

VII- Sous-interrogations et opérateurs ensemblistes

7.1. Syntaxe

Exemple:

```
SELECT last_name
FROM   employees
WHERE  salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Abel');
```

11000

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

Quelques règles:

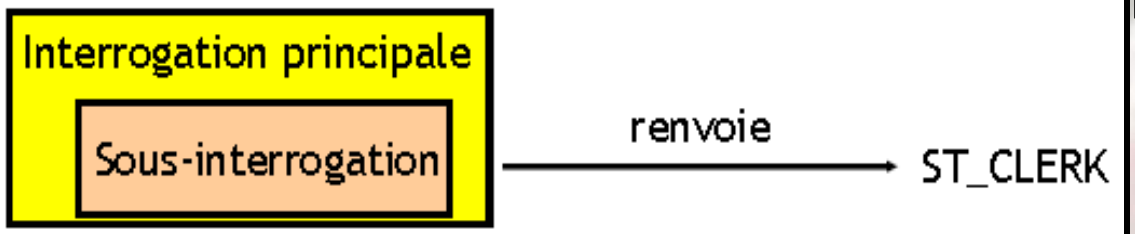
- Incluez les sous-interrogations entre parenthèses.
- Placez les sous-interrogations à droite de la condition de comparaison.
- La clause order by de la sous-interrogation n'est pas nécessaire.
- Utilisez des opérateurs monoligne avec les sous-interrogations monoligne, et des opérateurs multiligne avec les sous interrogations multiligne.

VII- Sous-interrogations et opérateurs ensemblistes

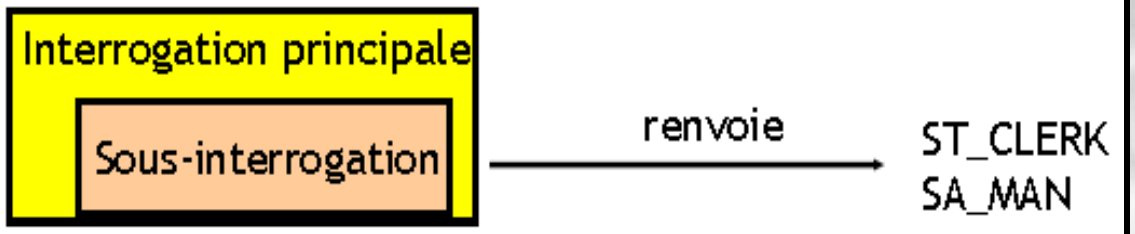
7.1. Syntaxe

Types de sous-interrogations:

- Sous-interrogation monoligne



- Sous-interrogation multiligne



- Renvoient une seule ligne
- Utilisent des opérateurs de comparaison monoligne

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Non égal à

- Renvoient plusieurs lignes
- Utilisent des opérateurs de comparaison multiligne

Opérateur	Signification
IN	Egal à un nombre quelconque de la liste
ANY	Compare la valeur à chaque valeur renvoyée par la sous-interrogation
ALL	Compare la valeur à toutes les valeurs renvoyées par la sous-interrogation

7.2. Sous-interrogations monoligne

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =                    ST_CLERK
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >                          2600
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

➢ Avec fonction de groupe:

```
SELECT last_name, job_id, salary
FROM   employees                   2500
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

7.2. Sous-interrogations monoligne

- Avec la clause HAVING

```
SELECT    department_id, MIN(salary)
FROM      employees
GROUP BY department_id
HAVING    MIN(salary) > 2500
                        (SELECT MIN(salary)
                         FROM   employees
                         WHERE  department_id = 50);
```

- erreur à éviter!!!

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT   MIN(salary)
                 FROM     employees
                 GROUP BY department_id);
```

A quel niveau se trouve l'erreur?

Opérateur monoligne avec sous-interrogation multiligne

7.3. Sous-interrogations multiligne

- Avec l'opérateur ANY

```
SELECT employee_id, last_name, job_id, salary
FROM   employees        9000, 6000, 4200
WHERE  salary < ANY
                (SELECT salary
                 FROM   employees
                 WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

- Avec l'opérateur ALL

```
SELECT employee_id, last_name, job_id, salary
FROM   employees        9000, 6000, 4200
WHERE  salary < ALL
                (SELECT salary
                 FROM   employees
                 WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

TAF: 1) Donner la liste des employés dont le salaire est supérieur à celui de l'employé ayant pour identifiant 003.
2) Donner la liste des clients qui sont plus âgés que le client ayant le contact '0020033000'.

7.4. Les opérateurs ensemblistes

- Utiliser UNION pour renvoyer toutes les lignes distinctes.
- Utiliser UNION ALL pour renvoyer toutes les lignes, y compris les doublons.
- Utiliser INTERSECT pour renvoyer toutes les lignes partagées par les deux interrogations.
- Utiliser MINUS pour renvoyer toutes les lignes distinctes sélectionnées par la première interrogation, mais pas par la deuxième.
- Utiliser ORDER BY uniquement à la toute fin de l'instruction.

Union

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```

Intersect

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

Minus

```
SELECT employee_id,job_id
FROM   employees
MINUS
SELECT employee_id,job_id
FROM   job_history;
```