Bases de données

Cours Semestre 2
2022-2023
Régine Laleau

Plan du cours

- 1ère partie – compléments SQL
  - Chapitre 1 : Fonctions de calcul et agrégats
  - Chapitre 2 : Rappels SQL - Requêtes complexes

- 2ème partie – fonctionnalités SQL
  - Chapitre 1 : Définition des données
  - Chapitre 2 : Maintien de la cohérence
  - Chapitre 3 : Gestion des utilisateurs
  - Chapitre 4 : Gestion de la confidentialité

Chapitre 1 : SQL - fonctions de calcul

Une fonction de calcul est une fonction qui s'applique sur un ensemble de tuples et qui renvoie une valeur unique

• Calcul sur les valeurs prises par un attribut

AVG (DISTINCT | ALL nom_attribut) = moyenne des valeurs prises par *nom_attribut*
SUM (DISTINCT | ALL nom_attribut) = somme des valeurs prises par *nom_attribut*

***nom_attribut* doit être de type entier ou réel**

Fournisseur (NomF, VilleF, AdresseF)

Livraison (NumLiv, NomP, NomF, DateLiv, Quantité)

Pièce (NomP, Prix, Couleur)

Ex : La quantité totale livrée des aspirateurs

NumLiv	NomP	NomF	DateLiv	Quantité
10	Aspirateur	Darty	10/10/18	10
5	MacBook	Auchan	25/03/19	29
27	Aspirateur	Carrefour	17/03/19	24

Chapitre 1 : SQL - fonctions de calcul

Une fonction de calcul est une fonction qui s'applique sur un ensemble de tuples et qui renvoie une valeur unique

• Calcul sur les valeurs prises par un attribut

AVG (DISTINCT | ALL nom_attribut) = moyenne des valeurs prises par *nom_attribut*
SUM (DISTINCT | ALL nom_attribut) = somme des valeurs prises par *nom_attribut*

***nom_attribut* doit être de type entier ou réel**

Fournisseur (NomF, VilleF, AdresseF)

Livraison (NumLiv, NomP, NomF, DateLiv, Quantité)

Pièce (NomP, Prix, Couleur)

Ex : La quantité totale livrée des aspirateurs

```
SELECT  SUM (ALL Quantité) FROM Livraison
WHERE NomP = 'Aspirateur ';
```

NumLiv	NomP	NomF	DateLiv	Quantité
10	Aspirateur	Darty	10/10/18	10
5	MacBook	Auchan	25/03/19	29
27	Aspirateur	Carrefour	17/03/19	24

Réponse : 34

Chapitre 1 : SQL - fonctions de calcul

MAX (nom_attribut) = maximum des valeurs prises par *nom_attribut*
MIN (nom_attribut) = minimum des valeurs prises par *nom_attribut*

***nom_attribut* doit être de type entier, réel, caractère ou date**

Ex : La livraison maximale d'aspirateurs.

- **Comptage du nombre de tuples**

COUNT(* | [ALL | DISTINCT nom_attribut]) =
nombre de valeurs de l'ensemble résultat
- dans le cas de * : y compris les valeurs nulles
- avec DISTINCT : sans les doublons
- avec ALL : avec les doublons

Chapitre 1 : SQL - fonctions de calcul

MAX (nom_attribut) = maximum des valeurs prises par *nom_attribut*
MIN (nom_attribut) = minimum des valeurs prises par *nom_attribut*

***nom_attribut* doit être de type entier, réel, caractère ou date**

Ex : La livraison maximale d'aspirateurs.

```
SELECT  MAX (Quantité)
FROM    Livraison
WHERE NomP = 'Aspirateur'
```

Réponse = 24

- **Comptage du nombre de tuples**

COUNT(* | [ALL | DISTINCT nom_attribut]) =
nombre de valeurs de l'ensemble résultat
- dans le cas de * : y compris les valeurs nulles
- avec DISTINCT : sans les doublons
- avec ALL : avec les doublons

Chapitre 1 : SQL - fonctions de calcul

Ex : Nombre de livraisons de la pièce de nom ‘ xxa1 ’

NumLiv	NomP	NomF	DateLiv	Quantité
10	xxa1	Darty	10/10/18	10
5	MacBook	Auchan	25/03/19	29
27	xxa1	Carrefour	17/03/19	24

Ex : Nombre de couleurs différentes dans les pièces

NomP	Prix	Couleur
xxa1	200	Rouge
MacBook	1200	Noir
Aspirateur	500	Gris
wscd	50	Rouge
Iphone	450	Vert

Chapitre 1 : SQL - fonctions de calcul

Ex : Nombre de livraisons de la pièce de nom ‘ xxa1 ’

SELECT COUNT(*) FROM Livraison WHERE NomP = ‘ xxa1 ’

NumLiv	NomP	NomF	DateLiv	Quantité
10	xxa1	Darty	10/10/18	10
5	MacBook	Auchan	25/03/19	29
27	xxa1	Carrefour	17/03/19	24

Réponse = 2

Ex : Nombre de couleurs différentes dans les pièces

SELECT COUNT(DISTINCT Couleur) FROM Pièce

NomP	Prix	Couleur
xxa1	200	Rouge
MacBook	1200	Noir
Aspirateur	500	Gris
wscd	50	Rouge
Iphone	450	Vert

Réponse = 4

Chapitre 1 : SQL - fonctions de calcul

- **Expression arithmétique**

  Exemple :

```
SELECT NumLiv, NomP, Quantité * Prix
FROM Livraison Natural Join Pièce
```

Les fonctions de calcul peuvent être utilisées dans la clause WHERE, elles portent alors sur un SELECT

EX : Donner les noms et prix des pièce ayant fait l 'objet d 'au moins 10 livraisons

```
SELECT P.NomP, P.Prix
FROM Piece P
WHERE (SELECT Count (*)
        FROM Livraison L
        WHERE P.NomP = L.NomP) >= 10
```

Chapitre 1 : SQL - partitionnement

GROUP BY < attribut$_1$, …, attribut$_n$ > permet de partitionner la relation en sous-relations ayant les mêmes valeurs sur les attributs précisés : on peut alors appliquer des fonctions (déclarées derrière SELECT) aux attributs de chaque sous-relation.

Ex: Livraison (NomP, NomF, Quantité)

GROUP BY NomP

	NomP	NomF	Quantité
sous-relation ⇨	a	x	5
	a	y	1
sous-relation ⇨	b	x	1
	b	t	5
	b	u	1
sous-relation ⇨	c	y	4

Les fonctions AVG, SUM, MIN, MAX, COUNT, placées dans la clause SELECT s'appliquent à chacune des sous-relations créées par le GROUP BY :

SELECT NomP, COUNT(*), SUM(Quantité)

FROM Livraison

GROUP BY NomP

NomP	Count(*)	Sum(Quantité)
a	2	6
b	3	7
c	1	4

Les attributs présents après SELECT sont forcément présents dans la clause GROUP BY.

Chapitre 1 : SQL - partitionnement (suite)

La clause HAVING permet de poser une condition portant sur chacune des sous-relations générées par le GROUP BY. Les sous-relations ne vérifiant pas la condition sont écartées du résultat.

Ex 1: Liste des fournisseurs qui ont effectué au moins 3 livraisons.

```
SELECT NomF
FROM Livraison
GROUP BY NomF
HAVING COUNT(*) >= 3
```

NumLiv	NomP	NomF	DateLiv	Quantité
10	xxa1	Darty	10/10/18	10
5	MacBook	Auchan	25/03/19	29
27	xxa1	Carrefour	17/03/19	24
18	Lait	Carrefour	24/08/2019	145
87	Fleur	Truffaut	08/08/2019	23
45	Pain	Auchan	14/03/2019	167
38	Lait	Auchan	09/11/2018	99
64	Pain	Carrefour	24/02/2019	109

Chapitre 1 : SQL - partitionnement (suite)

GROUP BY NomF

NumLiv	NomP	NomF	DateLiv	Quantité
45	Pain	Auchan	14/03/2019	167
5	MacBook	Auchan	25/03/19	29
38	Lait	Auchan	09/11/2018	99
27	xxa1	Carrefour	17/03/19	24
64	Pain	Carrefour	24/02/2019	109
18	Lait	Carrefour	24/08/2019	145
10	xxa1	Darty	10/10/18	10
87	Fleur	Truffaut	08/08/2019	23

SELECT NomF FROM Livraison GROUP BY NomF HAVING COUNT(*) >= 3

Chapitre 1 : SQL - partitionnement (suite)

GROUP BY NomF

NumLiv	NomP	NomF	DateLiv	Quantité
45	Pain	Auchan	14/03/2019	167
5	MacBook	Auchan	25/03/19	29
38	Lait	Auchan	09/11/2018	99
27	xxa1	Carrefour	17/03/19	24
64	Pain	Carrefour	24/02/2019	109
18	Lait	Carrefour	24/08/2019	145
10	xxa1	Darty	10/10/18	10
87	Fleur	Truffaut	08/08/2019	23

SELECT NomF FROM Livraison GROUP BY NomF HAVING COUNT(*) >= 3

Réponse

NomF
Auchan
Carrefour

Chapitre 1 : SQL - partitionnement (fin)

EMPLOYE (NumEmp, NomEmp, Salaire, Fonction, NumDepartement)

DEPARTEMENT (NumDepartement, NomDepartement, Localisation)

Ex 2 : Liste des numéros et noms de département et de leur nombre d'employés de fonction vendeur, dont la moyenne des salaires est supérieure à 24000 euros.

Chapitre 1 : SQL - partitionnement (fin)

EMPLOYE (NumEmp, NomEmp, Salaire, Fonction, NumDepartement)

DEPARTEMENT (NumDepartement, NomDepartement, Localisation)

Projection + Jointure

Ex 2 : Liste des numéros et noms de département et de leur nombre d'employés de fonction vendeur, dont la moyenne des salaires est supérieure à 24000 euros.

Restriction sur les tuples

Restriction sur les partitions

Chapitre 1 : SQL - partitionnement (fin)

EMPLOYE (NumEmp, NomEmp, Salaire, Fonction, NumDepartement)

DEPARTEMENT (NumDepartement, NomDepartement, Localisation)

Projection + Jointure

Ex 2 : Liste des numéros et noms de département et de leur nombre d'employés de fonction vendeur, dont la moyenne des salaires est supérieure à 24000 euros.

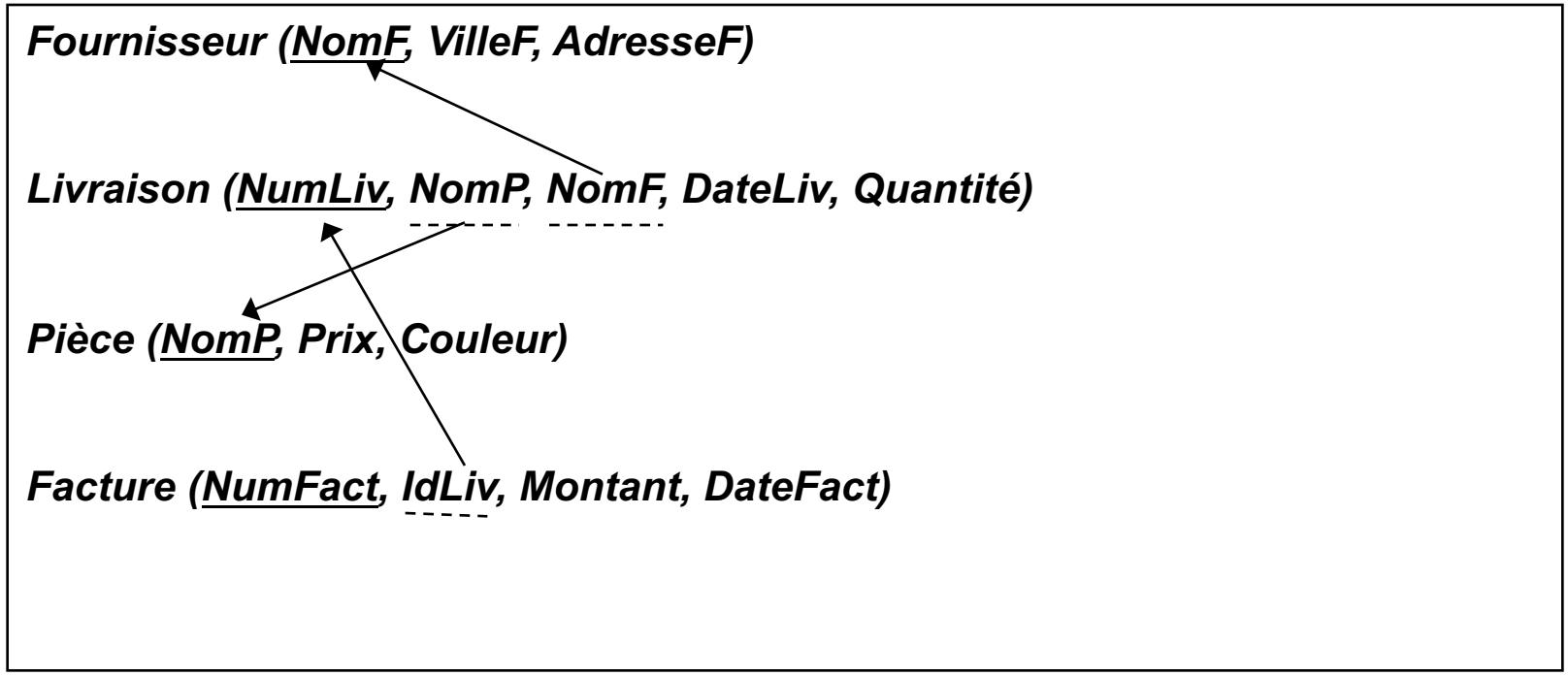
Restriction sur les tuples

Restriction sur les partitions

```
SELECT NumDepartement, NomDepartement, COUNT(*)
FROM Employe Natural Join Departement
WHERE Fonction = 'Vendeur'
GROUP BY NumDepartement, NomDepartement
HAVING AVG(Salaire) > 24000
```

Chapitre 2 : Rappels SQL - Requêtes complexes

***Jointure** : opération portant sur deux relations R1 et R2 consistant à construire une relation R3 ayant pour schéma la concaténation de celui de R1 et de R2 et pour tuples les tuples du produit cartésien de R1 par R2 vérifiant le prédicat de jointure.*



SQL - expression des jointures – SQL89

SQL-89 : on sélectionne des tuples du produit cartésien des relations citées dans la clause FROM

Noms et ville des fournisseurs qui ont des livraisons, avec le nom des pièces livrées

```
SELECT Fournisseur.NomF, VilleF, NomP
FROM Fournisseur, Livraison
WHERE Fournisseur.NomF = Livraison. NomF ;
```

Fournisseur

NomF	VilleF	AdresseF
Carrefour	Melun	Rue Dupond
Auchan	Paris	La Bastille
Casino	Melun	Rue de la Gare
SuperU	Nice	Rue du Marché

Livraison

NumLiv	NomP	NomF	DateLiv	Quantité
10	Lait	Carrefour	10/10/18	10
5	Pain	Auchan	25/03/19	29
27	Pizza	Carrefour	10/10/18	24
13	Fruit	Casino	15/07/19	15

NomF	VilleF	NomP
Carrefour	Melun	Lait
Auchan	Paris	Pain
Carrefour	Melun	Pizza
Casino	Melun	Fruit

SQL - expression des jointures – SQL89

SQL-89 : on sélectionne des tuples du produit cartésien des relations citées dans la clause FROM

Noms et ville des fournisseurs qui ont des livraisons, avec le nom des pièces livrées

```
SELECT Fournisseur.NomF, VilleF, NomP
FROM Fournisseur, Livraison
WHERE Fournisseur.NomF = Livraison. NomF ;
```

Fournisseur

NomF	VilleF	AdresseF
Carrefour	Melun	Rue Dupond
Auchan	Paris	La Bastille
Casino	Melun	Rue de la Gare
SuperU	Nice	Rue du Marché

Livraison

NumLiv	NomP	NomF	DateLiv	Quantité
10	Lait	Carrefour	10/10/18	10
5	Pain	Auchan	25/03/19	29
27	Pizza	Carrefour	10/10/18	24
13	Fruit	Casino	15/07/19	15

NomF	VilleF	NomP
Carrefour	Melun	Lait
Auchan	Paris	Pain
Carrefour	Melun	Pizza
Casino	Melun	Fruit

SQL - expression des jointures – SQL92

SQL-92 : spécification de la jointure dans la clause FROM

- **Jointure naturelle**

Prédicat : *égalité de tous les attributs communs (même nom)* aux deux relations

Schéma de la relation résultat : on ne garde qu'une seule fois les attributs communs

Noms et numéros de livraison des fournisseurs de Paris qui ont livré

```
SELECT DISTINCT NomF, NumLiv
FROM Fournisseur NATURAL JOIN Livraison
WHERE VilleF = ' Paris ';
```

NomF	NumLiv
Auchan	5

Ville des fournisseurs qui ont livré des pièces rouges

```
SELECT DISTINCT VilleF
FROM Fournisseur NATURAL JOIN Livraison  NATURAL JOIN Pièce
WHERE Couleur = ' rouge ';
```

SQL - expression des jointures – SQL92

• **Equi-jointure en spécifiant les attributs de jointure (qui ne sont mis qu'une seule fois dans le résultat)**

```
SELECT DISTINCT NomF, NumLiv
FROM Fournisseur JOIN Livraison USING (NomF)
WHERE VilleF = ' Paris ';
```

• **Equi-jointure en spécifiant le critère de jointure**

Date et numéro des factures des livraisons du fournisseur 'Carrefour'

```
SELECT NumFact, DateFact,
FROM Facture F JOIN Livraison L ON F.IdLiv = L.NumLiv
WHERE NomF = 'Carrefour' ;
```

```
SELECT NumFact, DateFact,
FROM Facture NATURAL JOIN Livraison
WHERE NomF = 'Carrefour' ;
```

donne un résultat faux !!!
(pas d'erreur d'exécution)

SQL - expression des jointures externes – SQL92 (1)

SQL-92 : on garde dans le résultat les tuples qui ne joignent pas (d'une relation, de l 'autre, ou des deux)

livre

Titre	N°auteur
La peste	56
L'assommoir	10

auteur

N°auteur	Nom
10	Zola
15	Tolkien

SELECT * FROM livre LEFT OUTER JOIN auteur ON livre.n° auteur = auteur.n° auteur ;
==>

Titre	Livre.N°auteur	Auteur.N°auteur	Nom
La peste	56	**Null**	**Null**
L'assommoir	10	10	Zola

SELECT * FROM livre RIGHT OUTER JOIN auteur ON livre.n° auteur = auteur.n° auteur ;
==>

Titre	Livre.N°auteur	Auteur.N°auteur	Nom
Null	**Null**	15	Tolkien
L'assommoir	10	10	Zola

SQL - expression des jointures externes – SQL92 (1)

SQL-92 : on garde dans le résultat les tuples qui ne joignent pas (d'une relation, de l 'autre, ou des deux)

livre

Titre	N°auteur
La peste	56
L'assommoir	10

auteur

N°auteur	Nom
10	Zola
15	Tolkien

SELECT * FROM livre LEFT OUTER JOIN auteur ON livre.n° auteur = auteur.n° auteur ;
==>

Titre	Livre.N°auteur	Auteur.N°auteur	Nom
La peste	56	**Null**	**Null**
L'assommoir	10	10	Zola

SELECT * FROM livre RIGHT OUTER JOIN auteur ON livre.n° auteur = auteur.n° auteur ;
==>

Titre	Livre.N°auteur	Auteur.N°auteur	Nom
Null	**Null**	15	Tolkien
L'assommoir	10	10	Zola

SQL - expression des jointures externes – SQL92 (2)

livre

Titre	N°auteur
La peste	56
L'assommoir	10

auteur

N°auteur	Nom
10	Zola
15	Tolkien

SELECT * FROM livre FULL OUTER JOIN auteur ON livre.n° auteur = auteur.n° auteur ;
==>

Titre	Livre.N°auteur	Auteur.N°auteur	Nom
Null	**Null**	15	Tolkien
L'assommoir	10	10	Zola
La peste	56	**Null**	**Null**

SQL - expression des jointures externes – SQL92 (2)

livre

Titre	N°auteur
La peste	56
L'assommoir	10

auteur

N°auteur	Nom
10	Zola
15	Tolkien

SELECT * FROM livre FULL OUTER JOIN auteur ON livre.n° auteur = auteur.n° auteur ;
==>

Titre	Livre.N°auteur	Auteur.N°auteur	Nom
Null	**Null**	15	Tolkien
L'assommoir	10	10	Zola
La peste	56	**Null**	**Null**

SQL - Produit cartésien

Produit cartésien A × B : sélectionne tous les couples possibles de A et B

Fournisseur

NomF	VilleF	AdresseF
Carrefour	Melun	Rue Dupond
Auchan	Paris	La Bastille

Livraison

NumLiv	NomP	NomF	DateLiv	Quantité
10	Lait	Carrefour	10/10/18	10
5	Pain	Auchan	25/03/19	29
27	Pizza	Carrefour	10/10/18	24

SELECT * FROM Fournisseur, Livraison;

NomF	VilleF	AdresseF	NumLiv	NomP	NomF	DateLiv	Quantité
Carrefour	Melun	Rue Dupond	10	Lait	Carrefour	10/10/18	10
Carrefour	Melun	Rue Dupond	5	Pain	Auchan	25/03/19	29
Carrefour	Melun	Rue Dupond	27	Pizza	Carrefour	10/10/18	24
Auchan	Paris	La Bastille	10	Lait	Carrefour	10/10/18	10
Auchan	Paris	La Bastille	5	Pain	Auchan	25/03/19	29
Auchan	Paris	La Bastille	27	Pizza	Carrefour	10/10/18	24

SQL - fonctions ensemblistes dans la clause WHERE

Le résultat d 'une requête SELECT est un *ensemble* de tuples. On peut utiliser ce résultat, avec des fonctions ensemblistes, dans un prédicat de la clause WHERE

nom_attribut [NOT] IN (< sous-requête >)
évalué à VRAI si nom_attribut appartient à l 'ensemble des résultats de <sous-requête>

Ex: Lister les noms des pièces qui n'ont pas été livrées

```
SELECT NomP
FROM  Pièce
WHERE NomP NOT IN (SELECT NomP FROM Livraison);
```

Remarque : on peut aussi utiliser des jointures externes (SQL 92)

```
SELECT NomP
FROM  Pièce LEFT OUTER JOIN Livraison on (Piece.NomP = Livraison.NomP)
WHERE NumF IS NULL ;
```

SQL - fonctions ensemblistes dans la clause WHERE

Le résultat d 'une requête SELECT est un *ensemble* de tuples. On peut utiliser ce résultat, avec des fonctions ensemblistes, dans un prédicat de la clause WHERE

nom_attribut [NOT] IN (< sous-requête >)
évalué à VRAI si nom_attribut appartient à l 'ensemble des résultats de <sous-requête>

Ex: Lister les noms des pièces qui n'ont pas été livrées

```
SELECT NomP
FROM  Pièce
WHERE NomP NOT IN (SELECT NomP FROM Livraison);
```

Remarque : on peut aussi utiliser des jointures externes (SQL 92)

```
SELECT NomP
FROM  Pièce LEFT OUTER JOIN Livraison on (Piece.NomP = Livraison.NomP)
WHERE NumF IS NULL ;
```

SQL - expression des opérateurs ensemblistes

- **< requête > UNION < requête >**
- **< requête > INTERSECT < requête >**
- **< requête > EXCEPT < requête >**

les 2 requêtes doivent avoir le même nombre d'attributs avec des types comparables dans la clause SELECT

Ex: Nom des fournisseurs qui livrent la pièce SSZ4 mais pas la pièce A2

```
(SELECT NomF FROM Livraison WHERE  NomP = 'SSZ4')
 EXCEPT
(SELECT NomF FROM Livraison WHERE NomP = 'A2');
```

SQL - fonctions quantifiées dans le WHERE

[NOT] EXISTS (< sous-requête >)
évalué à VRAI si <sous-requête> retourne au moins un élément

Ex: Lister les pièces qui n'ont pas été livrées

```
SELECT NomP
FROM  Pièce P
WHERE NOT EXISTS (SELECT *
                  FROM Livraison
                  WHERE Livraison.NomP = P.NomP);
```

Liste des fournisseurs qui ont livré des pièces rouges et des pièces noires

```
SELECT NomF, AdresseF
FROM  Fournisseur F
WHERE EXISTS (SELECT *
              FROM Livraison Natural Join Piece
              WHERE Livraison.NomF = F.NomF
              AND Couleur = "Rouge")
AND EXISTS (SELECT *
            FROM Livraison Natural Join Piece
            WHERE Livraison.NomF = F.NomF
            AND Couleur = "Noir")
);
```

SQL - fonctions quantifiées dans le WHERE

[NOT] EXISTS (< sous-requête >)
évalué à VRAI si <sous-requête> retourne au moins un élément

Ex: Lister les pièces qui n'ont pas été livrées

```
SELECT NomP
FROM  Pièce P
WHERE NOT EXISTS (SELECT *
                  FROM Livraison
                  WHERE Livraison.NomP = P.NomP);
```

Liste des fournisseurs qui ont livré des pièces rouges et des pièces noires

```
SELECT NomF, AdresseF
FROM  Fournisseur F
WHERE EXISTS (SELECT *
              FROM Livraison Natural Join Piece
              WHERE Livraison.NomF = F.NomF
              AND Couleur = "Rouge")
AND EXISTS (SELECT *
            FROM Livraison Natural Join Piece
            WHERE Livraison.NomF = F.NomF
            AND Couleur = "Noir")
);
```

SQL – Exemple de requêtes complexes

Liste des fournisseurs qui ont livré des pièces rouges et des pièces noires

- **Avec fonction ensembliste IN**

```
SELECT NomF, AdresseF
 FROM  Fournisseur
 WHERE nomF IN (SELECT nomF
                FROM Livraison Natural Join Piece WHERE Couleur = "Rouge")
AND nomF IN (SELECT nomF
             FROM Livraison Natural Join Piece WHERE Couleur = "Noir"));
```

- **Avec des variables**

```
SELECT NomF, AdresseF
FROM  Fournisseur F, Livraison L1, Livraison L2, Piece P1, Piece P2
WHERE F.NomF = L1.NomF AND L1.NomP = P1.NomP AND P1.Couleur = "Rouge"
AND F.NomF = L2.NomF AND L2.NomP = P2.NomP AND P2.Couleur = "Noir");
```

- **Avec opérateur ensembliste**

```
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Rouge")
INTERSECT
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Noir")
```

SQL – Exemple de requêtes complexes

Liste des fournisseurs qui ont livré des pièces rouges et des pièces noires

- **Avec fonction ensembliste IN**

```
SELECT NomF, AdresseF
 FROM  Fournisseur
 WHERE nomF IN (SELECT nomF
                FROM Livraison Natural Join Piece WHERE Couleur = "Rouge")
AND nomF IN (SELECT nomF
             FROM Livraison Natural Join Piece WHERE Couleur = "Noir"));
```

- **Avec des variables**

```
SELECT NomF, AdresseF
FROM  Fournisseur F, Livraison L1, Livraison L2, Piece P1, Piece P2
WHERE F.NomF = L1.NomF AND L1.NomP = P1.NomP AND P1.Couleur = "Rouge"
AND F.NomF = L2.NomF AND L2.NomP = P2.NomP AND P2.Couleur = "Noir");
```

- **Avec opérateur ensembliste**

```
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Rouge")
INTERSECT
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Noir")
```

SQL – Exemple de requêtes complexes

Liste des fournisseurs qui ont livré des pièces rouges et des pièces noires

- **Avec fonction ensembliste IN**

```
SELECT NomF, AdresseF
 FROM  Fournisseur
 WHERE nomF IN (SELECT nomF
                FROM Livraison Natural Join Piece WHERE Couleur = "Rouge")
AND nomF IN (SELECT nomF
             FROM Livraison Natural Join Piece WHERE Couleur = "Noir"));
```

- **Avec des variables**

```
SELECT NomF, AdresseF
FROM  Fournisseur F, Livraison L1, Livraison L2, Piece P1, Piece P2
WHERE F.NomF = L1.NomF AND L1.NomP = P1.NomP AND P1.Couleur = "Rouge"
AND F.NomF = L2.NomF AND L2.NomP = P2.NomP AND P2.Couleur = "Noir");
```

- **Avec opérateur ensembliste**

```
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Rouge")
INTERSECT
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Noir")
```

SQL – Exemple de requêtes complexes

Liste des fournisseurs qui ont livré des pièces rouges et des pièces noires

- **Avec fonction ensembliste IN**

```
SELECT NomF, AdresseF
 FROM  Fournisseur
 WHERE nomF IN (SELECT nomF
                FROM Livraison Natural Join Piece WHERE Couleur = "Rouge")
AND nomF IN (SELECT nomF
             FROM Livraison Natural Join Piece WHERE Couleur = "Noir"));
```

- **Avec des variables**

```
SELECT NomF, AdresseF
FROM  Fournisseur F, Livraison L1, Livraison L2, Piece P1, Piece P2
WHERE F.NomF = L1.NomF AND L1.NomP = P1.NomP AND P1.Couleur = "Rouge"
AND F.NomF = L2.NomF AND L2.NomP = P2.NomP AND P2.Couleur = "Noir");
```

- **Avec opérateur ensembliste**

```
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Rouge")
INTERSECT
(SELECT NomF, AdresseF
 FROM  Fournisseur NATURAL JOIN Livraison Natural Join Piece
WHERE Couleur = "Noir")
```

SQL - tri du résultat

La clause ORDER BY permet de trier le résultat de la requête, en fournissant la liste des attributs (ou de n° s de colonnes) sur lesquels effectuer le tri et en spécifiant le sens du tri (ascendant ou descendant).

EMPLOYE (NumEmp, NomEmp, Salaire, Fonction, NumDepartement)

Ex 1 : Liste des employés triée par ordre alphabétique sur le nom

```
SELECT *
FROM Employe
ORDER BY NomEmp ASC
```

Ex 2 : Liste des employés triée sur le salaire par ordre croissant et sur le nom par ordre décroissant

```
SELECT *
FROM Employe
ORDER BY Salaire ASC, NomEmp DESC
```

SQL - Résumé

Forme générale d 'une requête SQL

5 SELECT attributs résultats [avec fonctions]

1 FROM relations utilisées

2 [WHERE condition de sélection indépendante du GroupBy]

3 [GROUP BY attributs de partitionnement

4 [HAVING condition de sélection portant sur les agrégats]]

6 [ORDER BY critère de tri du résultat]