SQL — Ch 5 – SELECT Sous-requêtes

I. Introduction

Une **SOUS-REQUETE,** ou requête imbriquée,(sous requête interne, *en anglais : inner subquery*) est une requête utilisée pour **CONSTITUER UN JEU DE RESULTAT** qui va **SERVIR DANS UNE REQUETE PRINCIPALE**, la requête appelante (requête externe, *en anglais : outer query*),

- - en général comme **élément de comparaison** dans la clause **WHERE**,
- - parfois comme **valeur d'une colonne** dans la clause **SELECT**.

Synonyme : requêtes imbriquées

A. Syntaxe générale

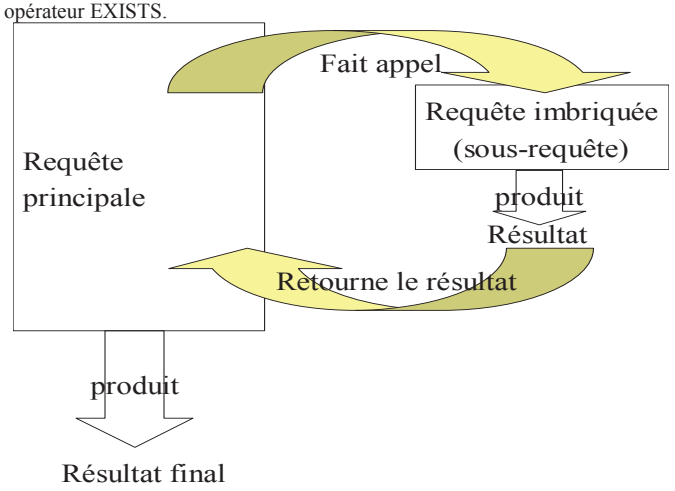
```
SELECT . . .[, (sous requête) AS alias_colonne]
FROM . . .
WHERE   nom_colonne  operateur  (sous requête)
[ GROUP BY . . . ]
[ HAVING nom_colonne  operateur  (sous requête) ]
[ ORDER BY . . . ]
;
```

Où : **(sous requête)** est une requête SQL

B. Résultat renvoyé par une sous-requête

La sous requête doit produire un résultat (ensemble de lignes et de colonnes) **conforme à ce qu'attend la requête appelante**. Ce résultat est :

- •**UNE VALEUR UNIQUE** (1 ligne et 1 colonne) **:**
  - il pourra être utilisé dans la requête appelante dans la clause WHERE ou HAVING :
    - opérateurs de comparaison =, <, >, <=,>=, <>,
    - opérateur BETWEEN,
  - il pourra être utilisé dans la requête appelante pour constituer la valeur d'une colonne dans la clause SELECT
- •**LIGNES MULTIPLES, COLONNE UNIQUE** (0 à N lignes – une seule colonne) **:**
  - il pourra être utilisé dans la requête appelante dans la clause WHERE ou HAVING :
    - opérateurs IN, NOT IN
    - opérateurs ALL, ANY combinés avec un opérateur de comparaison =, <, >, <=,>=, <>,
- •**Un JEU DE DONNEES QUELCONQUE** (0 à N LIGNES – 1 à N COLONNES) **:**
  - opérateur EXISTS.



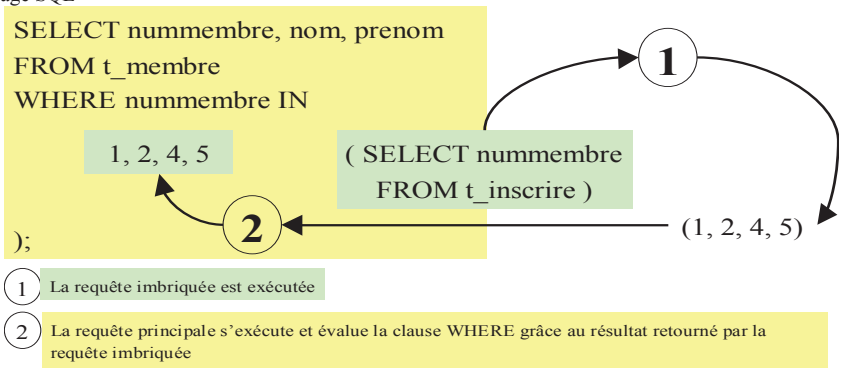
C. sous requête indépendantes et sous requête dépendantes

Deux formes de requêtes imbriquées sont définies, en fonction de leur lien avec la requête appelante :

- Les requêtes imbriquées **indépendantes**, sans aucun lien avec la requête appelante
- Les requêtes imbriquées **dépendantes**, ou **liées** ou **corrélées**, qui ont un lien avec la requête appelante

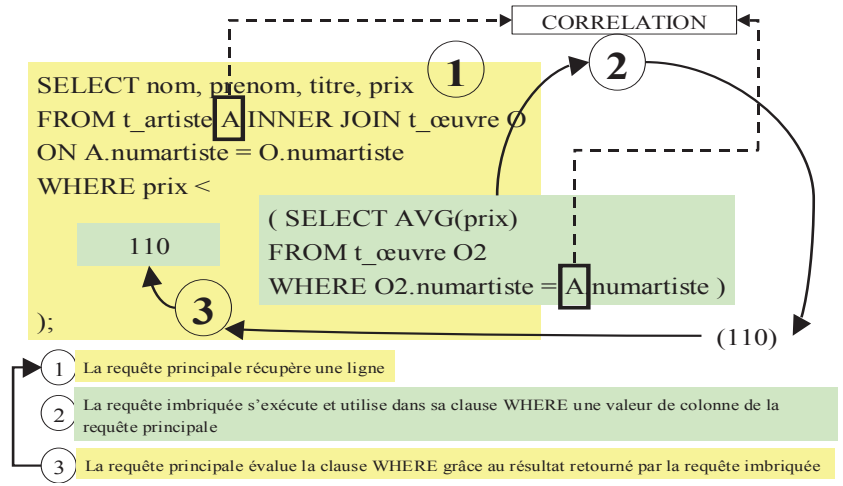
1. INDEPENDANTES

Une requête indépendante peut être lancée seule, elle est totalement indépendante de la requête appelante. L'évaluation de la requête imbriquée est d'abord effectué une fois pour toutes et va servir (comme valeur unique, ou liste de valeurs) à la requête appelante.



2. DEPENDANTES, ou CORRELEES, ou LIEES

La requête imbriquée corrélée va dépendre, cette fois, de valeurs de colonnes de la requête appelante, elle est liée à la requête appelante, elle en est dépendante, corrélée (en relation avec).
La requête imbriquée est alors effectué/évaluée à chaque ligne de la requête appelante.



II. Sous-requêtes à VALEUR UNIQUE (résultat)

EN CAS DE RETOUR D'UN ENSEMBLE DE RESULTAT MULTIPLE (PLUSIEURS LIGNES OU PLUSIEURS COLONNES, UNE ERREUR SE PRODUIT.

A. Exemples de requêtes produisant une valeur unique

►Lister la date de naissance du membre de numéro 12
SELECT dateNais_memb FROM t_membre WHERE id_memb = 12 ;

dateNais_memb
14/12/1985

►Lister le nombre de membres
SELECT COUNT(*) AS nbMembres FROM t_membre ;

nbMembres
225

►Lister le total des dernières cotisations versées par les membres
SELECT SUM(dernCotis) AS totalCotis FROM t_membre ;

totalCotis
2812,50

B. Constituer la valeur d'une colonne

1. **Exemple :**

« Le titre des peintures, leur largeur et la largeur moyenne de toutes les peintures) »
(1)_La largeur moyenne des peintures :

```
SELECT AVG(largeur) FROM t_peinture
;
```

Requête indépendante

(2)_Les titres des œuvres, leur largeurs et la largeur moyenne de toutes les peintures (résultat trouvé en (1) :

```
SELECT titre, largeur,
    (
    SELECT AVG(largeur) FROM t_peinture
    )
    AS largeur_moyenne
FROM t_oeuvre A INNER JOIN t_peinture B ON A.numoeuvre =
B.numoeuvre ;
```

C. Opérateurs =,<,<=,>,>= ,<> ou !=

1. **Exemple :**

« Le titre des œuvre d'un artiste dont le nom est 'picasso' »
(1)_Retrouver le numéro de l'artiste dont le nom est 'picasso' (il faut être sûr qu'

```
SELECT numartiste FROM t_artiste
WHERE LOWER(nom) = 'picasso'
;
```

Requête indépendante

(2)_Retrouver la liste des œuvres dont le numéro d'artiste est celui trouvé en (1) :

```
SELECT titre
FROM t_oeuvre
WHERE numartiste =
    (
    SELECT numartiste FROM t_artiste
    WHERE LOWER(nom) = 'picasso'
    )
```

```
;
```

2. **Exemple :**

« Le titre des peintures (leur largeur et la largeur moyenne de toutes les peintures) dont la largeur est inférieure à la moyenne »

(1)_La largeur moyenne des peintures :

```
SELECT AVG(largeur) FROM t_peinture
;
```

Requête indépendante

(2)_Les oeuvres dont la largeur est inférieure au résultat trouvé en (1) ;(on affiche également la largeur et sa moyenne – autre utilisation de la requête imbriquée à valeur unique - afin de vérifier la bonne exécution de la requête) :

```
SELECT titre, largeur,
    (
    SELECT AVG(largeur) FROM t_peinture
    )
    AS largeur_moyenne
FROM t_oeuvre A INNER JOIN t_peinture B ON A.numoeuvre =
B.numoeuvre
WHERE largeur <
    (
    SELECT AVG(largeur) FROM t_peinture
    )
;
```

3. **Exemple :**

« Le titre des peintures dont la largeur est supérieure à la largeur moyenne de toutes les peintures»

```
SELECT titre
FROM t_oeuvre A INNER JOIN t_peinture B ON A.numoeuvre =
B.numoeuvre
WHERE largeur > (SELECT AVG(largeur) FROM t_peinture) ;
```

D. Opérateur BETWEEN AND

1. **Exemple :**

« Le titre des peintures dont la largeur est égale à la moyenne de toutes les peintures, plus ou moins 50% »

```
SELECT titre
FROM t_oeuvre A INNER JOIN t_peinture B ON A.numoeuvre =
B.numoeuvre
```



```
WHERE largeur BETWEEN (
     SELECT AVG(largeur) * 0.50
     FROM t_peinture
     )
     AND (
     SELECT AVG(largeur) * 1.5
     FROM t_peinture
     )
;
```

2. **Exemple :**

« Les nom et prénom des membres qui se sont inscrits à une activité avant de s'inscrire à l'activité 'Paris expo.' »

(1)_Retrouver le numéro de l'activité 'Paris Expo.'

```
SELECT numactivite FROM t_activite
WHERE LOWER(intitule) = 'paris expo.'
;
```

Requête indépendante

(2)_Retrouver la date d'inscription à cette activité POUR CHAQUE MEMBRE (corrélation)

```
SELECT dateInscrire FROM t_inscrire
WHERE numactivite =
     (
     SELECT numactivite FROM t_activite
     WHERE LOWER(intitule) = 'paris expo.'
     )
AND nummembre = MEMBRE DE LA TABLE APPELANTE
;
```

(3)_Retrouver la liste des membres qui ont une date d'inscription antérieure à la date d'inscription trouvée en (2)

```
SELECT DISTINCT nom, prenom
FROM t_membre A INNER JOIN t_inscrire B
ON A.nummembre = B.nummembre
WHERE date_inscrire <
     (
     SELECT dateInscrire FROM t_inscrire
     WHERE numactivite =
          (
          SELECT numactivite FROM t_activite
          WHERE LOWER(intitule) = 'paris expo.'
          )
     AND nummembre = A.nummembre
     )
;
```

CORRELATION

Requête corrélée

3. **Exemple :**

« Le titre des peintures (leur largeur et la largeur moyenne des peintures pour chaque artiste) dont la largeur est supérieure à la largeur moyenne des peintures du même artiste»

```
SELECT titre, largeur,
     (SELECT AVG(largeur) FROM   t_oeuvre A2 INNER JOIN
     t_peinture B2 ON A2.numoeuvre = B2.numoeuvre WHERE
     A2.numartiste = A.numartiste )
     AS moyenneParArtiste
FROM t_oeuvre A INNER JOIN t_peinture B ON A.numoeuvre =
B.numoeuvre
WHERE largeur > (
     SELECT AVG(largeur)
     FROM t_peinture
     WHERE numoeuvre IN
          (
          SELECT numoeuvre FROM t_œuvre
          WHERE numartiste = A.numartiste
          )
     )
;
```

CORRELATION

CORRELATION

III.Sous-requêtes à LIGNES MULTIPLES et COLONNE UNIQUE

Les **valeurs de la COLONNE UNIQUE** retournées par une requête imbriquée vont servir d'élément de comparaison avec un opérateur traitant d'un ensemble de valeurs : IN et NOT IN, ALL, ANY.

A. Exemples de requêtes

►Lister numéro des membres qui habitent Arras
SELECT id_memb FROM t_membre WHERE ville_memb = "Arras";

id_memb
13
231
57
121

►Lister dates de naissance des membres qui habitent Arras
SELECT dateNais_memb FROM t_membre WHERE ville_memb = "Arras";

dateNais_memb
14/12/1985
06/01/1095
15/03/1976
03/06/1977

B. Opérateurs IN et NOT IN

L'opérateur IN permet de vérifier qu'une valeur de colonne de la requête appelante existe dans la liste des valeurs retournées par la requête imbriquée.
Avec l'opérateur IN, **AU MOINS UNE DES VALEURS renvoyée par la sous requête doit être égale à la valeur de la colonne externe .**

1. **Exemple :**

« Le nom et le prénom des membres qui se sont inscrits au moins une fois à une activité»

(1)_les numéros des membres qui se sont inscrits :

```
SELECT nummembre FROM t_inscrire ;
```

(1')_un membre peut s'inscrire plusieurs fois, on peut optimiser le résultat (en n'ayant qu'une seule fois un numéro de membre – suppression des doublons grâce à la clause DISTINCT) :

```
SELECT DISTINCT nummembre FROM t_inscrire ;
```

(2)_liste des membres dont le numéro est parmi les numéros de (1)

```
SELECT nom, prenom
FROM membre
WHERE nummembre IN
     (SELECT DISTINCT nummembre FROM t_inscrire
     )
;
```

On aurait pu également traiter la requête grâce à une jointure :

```
SELECT DISTINCT nom, prenom
FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
B.nummembre
;
```

2. **Exemple :**

« Le nom et le prénom des membres qui NE se sont JAMAIS inscrits à une activité»

```
SELECT nom, prenom
FROM t_membre
WHERE nummembre NOT IN
     (SELECT DISTINCT nummembre FROM t_inscrire
     )
;
```

ou bien

```
. . .WHERE NOT (
      nummembre IN
          (SELECT DISTINCT nummembre FROM t_inscrire)
     );
```

3. **Exemple :**

« Les nom et prénom des membres inscrits à une activité consistant en une visite d'exposition à Paris»

(1)_les numéros des expositions ayant eu lieu à Paris

```
SELECT numexpo FROM t_exposition
WHERE LOWER(ville) = 'paris' ;
```

(2)_les numéros des activités qui sont parmi la liste des numéros de (1)

```
SELECT numactivite FROM t_visiter
WHERE numexpo IN
     (SELECT numexpo FROM t_exposition
     WHERE LOWER(ville) = 'paris'
     )
;
```

(3)_les numéros des membres qui se sont inscrits à une activité dont le numéro est l'un des numéros de (2)

```
SELECT nummembre FROM t_inscrire
WHERE numactivite IN
     (SELECT numactivite FROM t_visiter
     WHERE numexpo IN
          (SELECT numexpo FROM t_exposition
          WHERE LOWER(ville) = 'paris'
          )
     )
;
```

(4)_les membres dont le numéro est l'un des numéros de (3)

```
SELECT nom, prenom FROM t_membre
WHERE nummembre IN
```



```
     (SELECT nummembre FROM t_inscrire
     WHERE numactivite IN
          (SELECT numactivite FROM t_visiter
          WHERE numexpo IN
               (SELECT numexpo FROM t_exposition
               WHERE LOWER(ville) = 'paris'
               )
          )
     )
;
```

Une jointure nous permet également d'obtenir ce résultat :

```
SELECT DISTINCT nom, prenom
FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
B.nummembre
INNER JOIN t_visiter C ON B.numactivite = C.numactivite
INNER JOIN t_exposition D ON C.numexpo = D.numexpo
WHERE LOWER(D.ville) = 'paris'
;
```

C. Opérateur ALL

L'opérateur ALL est associé à un opérateur de comparaison et permet de contrôler une valeur de colonne de la requête appelante par rapport à toutes les valeurs retournées par la requête imbriquée.
Avec l'opérateur ALL, **TOUTES les valeurs de la sous requêtes doivent répondre au critère de comparaison**.

1. **Exemple :**

« Les membres qui sont plus jeunes que tous les membres qui se sont inscrits» (= dont la date de naissance est supérieure à toutes les dates de naissance des membres qui se sont inscrits à une activité)

```
SELECT * FROM t_membre
WHERE datenaissance  > ALL
     (SELECT datenaissance
     FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
     B.nummembre
     )
;
```

> ALL (SELECT colonne . . .) *est équivalent à* > (SELECT MAX(colonne). . .)

```
SELECT * FROM t_membre
WHERE datenaissance >
     (SELECT MAX(datenaissance)
     FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
     B.nummembre
```

```
    )
;
```

2. **Exemple :**

« Les membres qui sont plus âgés que tous les membres qui se sont inscrits» (= dont la date de naissance est inférieure à toutes les dates de naissance des membres qui se sont inscrits à une activité)

```
SELECT * FROM t_membre
WHERE datenaissance  < ALL
    (SELECT datenaissance
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

< ALL (SELECT colonne . . .) est équivalent à < (SELECT MIN(colonne). . .)

```
SELECT * FROM t_membre
WHERE datenaissance <
    (SELECT MIN(datenaissance)
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

3. **Exemple :**

« Les membres qui ne sont pas nés à la même date que les membres qui se sont inscrits» (= dont la date de naissance est différente de toutes les dates de naissance des membres qui se sont inscrits à une activité)

```
SELECT * FROM t_membre
WHERE datenaissance <> ALL
    (SELECT datenaissance)
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

<> ALL (SELECT colonne . . .) est équivalent à NOT IN (SELECT colonne . . .)

```
SELECT * FROM t_membre
WHERE datenaissance NOT IN
    (SELECT datenaissance)
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```


D. Opérateur ANY (ou SOME)

L'opérateur **ANY** est associé à un opérateur de comparaison et permet de contrôler une valeur de colonne de la requête appelante par rapport à toutes les valeurs retournées par la requête imbriquée.
Avec l'opérateur ANY, **AU MOINS UNE des valeurs de la sous requête doit répondre au critère de comparaison** pour que cette comparaison soit vraie.

```
SELECT * FROM t_membre
WHERE datenaissance  < ANY
    (SELECT datenaissance
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

< ANY (SELECT colonne . . .) *est équivalent à* < (SELECT MAX(colonne). . .)

```
SELECT * FROM t_membre
WHERE datenaissance  <
    (SELECT MAX(datenaissance)
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

```
SELECT * FROM t_membre
WHERE datenaissance  > ANY
    (SELECT datenaissance
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

> ANY (SELECT colonne . . .) *est équivalent à* > (SELECT MIN(colonne). . .)

```
SELECT * FROM t_membre
WHERE datenaissance  >
    (SELECT MIN(datenaissance)
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

```
SELECT * FROM t_membre
WHERE datenaissance  = ANY
    (SELECT datenaissance
```

```
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```

= ANY (SELECT colonne . . .) est équivalent à IN (SELECT colonne . . .)

```
SELECT * FROM t_membre
WHERE datenaissance  IN
    (SELECT DISINCT datenaissanc)
    FROM t_membre A INNER JOIN t_inscrire B ON A.nummembre =
    B.nummembre
    )
;
```


	ALL	ANY ou SOME
>	« supérieur à toutes les valeurs de la sous-requête » équivalent : **>** (SELECT **MAX**(colonne). . .) (supérieur à la plus grande des valeurs de la liste)	« supérieur à au moins une des valeurs de la sous-requête » » équivalent : **>** (SELECT **MIN**(colonne). . .) (supérieur à la plus petite des valeurs de la liste)
<	« inférieur à toutes les valeurs de la sous-requête » équivalent : **<** (SELECT **MIN**(colonne). . .) (inférieur à la plus petite des valeurs de la liste)	« inférieur à au moins une des valeurs de la sous-requête » équivalent : **<** (SELECT **MAX**(colonne). . .) (inférieur à la plus grande des valeurs de la liste)
<>	« différent de toutes les valeurs de la sous-requête » équivalent : **NOT IN** (SELECT colonne. . .) (n'est pas parmi les valeurs de la liste)	« différent d'au moins une des valeurs de la sous-requête »
=	« égal à toutes les valeurs de la sous-requête »	« égal à au moins une des valeurs de la sous-requête » équivalent : **IN** (SELECT colonne. . .) (est parmi les valeurs de la liste)

IV. Sous-requêtes à JEU DE DONNEES QUELCONQUE

Les requêtes exploitent le retour de ce type de sous-requêtes pour vérifier que le résultat est vide ou non.

A. Exemples de requêtes

►Lister les membres qui habitent 'Arras'
SELECT * FROM t_membre WHERE ville_memb = "Arras";;

id_memb	Nom_memb	Prenom_memb	. . .
13	Dupond	gérard	
231	Durant	julie	
57	Lambert	martine	
121	Assain	Marc	

►Lister les membres qui habitent 'Moscou
SELECT * FROM t_membre WHERE ville_memb = "Moscou";;

id_memb	Nom_memb	Prenom_memb	. . .

(vide)

B. Opérateur EXISTS

L'opérateur **EXISTS** permet de contrôler qu'une sous requête renvoie un résultat non vide. Le résultat de l'évaluation est

- •VRAI si le résultat de la sous requête comporte au moins 1 ligne, (résultat non vide),
- •FAUX sinon. (résultat vide, aucune ligne renvoyée par la sous requête).

1. Exemple :

« la liste des artistes s'il existe des membres dont l'anniversaire est fêté ce mois-ci»

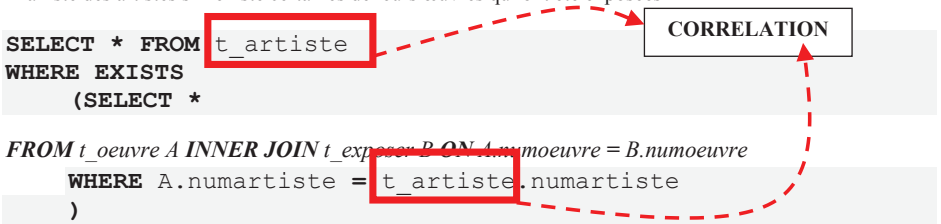
```
SELECT * FROM t_artiste
WHERE EXISTS
     (SELECT *
```

(1) ***FROM*** *t_membre*

```
     WHERE EXTRACT(MONTH FROM dateNaissance) = EXTRACT(MONTH
     FROM CURRENT_DATE)
     )
;
```


2. Exemple :

« la liste des artistes s'il existe certaines de leurs œuvres qui ont été exposées»




```
SELECT * FROM t_artiste
WHERE EXISTS
     (SELECT *
```

FROM *t_oeuvre A* ***INNER JOIN*** *t_exposer B* ***ON*** *A.numoeuvre = B.numoeuvre*

```
     WHERE A.numartiste = t_artiste.numartiste
     )
;
```

ATTENTION : si la sous requête comporte une fonction d'agrégation, l'application de l'opérateur EXISTS sera toujours vérifiée : en effet, un COUNT(*) retourne forcément une valeur, qui peut être 0 .

V. Apports supplémentaires de la norme SQL2

Les plus récentes normes SQL modifient sensiblement le format des jeux de valeurs utilisés avec certains opérateurs dans l'évaluation des requêtes imbriquées.
Il devient possible de comparer non plus une seule valeur à un ensemble de valeurs uniques, mais un jeu de valeurs à d'autres jeux de valeurs.

1. Exemple

« les membres (nom et prénom) qui sont aussi des artistes (nom, prenom)»

```
SELECT *
  FROM t_membre
  WHERE (nom, prenom) IN
     (
     SELECT nom, prenom FROM t_artiste
     )
;
```

2. Exemple

Utiliser une sous-requête à la place d'une table (dans la clause WHERE)

```
SELECT *
  FROM
  (
    SELECT . . .
    FROM . . .
    WHERE . . .
  )
..WHERE . . .
;
```