20_REQUÊTES IMBRIQUÉES
REQUÊTES SUR LE RÉSULTAT D'UNE REQUÊTE

http://patatos.over-blog.com/article-mysql-requetes-imbriquees-73265573.html
https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql/1964181-sous-requetes
https://sqlpro.developpez.com/cours/sqlaz/sousrequetes/

SOMMAIRE :

- Introduction
- Requêtes imbriquée retournant une table ou un champ
- Requêtes imbriquée retournant une colonne
- Requêtes imbriquées testant l'existence d'une valeur
- Requêtes imbriquées retournant une valeurs
- Les jointures entre requêtes de sélection : requêtes corrélées
  - • L'union : UNION
  - • L'intersection : INTERSECT
  - • L'exclusion : EXCEPT/MINUS

INTRODUCTION :

Dans certains cas, il peut être nécessaire de réaliser une requête non pas sur une table, mais sur le résultat d'une autre requête : c'est là que les requêtes imbriquées ou sous requêtes entre en jeu.

- L'imbrication n'a pas vraiment de limite technique.
- Lors de requêtes imbriqués, le premier SELECT est appelé requête principale, le deuxième sous requêtes 1, et ainsi de suite.
- L'exécution s'effectue toujours de la requête la plus « profonde » dans l'imbrication vers la requête la principale.

REQUÊTES IMBRIQUÉE RETOURNANT UNE TABLE OU UN CHAMP :

Ce type de requête peut être utilisé dans un FROM ou un SELECT, comme source de données de la requête principale.
C'est utile dans le FROM pour limiter notamment le nombre d'enregistrements sur lequel le reste de la requête doit s'appliquer.
Pour le SELECT, cela permet d'afficher dans une requête un champs provenant d'une autre requête, sans avoir à réaliser de jointure particulière.

Exemple :
On peut calculer le nombre de film de la catégorie « Action » avec la requête suivante (même si on peut le faire avec un group by) :

```
select category.name, (
    select count(film_category.film_id)
        from film_category
        join category
        on film_category.category_id = category.category_id
        where category.name =  'Action') as Nb_films
    from category
    where category.name =  'Action';
```

name	Nb_films
Action	64

REQUÊTE IMBRIQUÉ RETOURNANT UNE COLONNE

Ce type de requête peut être utilisé avec le prédicat IN (ou NOT IN) pour l'évaluation d'un champ par rapport à une liste de valeurs retournées par cette requête.

Exemple :
On peut afficher la liste des films qui ont été loués, c'est à dire se dont on retrouve une trace dans la table rental.

```
select film.title from film
    where film_id in (
        select film_id from inventory
            inner join rental on inventory.inventory_id = rental.inventory_id);
```

title
ACADEMY DINOSAUR
ACE GOLDFINGER
ADAPTATION HOLES
AFFAIR PREJUDICE
AFRICAN EGG
AGENT TRUMAN

REQUÊTES IMBRIQUÉES TESTANT L'EXISTENCE D'UNE VALEUR :

L'utilisation d'une requête imbriquée avec le prédicat IN ne doit être confondu avec l'utilisation du prédicat EXISTS (ou NOT EXISTS).

- Celui-ci ne vérifie pas la concordance avec une ou plusieurs valeurs, mais bien si une valeur existe ou non.
- Pour un jeu d'enregistrement équivalents, ce prédicat est donc plus performant que IN car il ne va pas traiter toute les possibilités, mais s'arrêter dès que l'existence est vérifier.

Exemple :

On peut connaitre les films ayant uniquement pour acteur référencé « MCCONAUGHEY CARY ».

```
select f.film_id, title from film f
    join film_actor on f.film_id = film_actor.film_id
    join actor on film_actor.actor_id = actor.actor_id
        and concat(last_name, ' ', first_name) = 'MCCONAUGHEY CARY'
    where not exists (
    select film.film_id, title from film
        join film_actor on film.film_id = film_actor.film_id
        join actor on film_actor.actor_id = actor.actor_id
            and concat(last_name, ' ', first_name) <> 'MCCONAUGHEY CARY'
    where f.film_id = film.film_id);
```

Le 1er select renvoie la liste des films dans les quel à joué « MCCONAUGHEY CARY »

Le 2ème select renvoie la liste des films dans les quel à joué n'importe quel acteurs sauf « MCCONAUGHEY CARY »

La clause where permet de spécifier sur quel champs des deux requêtes le test doit être fait notamment en spécifiant un aliace dans le 1er select.

film_id	title
240	DOLLS RAGE

Ce type de requête peut être utilisé comme champ d'un SELECT de la requête principale, ou comme valeur d'une condition dans un WHERE ou un HAVING.

Exemple :

Pour afficher les films appartenant à la même catégorie que le film « DOLLS RAGE »

```
select title from film
    join film_category on film.film_id = film_category.film_id
    where category_id = (
        select category_id from film_category
        join film on film_category.film_id = film.film_id
        where title = 'DOLLS RAGE');
```

title
ANNIE IDENTITY
ARMAGEDDON LOST
ATTACKS HATE
BADMAN DAWN
BARBARELLA STREETCAR
BEVERLY OUTLAW

LES JOINTURE ENTRE REQUÊTES DE SÉLECTION : REQUÊTE CORRÉLÉES

Il est possible d’effectuer des opérations ensemblistes entre les résultats de deux requêtes de sélection ayant la même structure, c’est-à-dire ayant le même nombre de champs de type identique.
Ces opération sont :

- UNION
- INTERSECT
- EXCEPT (ou MINUS suivant les SGBD)

Travail à faire : à vous de creuser, faite une veille sur ces trois opérations.

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
Par exemple, pour connaˆıtre les films en allemand :
SELECT ∗ FROM sakila.f i l m
WHERE l a n g u a g e i d IN
(SELECT l a n g u a g e i d FROM s a k i l a . l an gu a ge WHERE name=’German ’) ;

&&&&&&&&&&&&&&&&&&&&
Pour connaˆıtre les acteurs qui ont jou´e dans au moins un film, on pourra ´ecrire :
SELECT ∗ FROM s a k i l a . a c t o r AS ac
WHERE EXISTS (SELECT ∗ FROM s a k i l a . f i l m a c t o r a s f a
WHERE f a . a c t o r i d = ac . a c t o r i d) ;