

## BD01 et WGBD

*SQL comme langage d'interrogation :  
Requêtes imbriquées corrélées et opérations ensemblistes.*

EPFC-ULB

Boris Verhaegen

boris@verhaegen.me

## Requêtes imbriquées avec EXISTS ( )

Employe

<u>SSN</u>	Nom
123	Jacques
456	Boris
789	Alain

Patron

<u>SSN</u>	<u>PSSN</u>
456	123
789	123

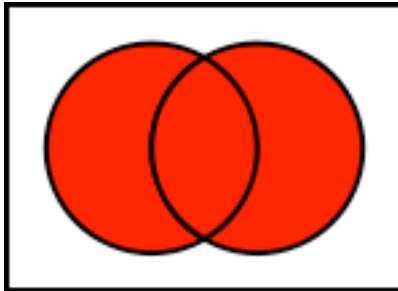
EXISTS ( ) est une fonction booléenne qui renvoie vrai si la sous-requête donnée comme argument est non-vide.

Noms des employés qui sont des patrons

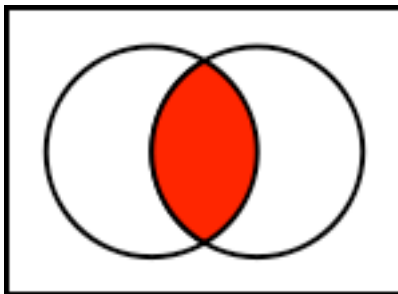
```
SELECT e.Nom
FROM Employe e
WHERE EXISTS ( SELECT *
                FROM Patron p
                WHERE e.SSN = p.PSSN )
```

La sous requête peut employer des variables de la super-requête. On parle alors requêtes **corrélées**.

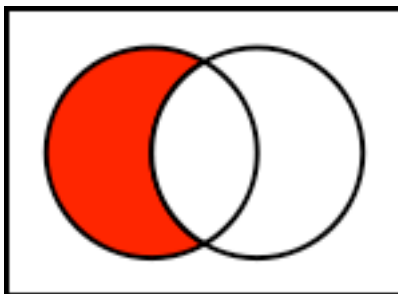
# Opérations ensemblistes



**Union** de deux ensembles :  
Mots clés UNION et UNION ALL



**Intersection** de deux ensembles :  
Mot clé INTERSECT (n'existe pas dans MySQL)  
Requêtes avec IN ou EXISTS

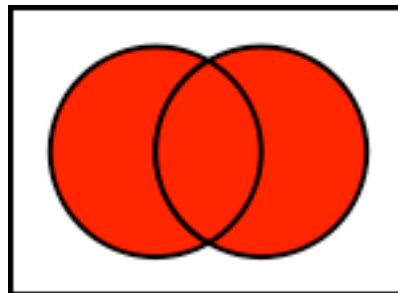


**Différence** de deux ensembles :  
Mot clé MINUS (n'existe pas dans MySQL)  
Requêtes avec NOT IN

# Unions

SQL permet de combiner des requêtes à l'aide des **opérateurs ensemblistes** UNION et UNION ALL

Ces opérateurs fonctionnent sur des **ensembles compatibles**, c'est à dire des relations ayant les mêmes colonnes.



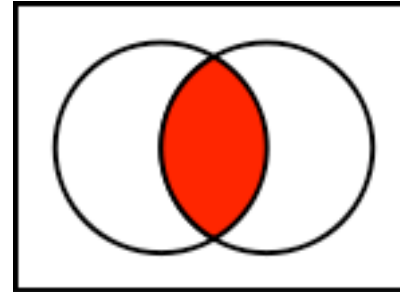
# UNION et UNION ALL

UNION fait l'union de deux ensembles et supprime les doublons.

```
SELECT product_id FROM order_items  
UNION  
SELECT product_id FROM inventories
```

UNION ALL garde les doublons.

```
SELECT location_id FROM locations  
UNION ALL  
SELECT location_id FROM departments
```



# Intersections

INTERSECT renvoie l'intersection de deux ensembles mais n'est pas (encore) implémenté dans MySQL.

```
SELECT product_id FROM inventories
INTERSECT
SELECT product_id FROM order_items
```

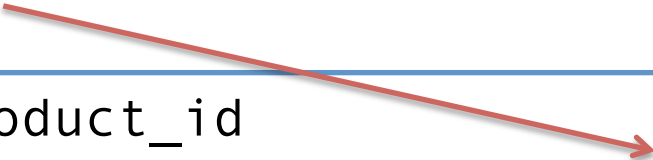
Il est possible de s'en passer à l'aide de requêtes imbriquées

```
SELECT product_id
FROM inventories
WHERE product_id IN ( SELECT product_id
                      FROM order_items )
```

## Se passer d'INTERSECT à l'aide d'EXISTS ( )

```
SELECT product_id FROM inventories
INTERSECT
SELECT product_id FROM order_items
```

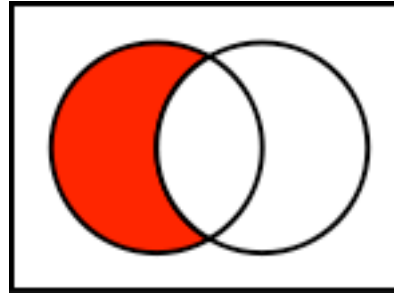
Les produits de la table inventaire pour lesquels il existe un **même produit** dans la table des commandes.



```
SELECT product_id
FROM inventories i
WHERE EXISTS( SELECT *
               FROM order_items o
               WHERE o.product_id = i.product_id )
```

#!/\ corrélation /\

## Différences



MINUS renvoie la différence entre deux ensembles (mais n'est pas encore implémenté dans MySQL)

```
SELECT product_id FROM inventories
MINUS
SELECT product_id FROM order_items
```

Il est possible de s'en passer à l'aide de requêtes imbriquées

```
SELECT product_id
FROM inventories
WHERE product_id NOT IN ( SELECT product_id
                           FROM order_items )
```




## Se passer de MINUS à l'aide d'EXISTS ( )

```
SELECT product_id FROM inventories
MINUS
SELECT product_id FROM order_items
```

Les produits de la table inventaire pour lesquels il n'existe **pas** le **même produit** dans la table des produits commandés.

```
SELECT product_id
FROM inventories i
WHERE NOT EXISTS( SELECT *
                  FROM order_items o
                  WHERE o.product_id = i.product_id )
```

 /!\ corrélation /!\