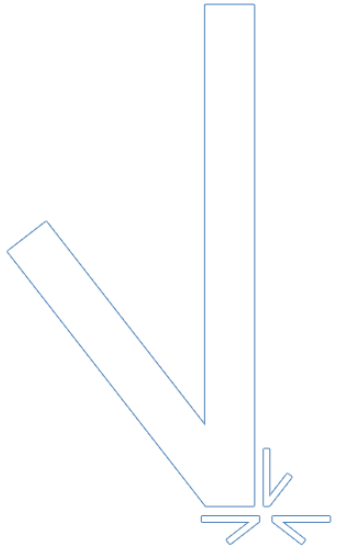
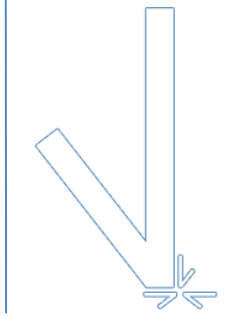
# SQL
# MySQL Avancé

Objectif:

**Découvrir ou re-découvrir les fonctionnalités avancées de MySQL et les requêtes complexes**

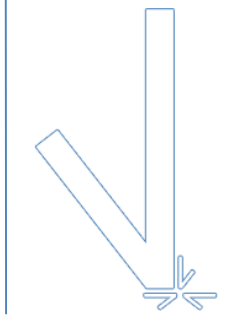


# Introduction



- Opérations CRUD de base : insert, select, update, delete
- Jointures
- Triggers
- Procédures stockées
- Vues

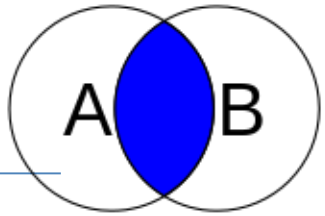
# Jointures



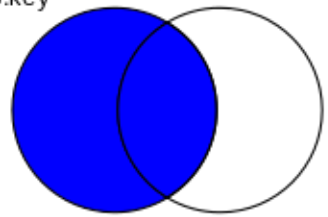
- But : tirer le meilleur parti des bases de données relationnelles
- Comment : dans une même requête, requêter les données issues de plusieurs tables
- Différents types de jointures :
  - Jointures internes : INNER JOIN
  - Jointures externes : LEFT JOIN , RIGHT JOIN, FULL JOIN

# SQL JOINS

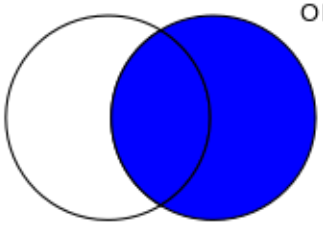
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



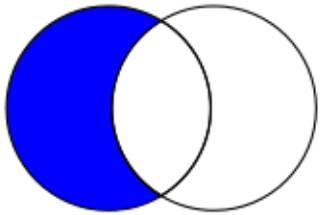

SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key



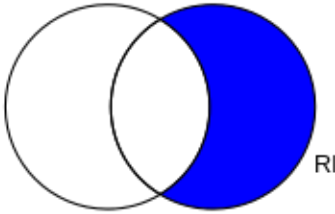
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



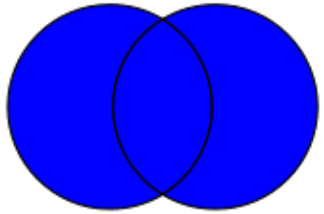
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL



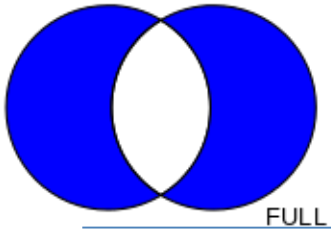
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL



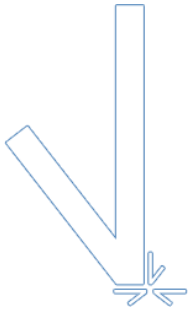
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL



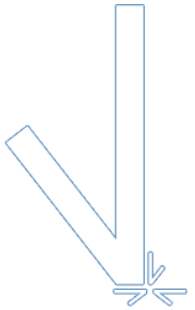
# Triggers (ou déclencheurs)



- **CREATE TRIGGER trigger_name action_time trigger_event ON table_name**
- But : déclencher automatiquement des actions

- action_time : BEFORE ou AFTER
- trigger_event : INSERT, UPDATE ou DELETE
- Donc 6 possibilités !

# Triggers (ou déclencheurs)

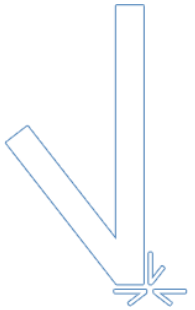


- Les mots clés **NEW** et **OLD** permettent d'acccéder aux nouvelles valeurs et aux anciennes valeurs des enregistrements

- La déclaration **FOR EACH ROW** permet d'appliquer le trigger pour tous les enregistrements concerné

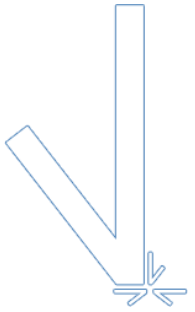
https://dev.mysql.com/doc/refman/8.0/en/trigger-syntax.html

# Triggers (ou déclencheurs)



- **Avantages :**
  - Rapidité
  - Facilité d'implémentation
- **Inconvénients :**
  - Difficilement traçable, donc debug difficile
  - Exécution invisible depuis le code metier
  - …

# Routines (ou procédures stockées)



- **CREATE PROCEDURE routine_name (parameters)**

**BEGIN**

**instructions**

**END**

- But : déclencher des actions à la demande
- Utilisation : CALL routine_name()

- Suppression : DROP PROCEDURE routine_name;

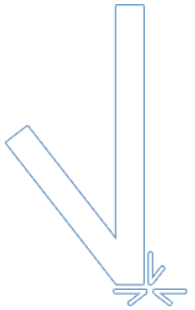
# Exemple Routine

```
CREATE PROCEDURE `afficher_villes_du_departement` (id
    INT)
BEGIN
    SELECT v.nom, d.nom
    FROM villes v
    INNER JOIN departements d
        ON v.departements_id = d.id
    WHERE d.id = id ;
END
```

Pour l'utiliser : CALL afficher_villes_du_departement(1)

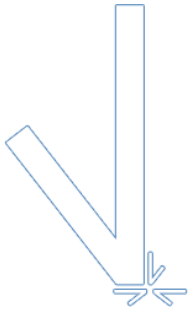
Pour la supprimer DROP PROCEDURE afficher_villes_du_departement

# Routines (ou procédures stockées)



- Les paramètres peuvent être de 3 modes :
  - **IN** : paramètre en entrée (comportement par défaut)
  - **OUT** : paramètre en sortie
  - **INOUT** : paramètre en entrée/sortie

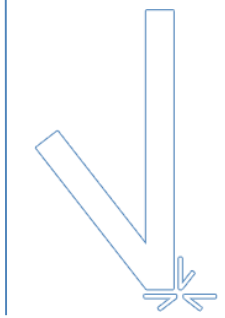
# Routine



- **Avantages :**
  - Les mêmes que les triggers !
  - Réutilisabilité
  - Plus un déclenchement décidé/contrôlé
  - Persistences
- **Inconvénients :**
  - Logique toujours séparée du code metier
  - Surcharge du serveur BDD

- https://dev.mysql.com/doc/refman/8.0/en/stored-routines-syntax.html

# Views (ou vues)



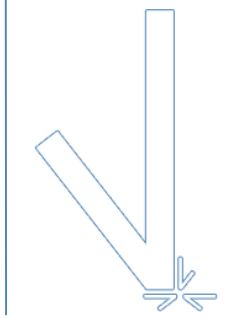
**CREATE [OR REPLACE] VIEW view_name**
**AS select_request;**

- But : créer un objet  de la base de données pour faciliter l'affichage de certaines informations
- On peut requêter  une vue comme on requête une table
- Créer de manière durable
- Ce qui est stocké est la requête, pas les résultats => donc pas de gain de performance

# Exemple view

```
CREATE VIEW `v_villes_departements` AS
SELECT v.nom AS Ville, d.nom AS Département
FROM villes v
INNER JOIN departements d ON
   v.departements_id = d.id
```

# Views (ou vues)



- Avantages :
  - Permettre d'avoir une requête complexe en un objet persistant sans avoir à modifier le schema de la base
  - Réutilisable dans des jointures pour les rendre plus lisibles
- Inconvénients :
  - Des changements fréquents de schéma rendent les vues difficilement maintenables.

- https://dev.mysql.com/doc/refman/8.0/en/views.html