

# Exercices

## Les données

### Utilisation du GUI phpmyadmin

- Une startup souhaite lancer un réseau social minimaliste où les utilisateurs peuvent avoir un profil détaillé et où certains utilisateurs peuvent être certifiés
  - Création de compte
  - Utilisateur et profil détaillé dans des tables séparées
  - Un utilisateur possède un champs supplémentaire: date de certification

# Exercices

## Les données

### Utilisation du SQL dans phpmyadmin

- Une entreprise souhaite mettre en place un système de support client permettant aux utilisateurs de créer des tickets d'assistance pour signaler des problèmes techniques. Les administrateurs peuvent ajouter des commentaires aux tickets, mais ils peuvent également commenter d'autres types d'éléments comme les articles de la FAQ
  - Ouvrir un ticket d'assistance (A suivre jusqu'à la résolution)
  - Les administrateurs peuvent ajouter des commentaires aux tickets ouverts et aux articles de la FAQ
  - Statut des tickets: ouvert, en cours, résolu

# Exercices

## Les données

## Utilisation du SQL/php

Une entreprise possède plusieurs salles de réunion et souhaite mettre en place un système de réservation pour que les employés puissent réserver des salles pour des réunions, formations ou événements internes.

1. Les réservations des salles se font par les employés. (Date, heure de début, heure de fin)
2. Un système hiérarchique des employés, où chaque employé peut avoir un supérieur direct.
3. Seul un supérieur peut annuler une réservation

# Vocabulaire

## Base de données

- **MCD (Modèle Conceptuel de Données):** Représentation graphique et abstraite des données et de leurs relations

**Exemple: Entités et associations:**

### 1. User (Utilisateur)

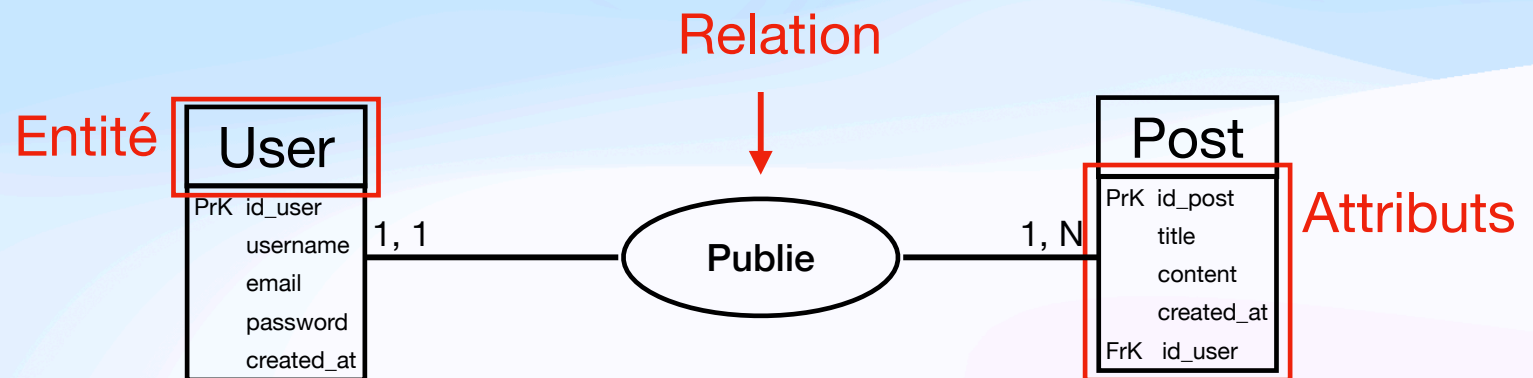
- id\_user (Identifiant, clé primaire)
- username (Nom d'utilisateur)
- email (Adresse e-mail)
- password (Mot de passe)
- created\_at (Date d'inscription)

### 2. Post (Article)

- id\_post (identifiant, clé primaire)
- title (Titre de l'article)
- content (Contenu de l'article)
- created\_at (Date de publication)
- id\_user (identifiant de l'auteur, clé étrangère vers users)

### 3. Relation « Publie » (Un utilisateur peut publier plusieurs posts)

- **Cardinalité: 1,N** (1 utilisateur peut écrire plusieurs articles, mais un article appartient à un seul utilisateur)



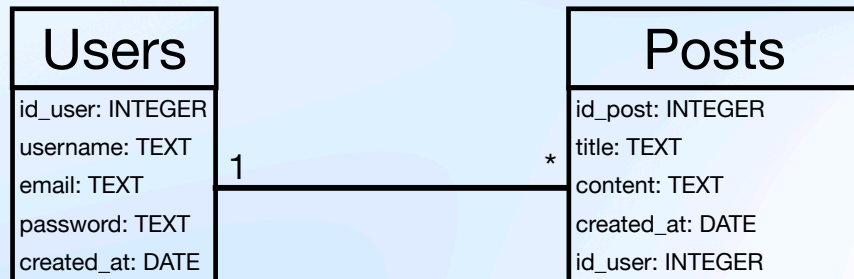
- On ne précise pas le type
- Règle de gestion

# Vocabulaire

## Base de données

- **MLD (Modèle Logique de Données):** Représentation structurée des données sous forme de tables avec leurs relation, destinée à être implémenté dans un SGBD

Exemple: Structure des tables relationnelles



- Tables et colonnes
- Contraintes

- **MPD (Modèle Physique de Données)** traduction du MLD en une implémentation concrète dans un SGBD, incluant les types de données spécifiques, les index, les contraintes et les options de stockage

### Exemple

#### 1. Table users

```
CREATE TABLE users (  
  id_user INT PRIMARY KEY AUTO_INCREMENT,  
  username VARCHAR(50) NOT NULL UNIQUE,  
  email VARCHAR(100) NOT NULL UNIQUE,  
  password VARCHAR(255) NOT NULL,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

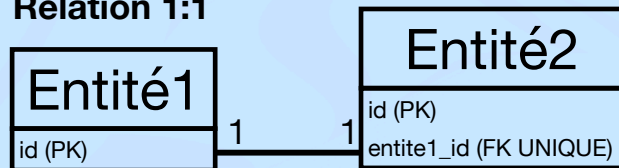
#### 2. Table posts

```
CREATE TABLE posts (  
  id_post INT PRIMARY KEY AUTO_INCREMENT,  
  title VARCHAR(255) NOT NULL,  
  content TEXT NOT NULL,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  id_user INT NOT NULL,  
  FOREIGN KEY (id_user) REFERENCES users(id_user) ON DELETE CASCADE  
);
```

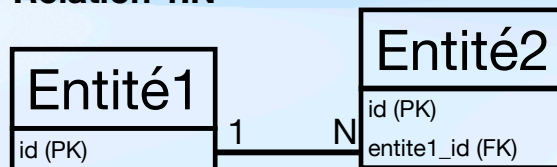
# Vocabulaire

## Base de données

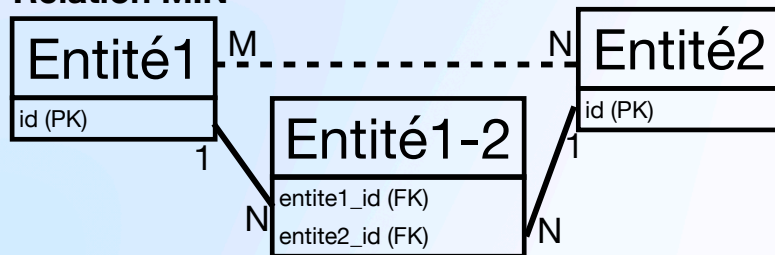
- Relation 1:1



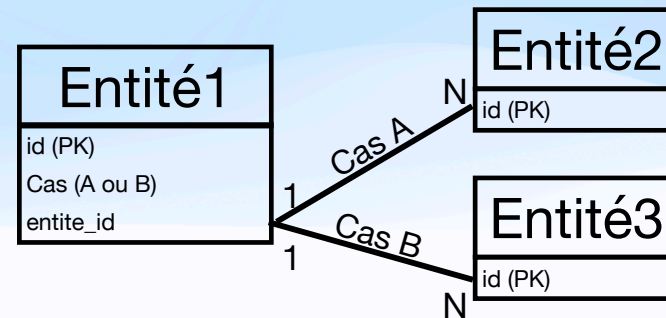
- Relation 1:N



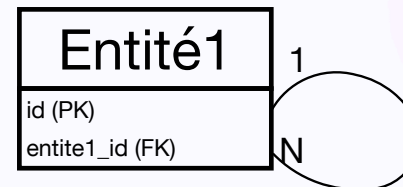
- Relation M:N



- Relation Polymorphique



- Relation Récursive



# **CRUD**

## **(Create, Read, Update, Delete)**

# CRUD

## Create - Création:

Ex: INSERT INTO users (... , nom, ...) VALUES (... , "Albert", ...)

```
<?php
$nom="";
if($_SERVER["REQUEST_METHOD"] == "POST"){
    ...
    $nom = trim($_POST["nom"]);
    ...
    <<INSERT IN DB>>
    ...
    header("location: index.php");
    exit();
}
?>
```

- ← Cas d'un POST du formulaire
- ← Récupération du nom
- ← Insertion dans la DB si les informations sont valides
- ← Retour à la page d'accueil

Sécurise l'URL

```
<body>
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
    ...
    <input type="text" name="nom" value="<?php echo $nom; ?>">
    ...
    <input type="submit" value="Enregistrer">
    <a href="« index.php">Annuler</a>
</form>
</body>
```

Chemin relatif complet de la page  
Ex: <https://www.exemple.com/dossier/page.php?id=5>  
► Retourne "/dossier/page.php"

# CRUD

## Read - Lecture

Ex: SELECT \* FROM users WHERE id = 3

### Appelant:

echo '<a href="read.php?id='.\$row['id'].'>Lecture</a>'; ← Appel de la page d'affichage avec le paramètre id

### read.php

```
<?php
    $if(isset($_GET["id"]) && !empty(trim($_GET["id"]))) {
        ...
        <<READ IN DB>>
        if(mysqli_num_rows($result) == 1) {
            ...
        }
    }
?>
```

← Cas d'un GET avec un paramètre id

← Si un seul enregistrement dans le résultat

```
<body>
    <label>Nom</label>
    ...
    <?php echo $row["Nom"]; ?>
    ...
</body>
```

← Affichage des informations

# CRUD

## Update - Mise à jour

Ex: UPDATE users SET nom="Albert", ... WHERE id=3

### Appelant:

```
echo '<a href="update.php?id='. $row['id'] .'>Mise à jour</a>';
```

### update.php (partie initialization)

```
<?php
    $if(isset($_GET["id"]) && !empty(trim($_GET["id"]))) {
        ...
        <<READ IN DB>>
        if(mysqli_num_rows($result) == 1) {
            ...
        }
    }
?>

<body>
    <label>Nom</label>
    ...
    <?php ... echo $row["Nom"]; ... ?>
    ...
</body>
```

### update.php (partie mise à jour)

```
<form action="<?php echo
htmlspecialchars(basename($_SERVER['REQUEST_URI'])); ?
" method="post">

<?php
    $if(isset($_POST["id"]) && !empty(trim($_POST["id"]))) {
        ...
        <<UPDATE IN DB>>
        ...
    }
?>
```

# CRUD

## Delete - Suppression Ex: DELETE FROM users WHERE id=3

### Appelant:

echo '<a href="delete.php?id='.\$row['id'].'>Supprimer</a>'; ← Appel de la page d'affichage avec le paramètre id

### delete.php

```
<?php
    $if(isset($_GET["id"]) && !empty(trim($_GET[« id »]])){ ← Cas d'un GET avec un paramètre id, on continue
        ...
        $if(isset($_POST["id"]) && !empty(trim($_POST[« id »]])){ ← Cas d'un POST avec un paramètre id, on supprime
            <<DELETE IN DB>>
            ...
        ?>

<body>
    <input type="hidden" name="id" value="<?php echo trim($_GET["id"]); ?>"/>
    <input type="submit" value="Supprimer">
</body>
```