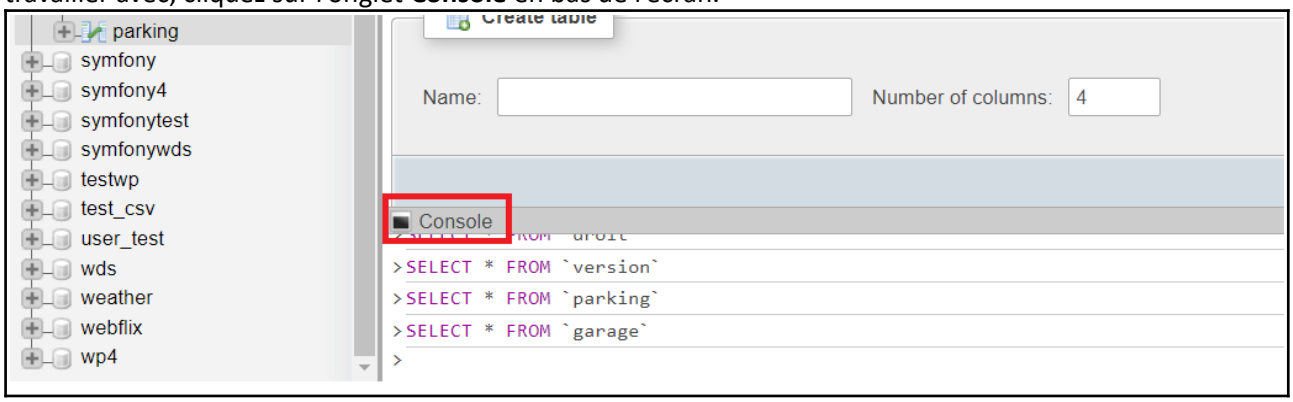
Requêtes SQL

Pour manipuler des données il faut savoir écrire des requêtes. Voici quelques indications pour commencer avec le SQL.

La console

PHPMyAdmin propose une console pour tester les requêtes directement sur les tables de l'interface. Pour travailler avec, cliquez sur l'onglet **Console** en bas de l'écran.



Pour l'exemple, nous travaillerons avec la table *Utilisateurs* suivante :

id_utilisateur	nom	prenom	naissance
1	Lannister	Tyrion	2000
2	Daenerys	Targaryen	2002
3	Jon	Snow	2002

Sélection de données

SELECT permet de sélectionner des données dans une table désignée par un **FROM** :

```
SELECT * FROM Utilisateurs ;
```

L'opérateur * permet la sélection d'une table complète.

Vous pouvez aussi choisir d'afficher une seule colonne, en la nommant :

```
SELECT nom FROM Utilisateurs ;
```

Ou plusieurs champs en les séparant par une virgule :

```
SELECT nom, prenom FROM Utilisateurs ;
```

Les colonnes sont toujours affichées dans l'ordre demandée dans le **SELECT**.

La clause WHERE

Le **WHERE** est une clause de condition de recherche très utile.

```
SELECT nom, prenom FROM Utilisateurs WHERE naissance = 2002 ;
```

Ce **SELECT** ne retournera que le nom et le prénom de l'utilisateur dont naissance=2002. La clause **WHERE** permet d'ajouter une condition à la requête. Il est possible de multiplier les conditions grâce à des opérateurs :

```
SELECT naissance FROM Utilisateurs WHERE nom='Lannister' AND prenom='Tyron' ;
```

Les opérateurs de conditions sont :

- **égale (=)**. Exemple : nom='Kikinou';
- **égale à NULL (IS, <=>)**. Exemple : nom<=>NULL;
- **différent (<>)**. Exemple : espece <> 'chat';
- **différent de NULL (IS NOT)**. Exemple : nom IS NOT NULL;
- **inférieur(<)**. Exemple : date_naissance < '2008-01-01';
- **supérieur(>)**. Exemple : date_naissance > '2008-01-01';
- **inférieur ou égal (<=)**.Exemple : date_naissance <= '2008-01-01';
- **supérieur ou égal(>=)**.Exemple : date_naissance >= '2008-01-01';
- **et(AND)**. Exemple : sexe='F' AND espece='chien';
- **ou(OR)**. Exemple : espece = ('chien' OR 'chat');
- **ou exclusif(XOR)**. Exemple : espece='tortue XOR naissance < '2008-01-01';
- **non(NOT, !)**. Exemple : espece != 'chat';

ORDER BY

La sélection est triée, dans l'ordre ascendant (ASC), par défaut. Le mot clé DESC permet de faire le tri dans le sens descendant. Il faut faire attention à l'ordre des colonnes lorsque plusieurs d'entre elles sont utilisées pour un tri. Le tri se fera dans l'ordre de nommage des colonnes.

```
SELECT * FROM Utilisateurs ORDER BY nom, prenom;
```

COUNT

Il est possible de compter un nombre d'enregistrements grâce à COUNT.

```
SELECT COUNT(*) FROM Utilisateur;
```

Pour compter le nombre de valeurs par colonnes, il faut utiliser le mot clé DISTINC :

```
SELECT COUNT(DISTINCT naissance) FROM Utilisateur;
```

Le COUNT peut être très efficace avec le GROUP BY, qui va permettre de compter le nombre d'enregistrements pour une valeur de colonne :

```
SELECT naissance, COUNT(*) FROM Utilisateur GROUP BY naissance;
```

Ajouter des données

La requête **INSERT INTO** permet d'ajouter de nouveaux enregistrements dans une table, en précisant les colonnes à remplir :

```
INSERT INTO Utilisateurs (nom, prenom, naissance) VALUES ('Lannister', 'Jaime', 1998),
('Tyrell', 'Margaery',2000);
```

Modifier des données

UPDATE remet à jour des données, sélectionnées à partir d'une clause **WHERE**:

```
UPDATE Utilisateurs SET naissance = 2001 WHERE nom='Snow';
```

Le **SET** définie la valeur à modifier.

Supprimer des données

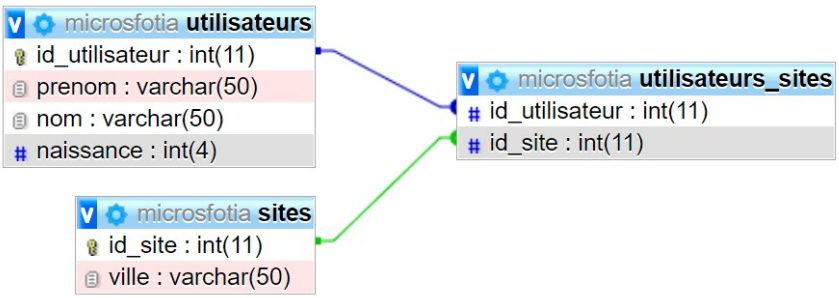
DELETE supprime des enregistrements. Attention à bien utiliser une clause WHERE pour sélectionner le bon enregistrement, sinon vous risquez de supprimer toute la table.

```
DELETE FROM Utilisateurs WHERE nom='Lannister' AND prenom='Jaime';
```

Jointures

Une jointure entre deux tables permet de combiner les données contenues dans les tables jointes. Grâce au relation mise en place dans la BDD, il va être possible de travailler avec plusieurs tables à la fois.

Soit pour exemple la base suivante :



Jointure naturelle

La jointure naturelle (**NATURAL JOIN**) permet de regrouper les lignes de table ayant les mêmes id pour les colonnes de même nom.

```
SELECT * FROM Utilisateur NATURAL JOIN utilisateurs_sites;
```

La multiple jointure est possible en ajoutant autant de **NATURAL JOIN** qu'utile :

```
SELECT * FROM Utilisateur NATURAL JOIN utilisateurs_sites NATURAL JOIN sites;
```

Il est possible d'appliquer une clause à l'une des colonnes de l'un des deux tableaux :

```
SELECT * FROM Utilisateur NATURAL JOIN utilisateurs_sites NATURAL JOIN sites WHERE
sites.ville = 'Port-Real';
```

Pour éviter toute ambiguïté avec les noms des colonnes, il est important de préciser le nom de la table avec le nom de la colonne, comme dans notre exemple.

Des problèmes peuvent survenir quand les tables ont des noms de colonnes identiques, en dehors des id. Ceci va perturber la jointure, puisque le SGBD va chercher à faire correspondre toutes les colonnes ayant le même nom.

Inner join

Il est alors possible de passer par **INNER JOIN**, ou jointure interne, pour préciser les colonnes à faire correspondre.

```
SELECT * FROM Utilisateur JOIN utilisateurs_sites ON Utilisateur.id_utilisateur =
utilisateurs_sites.id_utilisateur ;
```

Il est aussi possible de faire de multiples jointures :

```
SELECT Utilisateur.nom FROM Utilisateur
JOIN Utilisateur_site ON Utilisateur.id_utilisateur = utilisateurs_sites.id_utilisateur
JOIN Sites ON Utilisateur_site.id_site = Sites.id_site
WHERE Sites.ville = 'Port-Réal' ;
```