PHP & SQL

Gestion des contenus d'un

site web dynamique

Les bases de données

Pour un site web dynamique, la structure et le contenu sont deux éléments bien distincts. La première est régie par un système de pages PHP, le second est stocké à part de ce système, dans une base de données (BDD) sécurisée.

Les informations y sont organisées par tables, que l'on peut créer et structurer selon les besoins de l'application, avant de les renseigner.

Généralement, toute manipulation concernant les bases, tables et leurs contenus pourra être effectuée via des instructions écrites en langage SQL (Structured Query Language) ; on parle de requêtes, que l'on peut transmettre à la BDD pour y appliquer une modification ou y récupérer de l'information.

Ces requêtes peuvent en premier lieu être faites via un SGBD (Système de Gestion de Bases de Données), application web accessible sur le serveur, dont l'exemple le plus répandu est phpMyAdmin. Un tel système permet en outre d'obtenir une vision complète de la BDD, sa structure, ses contenus, et facilite également les opérations via une interface graphique, dispensant l'utilisateur d'écrire lui-même les requêtes SQL nécessaires.

Voici un exemple de table, correspondant à une liste (rudimentaire) d'articles de blog :

table **articles**

id	titre	date	contenu
1	Mon titre 1	2019-01-09	Lorem ipsum dolor sit amet…
2	Mon titre 2	2019-01-10	Utque aegrum corpus quas…
3	Mon titre 3	2019-01-28	Ut enim ad minim veniam…

Chaque article y correspondra à une ligne, dans laquelle on pourra accéder séparément aux différentes données qui le constituent ; dans cet exemple un titre, une date et le corps de son contenu.

Lors de la mise en place d'une table, on part du principe qu'une ligne correspondra à un élément d'une liste, et que les colonnes représenteront la structure interne commune à tous les éléments. Celle-ci est à déterminer selon les besoins de l'application destinée à accéder aux données.

Une quatrième colonne «id» est ici en place, afin de garantir un identifiant unique (et à priori immuable) pour chaque article. De manière générale, cela facilitera la mise en relation éventuelle avec d'autres tables ainsi que la récupération d'un élément bien précis ; on appelle cette colonne la clé primaire de la table.

La requête nécessaire à la création de la table ci-contre s'écrira ainsi :

```
CREATE TABLE `articles` (
  `id` INT(255) NOT NULL AUTO_INCREMENT ,
  `titre` TEXT ,
  `date` DATE ,
  `contenu` TEXT ,
  PRIMARY KEY (`id`)
);
```

• On commence par y nommer la table à créer, puis une liste des colonnes est établie entre parenthèses.

• Pour chaque colonne on spécifie un nom et un type de valeur attendu. «INT(255)» correspond à un nombre entier (à 255 chiffres maximum), et «DATE» à une valeur année/mois/jour valide, au format AAAA-MM-JJ.

• Pour la colonne «id», l'ajout des mots-clés «NOT NULL» lui interdit de contenir une valeur nulle, donc de ne pas être renseignée (un article devra toujours posséder un identifiant).

• Le mot-clé «AUTO_INCREMENT» permet que pour chaque nouvelle ligne, la valeur «id» soit automatiquement renseignée, et incrémentée par rapport à celle de la ligne précédente.

• Après la liste des colonnes, on précise que la clé primaire sera la colonne «id».

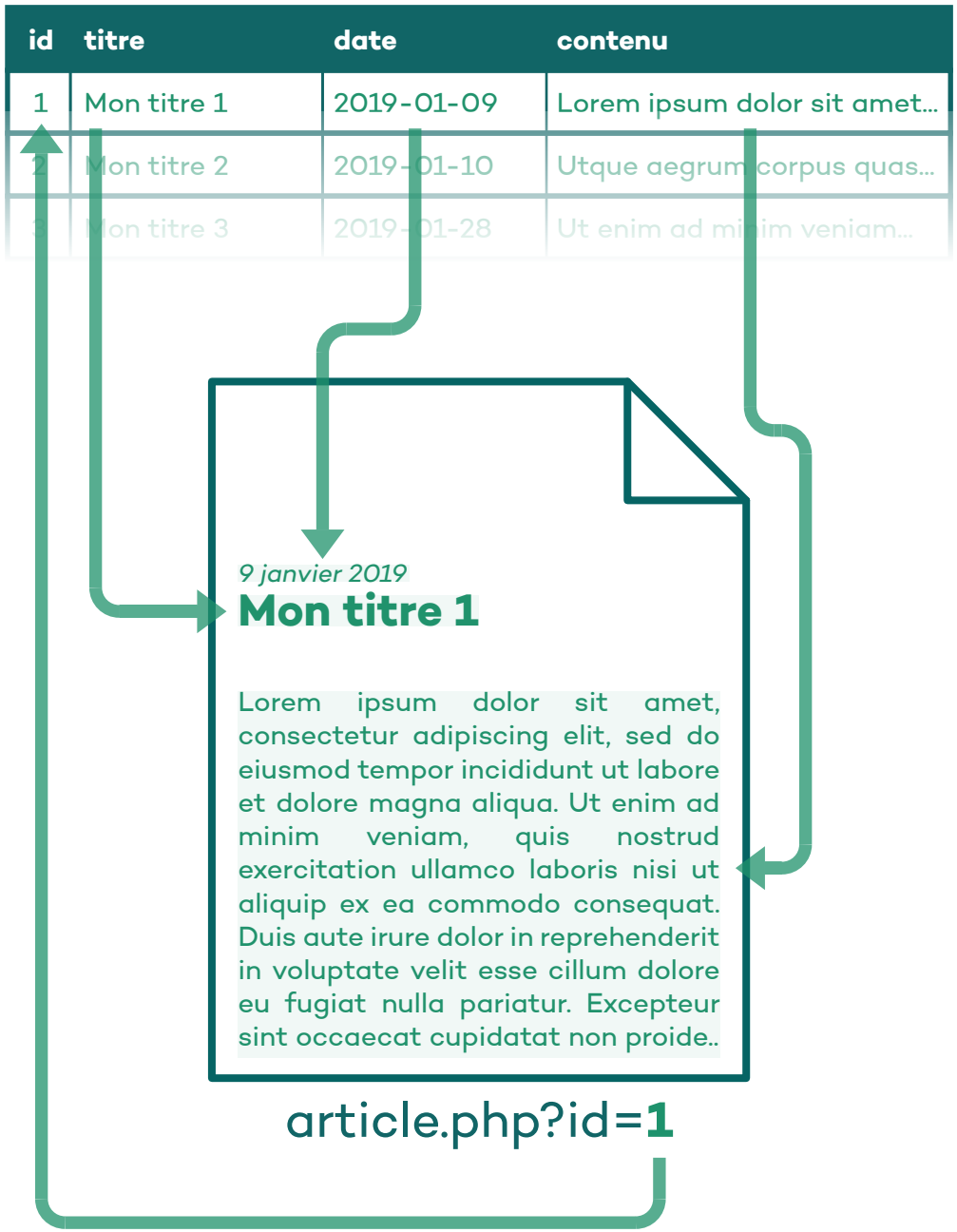
Récupération et affichage

Chaque élément de la table articles peut être affiché et mis en forme, via les pages «articles» d'un blog.

En apparence, chaque ligne de la table correspondra à une page web distincte, mais il est possible grâce à PHP de concevoir un gabarit HTML unique, dans lequel les données provenant de la BDD seront distribuées aux bons emplacements.

Le choix de l'article à afficher pourra dépendre de la manière dont l'utilisateur se rend sur la page-type (article.php), en se basant par exemple sur un identifiant ajouté en paramètre dans l'url.

article.php



Pour faire fonctionner cette logique, il faut que le fichier article.php inclue :

• Avant même l'ouverture de sa balise <html>, un bloc PHP destiné à récupérer dans l'URL l'identifiant de l'article voulu, s'en servir comme filtre dans une requête envoyée à la BDD et placer dans des variables les contenus récupérés : **recuperation-article.php**

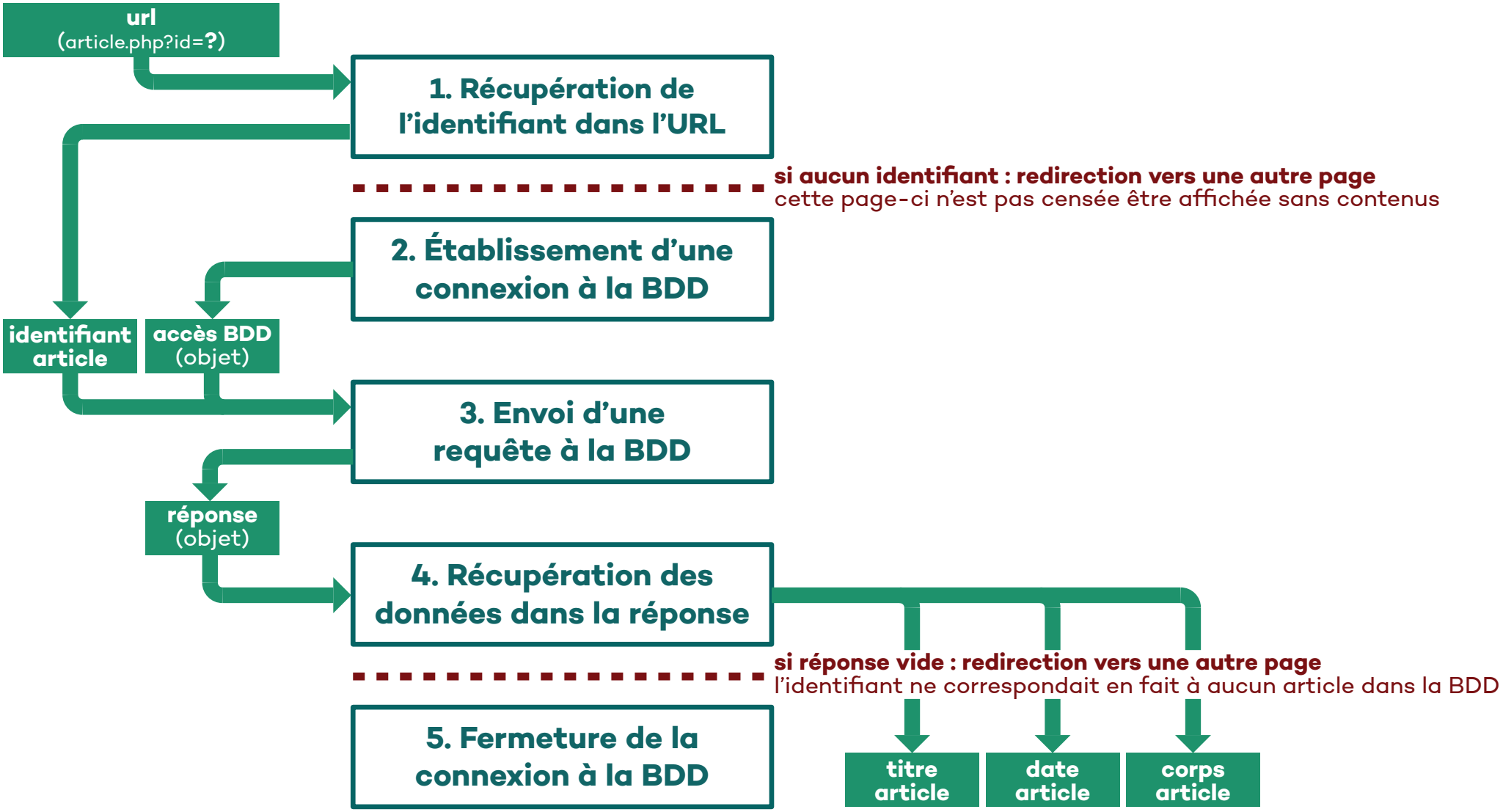
• Au bon emplacement dans le corps de page, un bloc PHP destiné à construire, en HTML, la structure d'un article en y insérant les variables mises à disposition par le premier bloc : **affichage-article.php**

• Additionnellement, on peut prévoir un bloc PHP à placer dans l'en-tête HTML, pour y insérer certaines des variables (par exemple faire figurer le titre de l'article dans la balise <title> de la page, ainsi que la première phrase du corps de l'article dans la description) : **balises-meta-article.php**

```
<?php
include("recuperation-article.php");
?>

<html>

    <head>
        ...

        <?php
        include("balises-meta-article.php");
        ?>

        ...
    </head>

    <body>
        ...

        <?php
        include("affichage-article.php");
        ?>

        ...
    </body>

</html>
```

article.php

recuperation-article.php - inclus en premier dans la page, avant l'ouverture <html>



recuperation-article.php - en détail :

1. Récupération de l'identifiant dans l'URL

```
if (isset($_GET["id"])) {
        $id_article = $_GET["id"];
} else {
        header("location:index.php");
}
```

Les paramètres passés dans l'URL sont accessibles via le tableau superglobal $_GET. On utilise la fonction isset() pour tester si un paramètre «id» est en place ; si oui on le stocke dans une variable $id_article, sinon l'utilisateur doit être redirigé vers une autre page, dans ce cas l'index.

2. Établissement d'une connexion à la BDD

```
$BDD_hote = "localhost";
$BDD_utilisateur = "root";
$BDD_motdepasse = "";
$BDD_base = "nom_de_la_BDD";
```

La fonction mysqli_connect() a besoin de quatre identifiants pour se connecter à la BDD. Ceux utilisés ici correspondent typiquement à la configuration par défaut d'un serveur local (MAMP / WAMP / etc). On stocke le résultat de la connexion dans une variable, pour l'utiliser à l'étape suivante.

```
$connexion = mysqli_connect($BDD_hote,$BDD_utilisateur,$BDD_motdepasse,$BDD_base);
```

3. Envoi d'une requête

```
$requete = "SELECT * FROM `articles` WHERE `id` = ".$id_article;

$reponse = $connexion->query($requete);
```

On ne veut récupérer qu'une ligne dans la table, celle pour laquelle l'identifiant correspond à celui récupéré dans l'URL de la page.

On utilise la méthode query() de l'objet $connexion établi précédemment, pour envoyer la requête SQL. La réponse à la requête doit être stockée, on en extraiera les informations nécessaires à l'étape suivante.

4. Récupération des données dans la réponse

```
if ($article = mysqli_fetch_array($reponse)) {
        $titre_article = $article["titre"];
        $date_article = $article["date"];
        $corps_article = $article["contenu"];
} else {
        mysqli_close($connexion);
        header("location:index.php");
}
```

La fonction mysqli_fetch_array() récupère une ligne à la fois dans une réponse SQL, et la présente sous forme d'un tableau associatif et numérique, où la valeur pour chaque colonne est accessible en utilisant comme clé le nom de la colonne ou comme indice la position de la colonne dans la liste. Par exemple ici $article["titre"] équivaut à $article[1] (seconde colonne).

Si on ne peut pas stocker le résultat de l'extraction dans la variable $article, cela signifie que la réponse ne contient aucune ligne ; il n'y a pas d'article pour l'identifiant demandé, il faut donc rediriger l'utilisateur.

5. Fermeture de la connexion

```
mysqli_close($connexion);
```

Fermer proprement toute connexion ouverte est une bonne pratique, permettant notamment d'éviter de multiplier les connexions simultanées. Selon la configuration du serveur, le nombre de connexions simultanées à la BDD autorisées sera plus ou moins limité.

Une fois ce script exécuté, on dispose donc sur la page de trois variables contenant le titre, la date et le corps de l'article à afficher. Placer ce script en tout début de page permet, en plus d'avoir accès à ces variables dès l'en-tête HTML, de faire fonctionner les éventuelles redirections : la fonction header() modifie l'en-tête HTTP de la page, et nécessite qu'aucun contenu n'ait été envoyé auparavant.

affichage-article.php - inclus dans le corps du HTML, à l'emplacement où l'article doit être affiché :

```
<article>

    <h1><?php echo $titre_article; ?></h1>

    <p><?php echo formatage_date($date_article); ?></p>

    <p><?php echo nl2br($corps_article); ?></p>

</article>
```

Il s'agit de construire la structure HTML de l'article, en y insérant les variables mises à disposition en ouverture de page par recuperation-article.php.

formatage_date() est une fonction personnalisée (voir page suivante).

La fonction nl2br() permet de convertir, dans une chaine de caractères, les retours à la ligne en leurs équivalents HTML (balises
). Sans cela, ils ne seront pas visibles dans le rendu HTML du corps de l'article.

balises-meta-article.php - inclus dans l'en-tête HTML, à la place des balises <title> et <meta name = "description"> :

```
<title>Mon Blog - <?php echo $titre_article; ?></title>

<meta name = "description" content = "<?php echo isoler_premiere_phrase($corps_article); ?>" />
```

isoler_premiere_phrase() est une fonction personnalisée (voir page suivante).

Fonctions personnalisées - à définir par défaut pour chacune au début du script où elle sera utilisée :

```
function formatage_date($date_sql) {

   $timestamp = strtotime($date_sql);

   $date_formatee = date("d/m/Y", $timestamp);

   return $date_formatee;

}
```

Permettra de transformer par exemple «2019-01-31» en «31/01/2019»

strtotime() convertit une chaine de caractères en un timestamp Unix (mesure temporelle, nombre de secondes écoulées depuis le 1er janvier 1970), à condition de pouvoir reconnaître dans cette chaîne une date valide. date() permet ensuite, à partir de cette valeur, un formatage (défini par l'expression passée en premier paramètre - voir documentation sur la fonction date() de PHP).

```
function isoler_premiere_phrase($chaine) {

   if ($pos_premier_point = strpos($chaine, ".")) {
      $premiere_phrase = substr($chaine, 0, $pos_premier_point);
   } else {
      $premiere_phrase = $chaine;
   }

   return $premiere_phrase;

}
```

Cherchera dans $chaine le premier point. S'il existe, on part du principe que tout ce qui le précède constitue la première phrase de $chaine. Sinon, on considère que $chaine n'est en fait qu'une seule phrase.

strpos() permet de trouver la position, dans une chaine (paramètre 1), de la première occurrence d'une expression (paramètre 2).

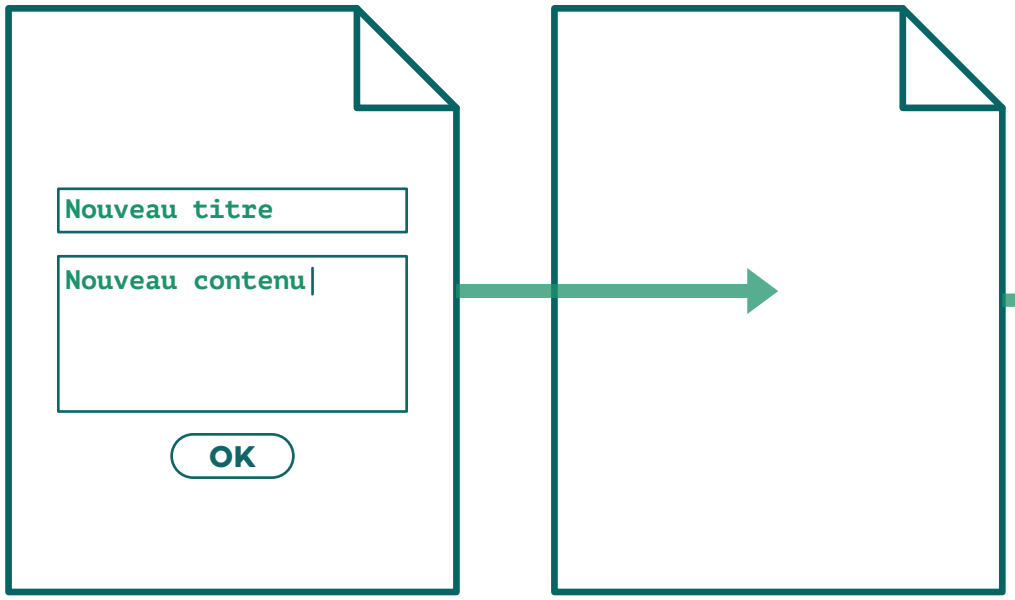
substr() permet d'isoler une partie d'une chaine (paramètre 1). Le paramètre 2 détermine la position où commence cette partie, le paramètre 3 détermine sa longueur. Ce dernier est optionnel ; en son absence la fin de la partie isolée correspond à la fin de la chaine initiale.

On constate que chacune de ces fonctions «retourne» un résultat, que l'on pourra ensuite manipuler (dans notre exmple les résultats sont affichés via echo).

Envoi d'informations vers la BDD

Pour la mise en place de contenus dynamiques, il faut prévoir une page depuis laquelle des informations peuvent être renseignées par l'utilisateur, puis envoyées vers la base de données.

Cela nécessitera deux choses : un formulaire HTML pour la saisie des données du côté utilisateur (**formulaire-article.php**), ainsi qu'un script PHP pour le traitement et l'envoi de ces données dans la table correspondante (**ajout-article.php**).



id	titre	date	contenu
1	Mon titre 1	2019-01-09	Lorem ipsum dolor sit amet...
2	Mon titre 2	2019-01-10	Utque aegrum corpus quas...
3	Mon titre 3	2019-01-28	Ut enim ad minim veniam...
4	**Nouveau titre**	**2019-01-31**	**Nouveau contenu**

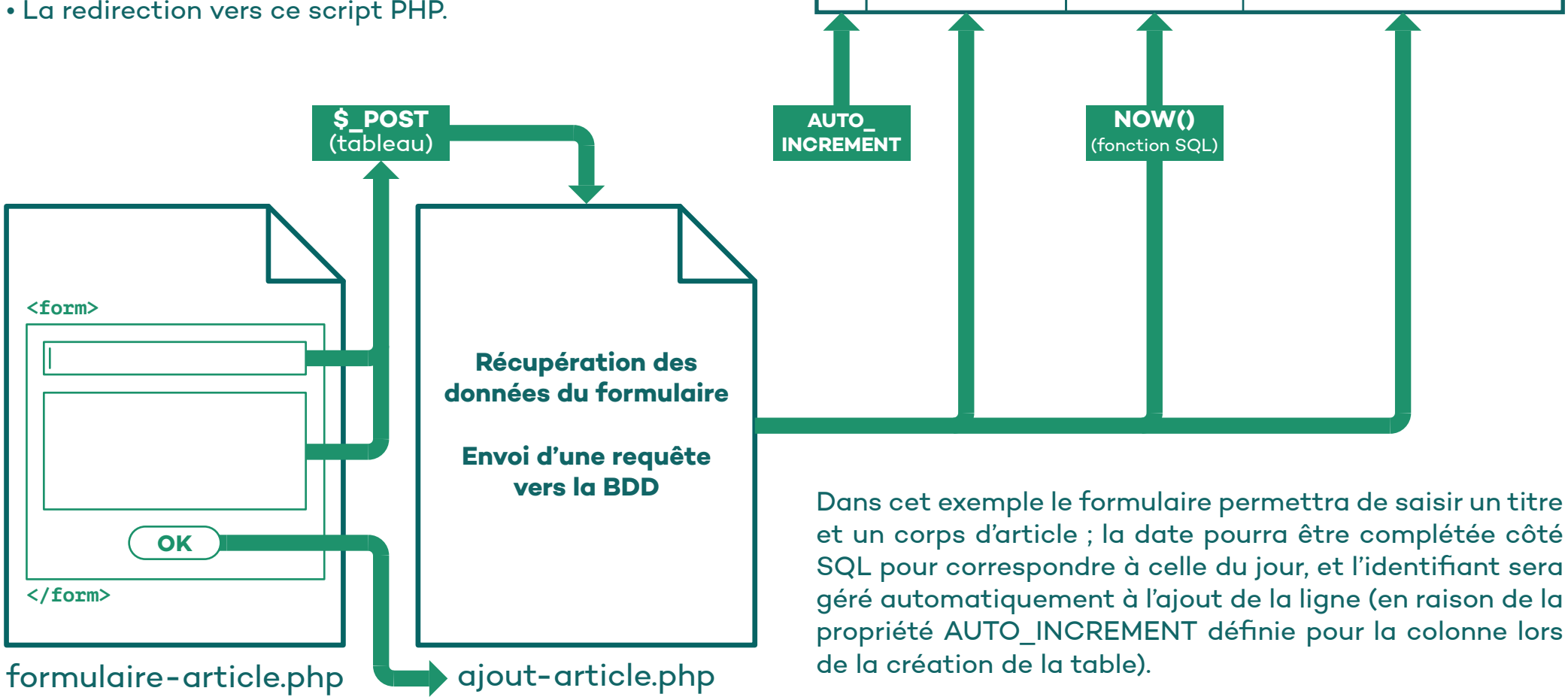
Envoi d'informations vers la BDD - ajout d'un article

La mise en place d'un formulaire HTML permet, lors de sa validation :

• Le stockage temporaire des données saisies dans un tableau superglobal (ici nous utiliserons $_POST), accessible depuis un script PHP externe qui traitera ces données

• La redirection vers ce script PHP.

id	titre	date	contenu
1	Mon titre 1	2019-01-09	Lorem ipsum dolor sit amet...
2	Mon titre 2	2019-01-10	Utque aegrum corpus quas...
3	Mon titre 3	2019-01-28	Ut enim ad minim veniam...
4	**Nouveau titre**	**2019-01-31**	**Nouveau contenu**



Dans cet exemple le formulaire permettra de saisir un titre et un corps d'article ; la date pourra être complétée côté SQL pour correspondre à celle du jour, et l'identifiant sera géré automatiquement à l'ajout de la ligne (en raison de la propriété AUTO_INCREMENT définie pour la colonne lors de la création de la table).

formulaire-article.php - en détail :

```
<form method = "post" action = "ajout-article.php">

    <input type = "text" name = "champ_titre" required />

    <textarea name = "champ_corps" ></textarea>

    <input type = "submit" value = "OK" />

</form>
```

Bien que placée dans un fichier PHP, cette structure ne consiste en rien d'autre que des éléments HTML. Il suffira de donner à ceux-ci quelques attributs spécifiques pour que le «passage de relais» au script de traitement se fasse correctement.

• Un élément <form> (le formulaire en lui-même) doit englober tous les éléments de saisie. Son attribut «method» indique la manière de transmettre les informations au script de traitement («post» ou «get»). L'avantage de «post» est que les informations sont transmises de manière invisible, au lieu de s'ajouter en paramètres de l'url du script. L'attribut «action» correspond au chemin vers le script de traitement.

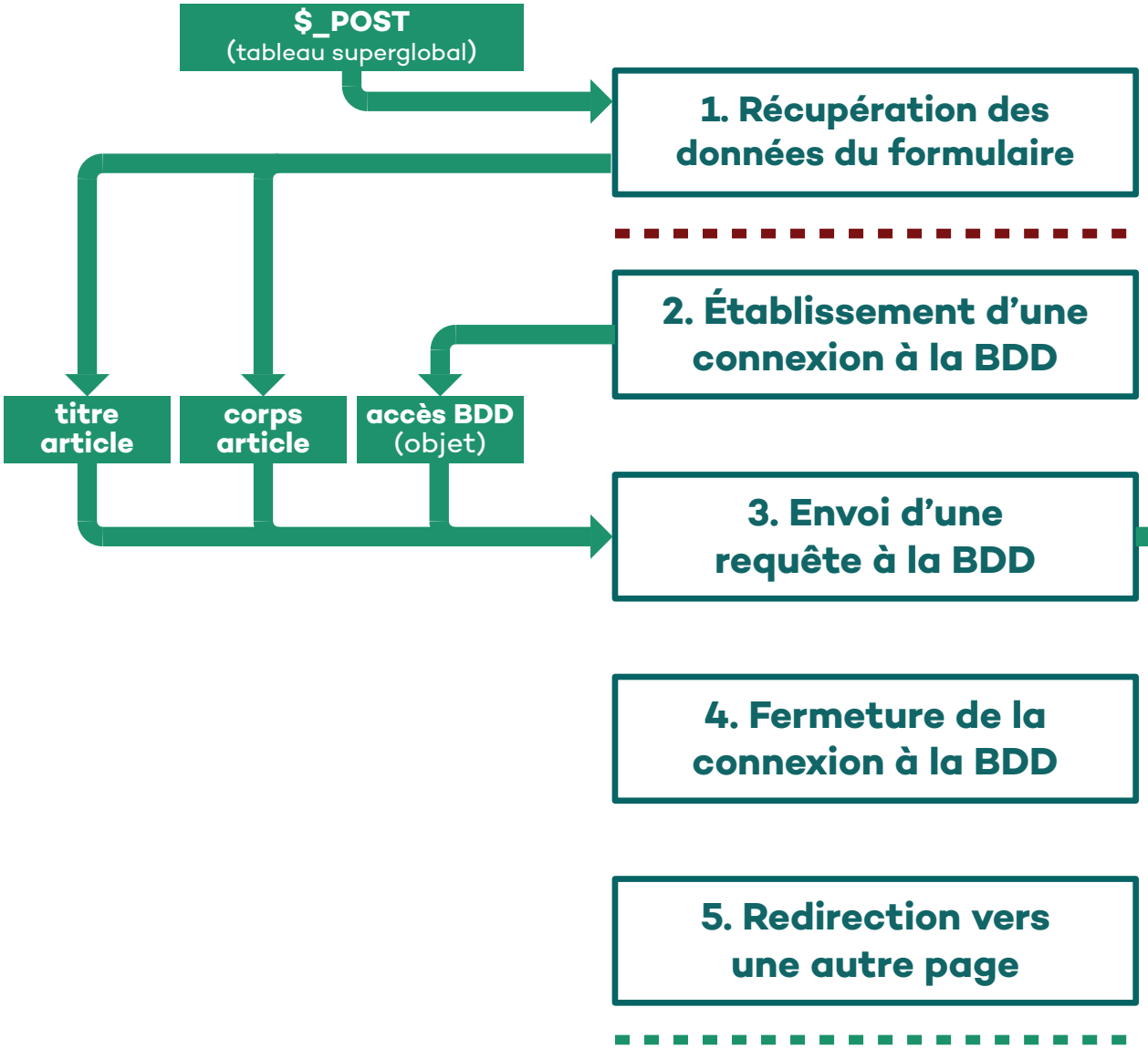
• Un élément <input> de type «submit» sera automatiquement reconnu comme bouton de validation du formulaire. Au clic, il redirigera l'utilisateur vers le script de traitement (dont le chemin est précisé en attribut «action» du formulaire).

• L'attribut «name» est important pour les éléments dont on doit récupérer les informations saisies. Lors de la validation, chacun verra son contenu stocké dans le tableau $_POST, dans lequel on pourra le récupérer en utilisant comme clé, pour chaque valeur, l'attribut «name» de l'élément correspondant dans le formulaire.

• L'attribut «required» (correspondant implicitement à « required = "true" ») permet d'empêcher la validation du formulaire si l'élément concerné n'est pas renseigné. Ici nous partons du principe que le titre est obligatoire.

• L'élément <input type = "text" /> correspond à un champ texte sur une seule ligne, l'élément <textarea> (nécessitant une balise fermante) correspond à un champ multi-lignes.

ajout-article.php - script de traitement des données



si aucune donnée en place :
abandon et redirection vers une autre page
ce n'est de toute évidence pas l'envoi du formulaire qui a amené l'utilisateur vers ce script, or il n'est prévuque pour ce cas spécifique.

Article ajouté à la table

ajout-article.php - en détail :

1. Récupération des données du formulaire

```
if (isset($_POST["champ_titre"]) && isset($_POST["champ_corps"])) {
       $titre_article = $_POST["champ_titre"];
       $corps_article = $_POST["champ_corps"];
} else {
       header("location:index-admin.php");
}
```

Les informations provenant du formulaire sont accessibles via le tableau superglobal $_POST. La logique de vérification / récupération / redirection est la même que pour recuperation-article.php (voir page 6).

2. Établissement d'une connexion à la BDD

```
$BDD_hote = "localhost";
$BDD_utilisateur = "root";
$BDD_motdepasse = "";
$BDD_base = "nom_de_la_BDD";

$connexion = mysqli_connect($BDD_hote,$BDD_utilisateur,$BDD_motdepasse,$BDD_base);
```

Ce segment est le même que pour recuperation-article.php (voir page 6).

3. Envoi d'une requête

```
$requete = "INSERT INTO `articles`
            (`titre`, `contenu`, `date`)
            VALUES
            ('$titre_article', '$corps_article', NOW())";

$connexion->query($requete);
```

Pour une requête d'ajout de ligne, il faut préciser les colonnes concernées puis les valeurs à y insérer, en faisant correspondre l'ordre des deux listes. Pour la colonne «date», on utilise ici la fonction SQL NOW(), désignant automatiquement la date du jour (au moment de l'ajout).

Il ne sera pas nécessaire dans ce script de manipuler de réponse provenant de la BDD, on ne stocke donc pas le résultat de l'appel à la méthode $connexion->query().

4. Fermeture de la connexion

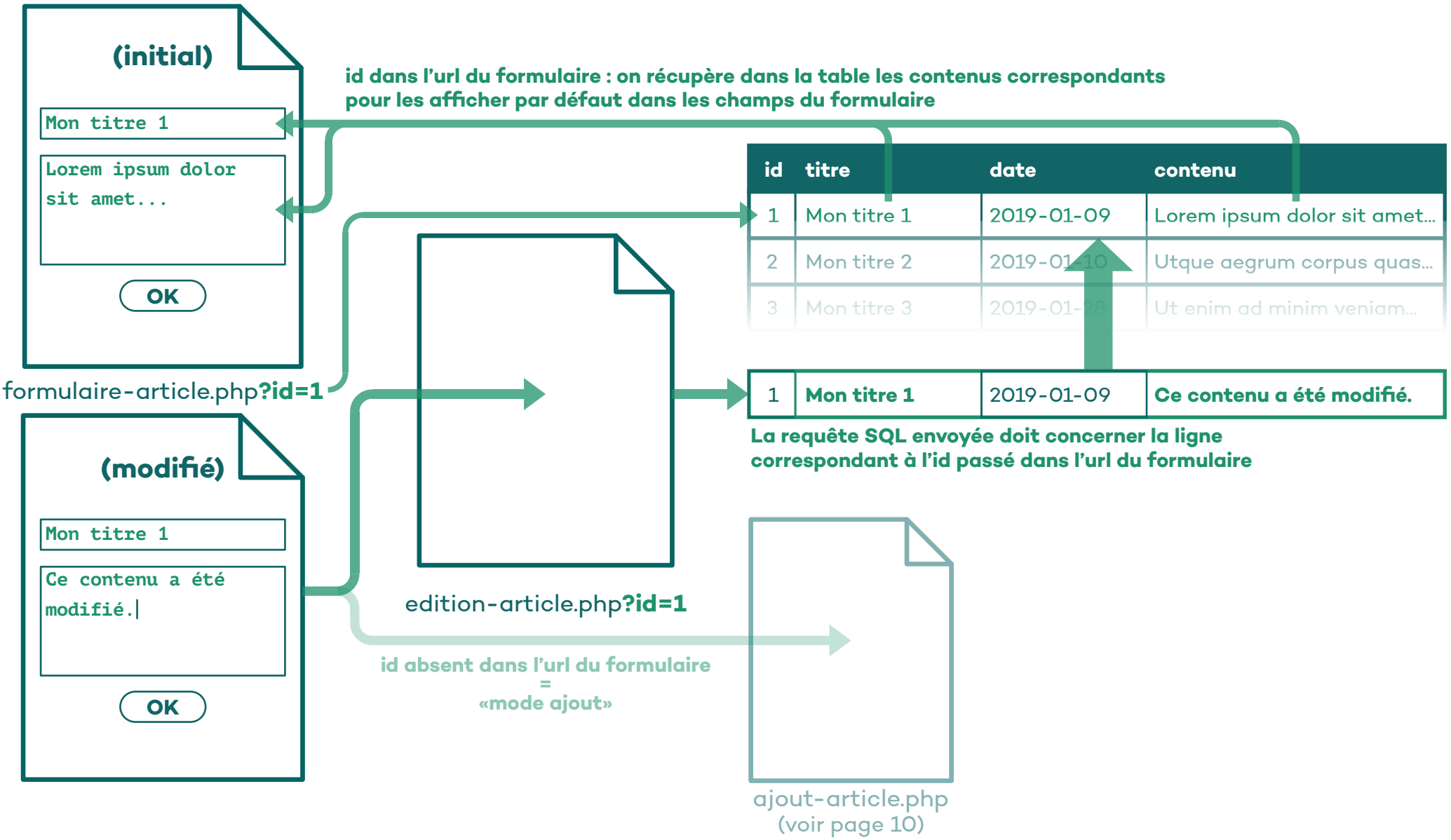
```
mysqli_close($connexion);
```

5. Redirection vers une autre page

```
header("location:index-admin.php");
```

Ce script n'est pas fait pour être «vu» par l'utilisateur, on l'en sort donc dès que l'envoi est terminé (en apparence : instantanément, le passage par ce script sera invisible dans le navigateur).

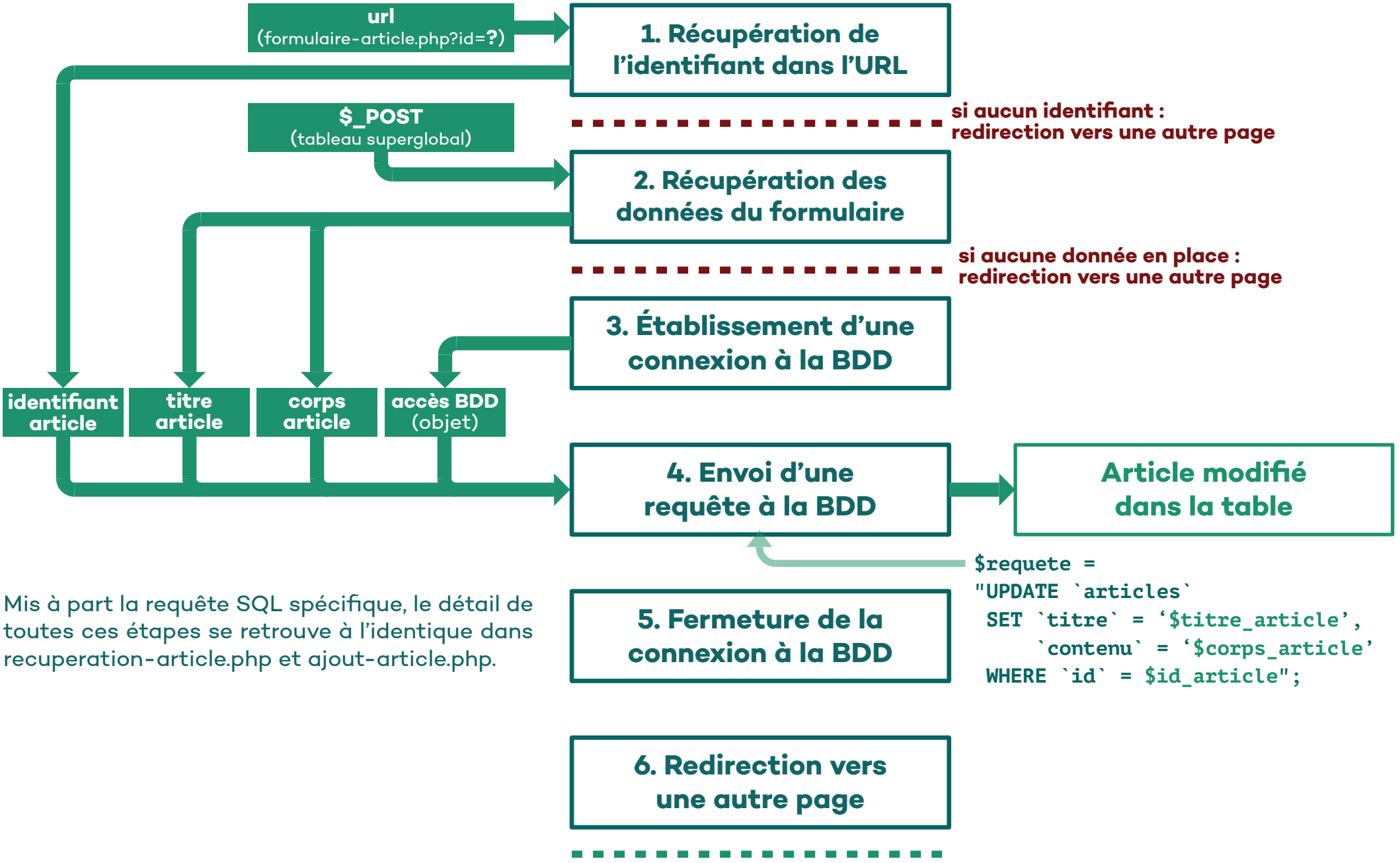
Envoi d'informations vers la BDD - modification d'un article existant



formulaire-article.php - modifié pour permettre un «mode édition» :

```php
<?php

if (isset($_GET["id"])) { // mode édition

   $script = "edition-article.php?id=".$_GET["id"];

   include("recuperation-article.php");
   // pour obtenir $titre_article et $corps_article - voir page 5

} else { // mode ajout

   $script = "ajout-article.php";
   $titre_article = "";
   $corps_article = "";

}

?>

<form method = "post" action = "<?php echo $script; ?>">

   <input type = "text" name = "champ_titre" value = "<?php echo $titre_article; ?>" required />

   <textarea name = "champ_corps" ><?php echo $corps_article; ?></textarea>

   <input type = "submit" value = "OK" />

</form>
```

edition-article.php



Mis à part la requête SQL spécifique, le détail de toutes ces étapes se retrouve à l'identique dans recuperation-article.php et ajout-article.php.

ANNEXE - Le problème éventuel des apostrophes
dans les contenus envoyés à la BDD

Selon la configuration du serveur, l'envoi de la requête suivante peut ne pas aboutir :

```
$valeur = "l'apostrophe";
$requete = "INSERT INTO `nom_table` (`nom_colonne`) VALUES ('$valeur')";
```

L'apostrophe contenue dans la chaine $valeur, si elle n'est pas automatiquement échappée, occasionnera un conflit en «fermant» prématurément l'emplacement prévu dans la requête pour la chaine à insérer, celui-ci étant délimité par des guillemets simples, utilisés également la plupart du temps comme apostrophes. La requête ne sera donc pas complète et n'aura aucun effet.

```
> INSERT INTO `nom_table` (`nom_colonne`) VALUES ('l'apostrophe')
```

Dans ce cas on peut utiliser la fonction PHP addslashes(), qui ajoutera automatiquement à une chaine les anti-slash nécessaires à l'échappement des caractères posant problème :

```
$valeur = "l'apostrophe";
$requete = "INSERT INTO `nom_table` (`nom_colonne`) VALUES ('".addslashes($valeur)."')";

> INSERT INTO `nom_table` (`nom_colonne`) VALUES ('l\'apostrophe')
```

Notons qu'il existe également une fonction stripslashes() pour obtenir l'inverse : elle supprimera les anti-slash de la chaine passée en paramètre.

ANNEXE - mysqli_fetch_array() : utilisation récursive

En considérant $reponse, résultat d'une requête récupérant toutes les lignes d'une table possédant plusieurs éléments :

```
$ligne = mysqli_fetch_array($reponse);
echo $ligne["nom_colonne"];
// affichera la valeur trouvée dans la colonne "nom_colonne" de la première ligne à récupérer

$ligne = mysqli_fetch_array($reponse);
echo $ligne["nom_colonne"];
// affichera la valeur trouvée dans la colonne "nom_colonne" de la SECONDE ligne à récupérer

// etc
```

Automatiquement, à chaque appel de mysqli_fetch_array() pour l'objet $reponse, la fonction «progressera» vers la ligne suivante à récupérer. La syntaxe suivante est donc possible pour passer en revue toutes les lignes à récupérer :

```
while ($ligne = mysqli_fetch_array($reponse)) {
   echo $ligne["nom_colonne"];
}
/* La boucle s'exécutera tant qu'il sera possible d'attribuer à $ligne le résultat de
mysqli_fetch_array($reponse).
Elle affichera à la suite toutes les valeurs trouvées dans la colonne "nom_colonne", pour chaque ligne à
récupérer. */
```

Pour gérer le cas où $reponse est vide (et la boucle ne s'exécute jamais), on peut ajouter à la suite :

```
if (mysqli_num_rows($reponse) == 0) {
   // Ce bloc s'exécutera si le nombre de lignes contenues dans $reponse est égal à 0
}
```