Description de la base de données

Sommaire

1 MCD 3
2 Détail des tables 4
2.1 _logs_types 4
2.2 _logs 4
2.3 _listings 5
2.4 _groupes_fonctionnalites 5
2.5 _fonctionnalites 6
2.6 _profils 6
2.7 _droits 7
2.8 _users 7
2.9 _params 9

1 MCD

La figure 1 montre le MCD de la base de données créée pour UniversalWeb. Nous allons étudier en détail la fonction et le contenu de chacune de ces tables.

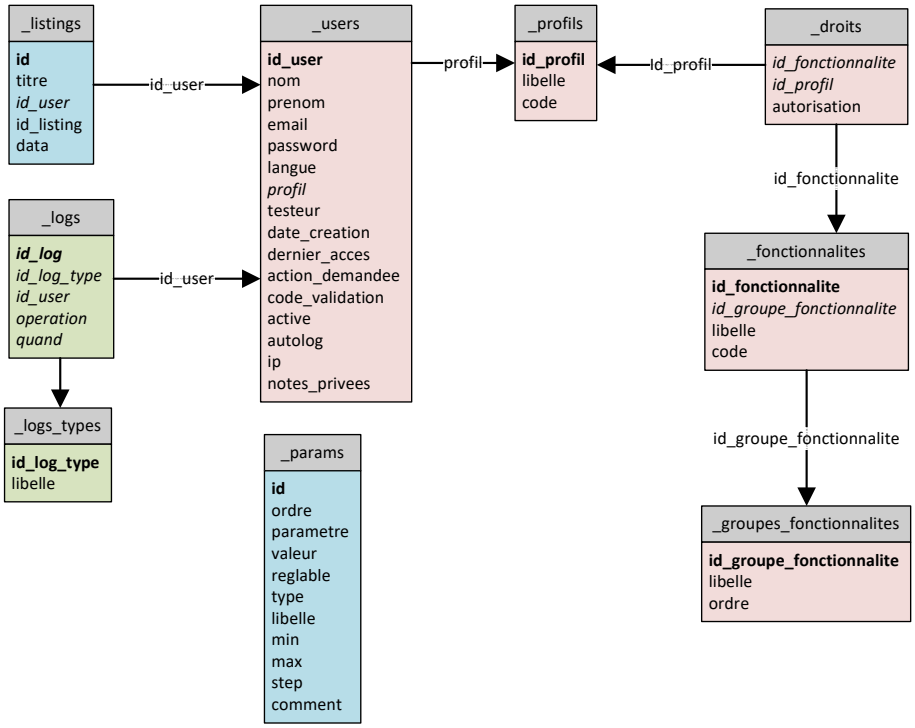

Figure 1 : MCD (fonctions regroupées par couleurs)

Rappel de certains types de données

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

L'attribut passé entre parenthèse à un type d'entier permet de n'afficher qu'un certain nombre de digits. Par exemple même si on peut saisir une valeur de signée de 0 à 255, TYNYINT(1) ne permet d'afficher que les valeurs de 0 à 9. TYNYINT(3) affichera la totalité du spectre de ce très petit entier (0..255)

Le MCD UniversalWeb respecte la première forme normale.

Afin de garantir l'intégrité de bout en bout de l'encodage en `UTF-8` de l'application, l'interclassement choisi de la base de données ainsi que de chaque table est `utf8_general_ci`. En conséquence, chaque champ textuel de vos tables devront également utiliser cet interclassement.

2 Détail des tables

2.1 _logs_types

Cette table contient tous les types de journaux disponibles pour votre application.

```
CREATE TABLE IF NOT EXISTS `_logs_types` (
  `id_log_type` tinyint(3) UNSIGNED NOT NULL,
  `libelle` varchar(128) NOT NULL,
  PRIMARY KEY (`id_log_type`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

id_log_type
Identifiant du type de log.
Entier non signé (0..255). **Clé primaire**.

libelle
Libellé du type de log.
128 caractères max.

Un seul type de log est prédéfinis, « Connexion », chargé de journaliser les connexions à l'application.

```
INSERT INTO `_logs_types` (`id_log_type`, `libelle`) VALUES
(1, 'Connexion');
```

2.2 _logs

Table de sauvegarde des journaux.

```
CREATE TABLE IF NOT EXISTS `_logs` (
  `id_log` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `id_log_type` tinyint(3) UNSIGNED NOT NULL,
  `id_user` varchar(100) NOT NULL,
  `operation` varchar(255) NOT NULL,
  `quand` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id_log`),
  KEY `id_user` (`id_user`),
  KEY `quand` (`quand`)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

id_log
Identifiant de l'entrée du journal.
*Entier non signé (0..*4294967295*30)*. **Clé primaire, auto-incrément**.

id_log_type
Identifiant du type d'entrée de journal. Il s'agit de l'identifiant fourni dans la table `_logs_type`.
*Entier non signé (0..*255*)*. **Clé étrangère** (table `_logs_types`).

id_user
Identifiant de l'utilisateur à loguer.
100 caractères max. **Clé étrangère** (table `_users`).

operation
Entrée de journal (votre information à loguer).
255 caractères max.

quand
Date et heure de l'entrée au journal au format *timestamp unix*. Il n'est pas nécessaire pour le développeur de saisir ce champ, celui-ci est automatiquement rempli par MySQL (CURRENT_TIMESTAMP).

timestamp. **Index**.

2.3 _listings

Table de sauvegarde des listes **UniversalList**. Voir la documentation correspondante à cette classe.

```
CREATE TABLE IF NOT EXISTS `_listings` (
  `id` tinyint(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `titre` varchar(255) NOT NULL,
  `id_user` varchar(100) NOT NULL,
  `id_listing` varchar(30) NOT NULL,
  `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `data` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `titre` (`titre`),
  KEY `id_user_2` (`id_user`),
  KEY `id_listing` (`id_listing`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

id
Identifiant de la liste complexe.
Entier non signé (0..255). **Clé primaire**, **auto-incrément**.

titre
Titre donné à la liste complexe.
255 caractères max. **Index**.

id_user
Identifiant de l'utilisateur qui a créé la liste complexe. **Clé étrangère** (table `_users`)
100 caractères max.

id_listing
Identifiant du type de liste. Pour les différencier, chaque type de liste reçoit un terme mnémonique de reconnaissance qui peut être utilisé directement par votre application.
30 caractères max. **Index**.

last_update
Date de dernière création ou de dernière modification de la liste. Ce champ se met automatiquement à jour (CURRENT_TIMESTAMP) dès que le tuple est modifié par l'application. Le développeur n'a donc pas à le mettre à jour par lui-même.
timestamp.

data
Description de la liste complexe (voir outil **UniversalList** de **UniversalWeb**).
Texte.

2.4 _groupes_fonctionnalites

Table permettant de regrouper les fonctionnalités de l'application par groupes et ainsi de proposer un affichage plus compact et plus efficace depuis l'interface du backend dans le cas de nombreuses fonctionnalités.

```
CREATE TABLE IF NOT EXISTS `_groupes_fonctionnalites` (
  `id_groupe_fonctionnalite` tinyint(3) UNSIGNED NOT NULL,
  `libelle` varchar(128) NOT NULL,
  `ordre` tinyint(3) UNSIGNED NOT NULL DEFAULT '1',
  PRIMARY KEY (`id_groupe_fonctionnalite`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

id_groupe_fonctionnalite
Identifiant du groupe de fonctionnalités.

*Entier non signé (0..*255*)*. **Clé primaire**.

libelle
Libellé en clair du groupe de fonctionnalités.
128 caractères max. **Clé unique**.

ordre
Ordre d'affichage du groupe de fonctionnalité sur l'interface backend.
*Entier non signé (0..*255*)*.

Deux groupes de fonctionnalités sont définis de base dans UniversalWeb :

```
INSERT INTO `uw_groupes_fonctionnalites` (`id_groupe_fonctionnalite`, `libelle`, `ordre`) VALUES
(1, 'Non classée', 1),
(2, 'Administration', 3);
```

2.5 _fonctionnalites

Table comportant les différentes fonctionnalités de l'application.

```
CREATE TABLE IF NOT EXISTS `uw_fonctionnalites` (
  `id_fonctionnalite` tinyint(3) UNSIGNED NOT NULL,
  `id_groupe_fonctionnalite` tinyint(3) UNSIGNED NOT NULL DEFAULT '1',
  `libelle` varchar(128) NOT NULL,
  `code` varchar(30) NOT NULL,
  PRIMARY KEY (`id_fonctionnalite`),
  UNIQUE KEY `code` (`code`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

id_fonctionnalite
Identifiant de la fonctionnalité.
*Entier non signé (0..*255*)*. **Clé primaire**.

id_groupe_fonctionnalite
Identifiant du groupe de fonctionnalités auquel est rattaché la fonctionnalité (voir table `_groupes_fonctionnalites`).
*Entier non signé (0..*255*)*.

libelle
Libellé en clair de la fonctionnalité.
128 caractères max.

code
Code mnémonique à utiliser par le développeur via les méthodes fournies pour faire référence à la fonctionnalité dans son code.
30 caractères max. **Clé unique**.

Une seule fonctionnalité est définie de base dans UniversalWeb :

```
INSERT INTO `_fonctionnalites` (`id_fonctionnalite`, `id_groupe_fonctionnalite`, `libelle`, `code`) VALUES
(1, 2, 'Administrer l\'application', 'FONC_ADM_APP');
```

2.6 _profils

Table regroupant les différents profils utilisateurs de l'application.

```
CREATE TABLE IF NOT EXISTS `_profils` (
  `id_profil` tinyint(3) UNSIGNED NOT NULL,
  `libelle` varchar(30) NOT NULL,
  `code` varchar(30) NOT NULL,
  PRIMARY KEY (`id_profil`),
  UNIQUE KEY `code` (`code`)
```

```
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

id_profil
Identifiant du profil utilisateur.
*Entier non signé (0..*255*)*. **Clé primaire**.

libelle
Libellé en clair du profil.
30 caractères max.

code
Code mnémonique à utiliser par le développeur via les méthodes fournies pour faire référence au profil dans son code.
30 caractères max. **Clé unique**.

Deux profils sont définis de base : Administrateur et Visiteur.

```
INSERT INTO `_profils` (`id_profil`, `libelle`, `code`) VALUES
(1, 'Administrateur', 'PROFIL_ADMIN'),
(0, 'Visiteur', 'PROFIL_VISITEUR');
```

2.7 _droits

Table distribuant les droits d'accès à une fonctionnalité pour chaque profil d'utilisateur.

```
CREATE TABLE IF NOT EXISTS `_droits` (
  `id_fonctionnalite` tinyint(3) UNSIGNED NOT NULL,
  `id_profil` tinyint(3) UNSIGNED NOT NULL,
  `autorisation` tinyint(1) UNSIGNED NOT NULL,
  UNIQUE KEY `fonctionalite` (`id_fonctionnalite`,`id_profil`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

id_fonctionnalite
Identifiant de la fonctionnalité.
*Entier non signé (0..*255*)*. **Clé étrangère** (table `_fonctionnalites`). **Clé primaire composée** avec `id_profil`.

id_profil
Identifiant du profil utilisateur.
*Entier non signé (0..*255*)*. **Clé étrangère** (table `_profils`). **Clé primaire composée** avec `id_fonctionnalite`.

autorisation
Autorisation finale pour le couple fonctionnalité / profil.
Booléen (1 = accès autorisé ; 0 = accès interdit).

Un seul droit est par défaut posé : administrer l'application pour le profil administrateur.

```
INSERT INTO `_droits` (`id_fonctionnalite`, `id_profil`, `autorisation`) VALUES
(1, 1, 1);
```

2.8 _users

Table des utilisateurs de l'application. Tous les champs ne sont pas utilisés dans le backend mais pourraient très bien l'être dans le frontend pour la gestion de clients par exemples. Ils sont directement exploitables par le développeur.

```
CREATE TABLE IF NOT EXISTS `_users` (
  `id_user` varchar(100) NOT NULL,
  `nom` varchar(100) NOT NULL,
```

```
  `prenom` varchar(100) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(40) NOT NULL,
  `langue` varchar(2) NOT NULL DEFAULT 'fr',
  `profil` tinyint(3) UNSIGNED NOT NULL DEFAULT '0',
  `testeur` tinyint(1) UNSIGNED NOT NULL DEFAULT '0',
  `date_creation` datetime DEFAULT NULL,
  `dernier_acces` datetime DEFAULT NULL,
  `action_demandee` tinyint(3) UNSIGNED NOT NULL DEFAULT '0',
  `code_validation` varchar(30) NOT NULL,
  `active` tinyint(1) UNSIGNED NOT NULL DEFAULT '0',
  `autolog` tinyint(1) UNSIGNED NOT NULL DEFAULT '0',
  `ip` varchar(39) NOT NULL,
  `notes_privees` text NOT NULL,
  PRIMARY KEY (`id_user`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 PACK_KEYS=1;
```

id_user
Identifiant de l'utilisateur.
100 caractères max. **Clé primaire**.

nom
Nom de l'utilisateur.
100 caractères max.

prenom
Prénom de l'utilisateur.
100 caractères max.

email
Email de l'utilisateur.
255 caractères max.

password
Mot de passe de l'utilisateur crypté par hachage SHA1.

langue
Langue par défaut de l'utilisateur (code ISO 3166-1 alpha-2).
2 caractères max. Par défaut -> '**fr**'.

profil
Profil de l'utilisateur.
Entier non signé (0..255). **Clé étrangère** (table `_profils`) Par défaut -> **0**.

testeur
Non utilisé actuellement. Ce champ est prévu (mais non programmé) dans le cas où un utilisateur devrait avoir un droit particulier pour tester l'application.
Booléen (1 = testeur ; 0 = non testeur). Par défaut -> **0**.

date_creation
Date de création de l'utilisateur.
Datetime. Par défaut -> **NULL**.

dernier_acces
Date du dernier accès de l'utilisateur à l'application.
Datetime. Par défaut -> **NULL**.

action_demandee
Action demandée par l'utilisateur (comme par exemple la modification de son mot de passe).
255 caractères max. Par défaut -> **0**.

code_validation
Code de validation permettant d'authentifier une action demandée par un utilisateur. Par exemple, dans le cas d'un changement de mot de passe, l'application peut créer un code de validation qui pourrait être transmise par email.
255 caractères max.

active
Utilisateur actif ? On s'en sert pour activer ou désactiver un utilisateur. Permet par exemple de le bannir sans le retirer.
Booléen (1 = utilisateur actif ; 0 = utilisateur non actif). Par défaut -> **0**.

autolog
Champ propriété permettant de loguer automatiquement un utilisateur lors de sa prochaine visite.
Booléen (1 = l'utilisateur sera automatiquement logué lors de sa prochaine visite ; 0 = pas de connexion automatique). Par défaut -> **0**.

ip
Dernière adresse IP connue de l'utilisateur (IP V4 ou V6).
39 caractères max.

notes_privees
Saisie libre concernant l'utilisateur ou le client (par exemple à destination des administrateurs.)
Texte.

2.9 _params

Cette table contient tous les paramètres de l'application.

```
CREATE TABLE `uw_params` (
  `id` int(10) UNSIGNED NOT NULL,
  `ordre` tinyint(3) UNSIGNED NOT NULL DEFAULT '0',
  `parametre` varchar(32) NOT NULL,
  `valeur` varchar(255) NOT NULL,
  `reglable` tinyint(1) UNSIGNED NOT NULL DEFAULT '0',
  `type` varchar(10) NOT NULL DEFAULT 'text',
  `libelle` varchar(64) NOT NULL,
  `min` varchar(32) DEFAULT NULL,
  `max` varchar(32) DEFAULT NULL,
  `step` varchar(32) DEFAULT NULL,
  `comment` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

id
Identifiant du paramètre.
*Entier non signé (0..*4294967295*30)*. **Clé primaire, auto-incrément**.

ordre
Information de l'ordre d'affichage des paramètres.
*Entier non signé (0..*255*)*
Par défaut -> **0**.

parametre
Nom du paramètre qui sera directement utilisable dans l'application comme constante.
32 caractères max.

valeur
Valeur du paramètre.
255 caractères max.

reglable
Si true, alors le paramètre peut-être affiché dans un formulaire de réglage du site destiné (la plupart du temps) au webmaster.
Booléen (1 = paramètre réglable par utilisateur ; 0 = paramètre non réglable). Par défaut -> **0**.

type
Détermine le type du paramètre
10 caractères max.
text : le type du paramètre est une chaine de caractères.
number : le type du paramètre est un nombre.
boolean : le type du paramètre est un booléen.
date : le type du paramètre est une date.
datetime : le type du paramètre est une date + heure.
Par défaut -> **text**.

type
Chaine de caractère à afficher sur le formulaire de réglage, si le paramètre est réglable.
64 caractères max.

min
Valeur minimale du paramètre si il est du type *number*. Cela permet d'ajouter des tests de limite sur le formulaire de réglages.
32 caractères max.

max
Valeur minimale du paramètre si il est du type *number*. Cela permet d'ajouter des tests de limite sur le formulaire de réglages.
32 caractères max.

step
Pas de modification de la valeur du paramètre si il est du type *number*. Cela permet d'ajouter de guider l'utilisateur sur le formulaire de réglages.
32 caractères max.

comment
Commentaire sur le paramètre afin d'expliquer à quoi il sert.
255 caractères max.

Un seul paramètre est prédéfini, `_PARAM_EMAIL_WEBMASTER_`, contenant l'email du webmaster.

```
INSERT INTO `uw_params` (`id`, `ordre`, `parametre`, `valeur`, `reglable`, `type`, `libelle`, `min`, `max`, `step`, `comment`) VALUES
(1, 1, '_PARAM_EMAIL_WEBMASTER_', 'webmaster@application.com', 1, 'text', 'WEBMASTER_EMAIL', NULL, NULL, NULL, 'WEBMASTER_EMAIL_HELP');
```