**MCD**

**ModifierContenu**

Date

**UserAdmin**

**- Id\_user**

- User

- Password

0-1

**ModifierProfil**

1-1

**Profil**

**- Id\_profil**

- Nom

- Prenom

- Adresse

- Adresse Mail

0-N

0-N

0-N

0-N

**Container**

**- Id\_container**

- Nom

- Désignation

1-N

**Contenir**

0-N

**Page WEB**

**- Id\_web**

- Adresse URL

N-N

**Contenu**

**- Id\_contenu**

- Description

- Adresse\_image

- Contenu\_Texte

**Associer**

N-N

N-N

**Message**

**- Id\_message**

- Nom

- Prenom

- Adresse

- Adresse Mail

- Telephone

- Texte

- Statut

**Traiter** *Contrainte**(Lire plusieurs fois,* supprimer une fois)

Date

N-N

0-N

0-N

1-1

1-1

**Action**

**- Id\_action**

- Type (Lire, supprimer)

**MLD**

Page\_WEB (Id\_web , Adresse\_URL )

Contenir (id(Id\_web, Id\_contenu), Id\_web#, *Id\_contenu#*)

Container (Id\_container, Nom, Désignation )

Associer (id (Id\_contenu, Id\_container), Id\_contenu# , Id\_container#)

Contenu (Id\_contenu, Description, Adresse\_image, Contenu\_Texte )

ModifierContenu(id(Id\_user, Id\_contenu), Id\_user#, Id\_contenu#, Date)

UserAdmin (Id\_user, User, Password, Nom, Prenom, Adresse, Adresse\_Mail )

Traiter (id(id\_user, id\_message, id\_action ), id\_user, id\_message, id\_action#)

Message (Id\_message, Nom, Prenom, Adresse, Adresse\_Mail, Telephone, Texte )

Action(Id\_action, Type)

**Script SQL**

-- Création des entités principales

CREATE TABLE Page\_WEB (

Id\_web INT PRIMARY KEY,

Adresse\_URL VARCHAR(255) NOT NULL

);

CREATE TABLE Container (

Id\_container INT PRIMARY KEY,

Nom VARCHAR(255) NOT NULL,

Désignation VARCHAR(255)

);

CREATE TABLE Contenu (

Id\_contenu INT PRIMARY KEY,

Description TEXT,

Adresse\_image VARCHAR(255),

Contenu\_Texte TEXT

);

CREATE TABLE Contenir (

Id\_web INT,

Id\_contenu INT,

PRIMARY KEY (Id\_web, Id\_contenu),

FOREIGN KEY (Id\_web) REFERENCES Page\_WEB(Id\_web),

FOREIGN KEY (Id\_contenu) REFERENCES Contenu(Id\_contenu)

);

CREATE TABLE Associer (

Id\_contenu INT,

Id\_container INT,

PRIMARY KEY (Id\_contenu, Id\_container),

FOREIGN KEY (Id\_contenu) REFERENCES Contenu(Id\_contenu),

FOREIGN KEY (Id\_container) REFERENCES Container(Id\_container)

);

CREATE TABLE Profil (

Id\_profil INT PRIMARY KEY,

Nom VARCHAR(255) NOT NULL,

Prenom VARCHAR(255) NOT NULL,

Adresse VARCHAR(255),

Adresse\_Mail VARCHAR(255)

);

CREATE TABLE UserAdmin (

Id\_user INT PRIMARY KEY,

User VARCHAR(255) NOT NULL,

Password VARCHAR(255) NOT NULL,

Nom VARCHAR(255),

Prenom VARCHAR(255),

Adresse VARCHAR(255),

Adresse\_Mail VARCHAR(255)

);

CREATE TABLE ModifierContenu (

Id\_user INT,

Id\_contenu INT,

Date DATETIME,

PRIMARY KEY (Id\_user, Id\_contenu),

FOREIGN KEY (Id\_user) REFERENCES UserAdmin(Id\_user),

FOREIGN KEY (Id\_contenu) REFERENCES Contenu(Id\_contenu)

);

CREATE TABLE ModifierProfil (

Id\_user INT PRIMARY KEY,

Id\_profil INT UNIQUE,

FOREIGN KEY (Id\_user) REFERENCES UserAdmin(Id\_user),

FOREIGN KEY (Id\_profil) REFERENCES Profil(Id\_profil)

);

CREATE TABLE Action (

Id\_action INT PRIMARY KEY,

Type VARCHAR(20) NOT NULL CHECK (Type IN ('Lire', 'Supprimer'))

);

-- La table Action contient deux types fixes : 'Lire' et 'Supprimer'

CREATE TABLE Message (

Id\_message INT PRIMARY KEY,

Nom VARCHAR(255),

Prenom VARCHAR(255),

Adresse VARCHAR(255),

Adresse\_Mail VARCHAR(255),

Telephone VARCHAR(20),

Texte TEXT,

Statut VARCHAR(20) -- Exemples : 'nouveau', 'lu', 'supprimé'

);

-- Le champ Statut est mis à jour par l'application selon les actions sur le message

CREATE TABLE Traiter (

Id\_user INT,

Id\_message INT,

Id\_action INT,

Date DATETIME,

PRIMARY KEY (Id\_user, Id\_message, Id\_action),

FOREIGN KEY (Id\_user) REFERENCES UserAdmin(Id\_user),

FOREIGN KEY (Id\_message) REFERENCES Message(Id\_message),

FOREIGN KEY (Id\_action) REFERENCES Action(Id\_action)

);

-- La clé primaire garantit qu'un même UserAdmin ne peut effectuer qu'une seule fois

-- une même action sur un message donné

-- (ex: Lire une seule fois, Supprimer une seule fois)

-- Remarque importante :

-- - Plusieurs UserAdmin peuvent lire un même message (plusieurs lignes 'Lire')

-- - Un seul UserAdmin peut supprimer un message

-- - C'est la responsabilité de l'application de :

-- -> changer le Statut à 'supprimé' après une action de type 'Supprimer'

-- -> empêcher d'autres actions après suppression (côté application ou trigger)

**🔥 Remarques importantes**

* **Application** ou **Trigger** devra empêcher toute action de lecture après qu'un message soit **supprimé**.
* Si besoin, je peux aussi t’écrire un **exemple de trigger** en SQL pour empêcher automatiquement qu’on lise un message déjà supprimé (optionnel).

| **Élément** | **Géré par** | **Statut** |
| --- | --- | --- |
| Clé d'unicité sur Traiter | PK (id\_user, id\_message, id\_action) | ✅ |
| Possibilité de lecture multiple par plusieurs UserAdmin | Oui | ✅ |
| Suppression unique d'un message | Oui (action 'supprimer' unique par user+message) | ✅ |
| Évolution du statut du message | Par application ou trigger | ✅ |

Script SQL d'exemples d'insertion

-- Insertion d'Actions

INSERT INTO Action (Id\_action, Type) VALUES

(1, 'Lire'),

(2, 'Supprimer');

-- Insertion de Pages Web

INSERT INTO Page\_WEB (Id\_web, Adresse\_URL) VALUES

(1, 'https://example.com');

-- Insertion de Containers

INSERT INTO Container (Id\_container, Nom, Désignation) VALUES

(1, 'Bannière', 'Haut de page'),

(2, 'Pied de page', 'Bas de page');

-- Insertion de Contenus

INSERT INTO Contenu (Id\_contenu, Description, Adresse\_image, Contenu\_Texte) VALUES

(1, 'Image principale', 'image1.png', 'Bienvenue sur notre site');

-- Association entre Page Web et Contenu

INSERT INTO Contenir (Id\_web, Id\_contenu) VALUES

(1, 1);

-- Association entre Contenu et Container

INSERT INTO Associer (Id\_contenu, Id\_container) VALUES

(1, 1),

(1, 2);

-- Insertion de Profils

INSERT INTO Profil (Id\_profil, Nom, Prenom, Adresse, Adresse\_Mail) VALUES

(1, 'Dupont', 'Jean', '10 rue Paris', 'jean.dupont@email.com');

-- Insertion de UserAdmin

INSERT INTO UserAdmin (Id\_user, User, Password, Nom, Prenom, Adresse, Adresse\_Mail) VALUES

(1, 'admin1', 'password1', 'Martin', 'Paul', '20 avenue Lyon', 'paul.martin@email.com'),

(2, 'admin2', 'password2', 'Durand', 'Sophie', '30 boulevard Nice', 'sophie.durand@email.com');

-- Insertion de Messages

INSERT INTO Message (Id\_message, Nom, Prenom, Adresse, Adresse\_Mail, Telephone, Texte, Statut) VALUES

(1, 'Tremblay', 'Alice', '50 rue Bordeaux', 'alice.tremblay@email.com', '0601020304', 'Demande de renseignements', 'nouveau'),

(2, 'Bernard', 'Luc', '70 rue Marseille', 'luc.bernard@email.com', '0605060708', 'Problème de connexion', 'nouveau');

⚡ Trigger pour bloquer la lecture d'un message supprimé

DELIMITER $$

CREATE TRIGGER prevent\_read\_deleted\_message

BEFORE INSERT ON Traiter

FOR EACH ROW

BEGIN

DECLARE msg\_status VARCHAR(20);

-- On récupère le statut du message

SELECT Statut INTO msg\_status

FROM Message

WHERE Id\_message = NEW.Id\_message;

-- Si l'action est Lire et que le message est déjà Supprimé => Interdire

IF (NEW.Id\_action = (SELECT Id\_action FROM Action WHERE Type = 'Lire')) AND (msg\_status = 'supprimé') THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Impossible de lire un message supprimé.';

END IF;

-- Si l'action est Supprimer et que le message est déjà Supprimé => Interdire

IF (NEW.Id\_action = (SELECT Id\_action FROM Action WHERE Type = 'Supprimer')) AND (msg\_status = 'supprimé') THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Ce message est déjà supprimé.';

END IF;

END$$

DELIMITER ;

Mise a jour 14062025