**BASE DE DONNEES**

**MCD =** modélisation conceptuelle des données

Cahier des Charges L'entreprise ACME souhaite développer une application web pour remplacer le catalogue papier de ses produits par un catalogue numérique. La base de données actuelle est sous la forme d'une simple feuille de tableur. L'application présentera une liste des produits avec, pour chacun d'eux, une photo d'illustration.

-Un objet de la vie réelle est une **ENTITE**

-Dans une entité, il doit y avoir obligatoirement **une clé id** qui identifie de façon unique chaque instance de l’entité.

Chaque produit ACME a les caractéristiques suivantes, ce sont des **ATTRIBUTS** :

● Un nom

● Une référence

● Un prix

**INSTANCE :**

Dans le contexte de la modélisation conceptuelle des données (MCD), une instance se réfère à un exemple spécifique ou à une occurrence réelle d'une entité, d'une association ou d'un attribut défini dans le modèle.

Pour mieux comprendre le concept d'instance, considérons un exemple simplifié de modèle de données pour une application de gestion d'une bibliothèque :

Entités :

- Livre

- Auteur

- Bibliothécaire

Associations :

- Emprunt (entre Livre et Bibliothécaire)

Attributs :

- Titre (dans Livre)

- Nom (dans Auteur)

- Nom (dans Bibliothécaire)

Dans ce modèle, une instance d'un Livre pourrait être "Harry Potter et la Chambre des Secrets" avec un Titre "Harry Potter et la Chambre des Secrets".

Une instance d'un Auteur pourrait être "J.K. Rowling" avec un Nom "Rowling".

Une instance d'un Bibliothécaire pourrait être "John Smith" avec un Nom "Smith".

En ce qui concerne l'association Emprunt, une instance spécifique pourrait être un emprunt particulier où le Livre "Harry Potter et la Chambre des Secrets" a été emprunté par le Bibliothécaire "John Smith".

Les instances représentent les données réelles et spécifiques qui existent dans le domaine d'application. Elles sont des occurrences concrètes des entités, des associations et des attributs définis dans le modèle de données.

Lorsque vous concevez un MCD, vous définissez les entités, les associations et les attributs qui représentent les types de données abstraits. Ensuite, lors de l'utilisation de l'application réelle, vous créez des instances de ces entités, associations et attributs, en leur attribuant des valeurs spécifiques qui reflètent les données réelles dans le système.

Les instances sont utilisées pour représenter les données concrètes et sont manipulées lors de l'interaction avec le système de gestion de bases de données ou lors de la mise en œuvre de la logique métier de l'application.

**MYSQL**

**DIAGRAM DE CLASSES**

database-> reverse engineer database->add diagram

**CARDINALITES(Modelisation)**

Pour voir les cardinalités ds le diagramme, cliquer = Model->relationship notation-> UML

**ENTITES (cardinalités)**

Indication du nombre min et max de liens entre 2 entités

Pour une association de 2 entités, il y a donc 4 cardinalités à indiquer(ex : 1.1---------0.n)

N = plusieurs , 0 et 1

One To One (1, 1)

Many To One (1, n)

Many To Many (n, m(remplace le n pour plus de visibilité))

Lorsque la table est créée et pour enlever l’index qui est à l’intérieur de la table = Model->Objet notation->workbench (simplyfied)

VALIDATION DATABASE DIAGRAM

-Pour diriger workbench et mysql = Database->forward Engineer->se connecter au serveur mysql

-Pour synchroniser le diagramme et voir si modification à faire entre les tables, = Database->Synchronyze Model

-----------------------------------------------------------------------------------------------------------------------------------

Quand on crée un mot de passe dans une table, il faut cocher le B (binary) pour qu’il ne confonde pas les majuscule et minuscule.

SUPPRIMER AJOUTER MODIFIER

il est généralement recommandé d'inclure des horodatages pour suivre quand un objet a été ajouté, modifié ou supprimé dans une base de données. Ces horodatages sont utiles pour l'audit, la traçabilité et la compréhension de l'historique des données. Voici comment vous pourriez les inclure :

1. **Date d'ajout (created\_at) :** Cette horodatage indique quand un objet a été initialement ajouté à la base de données. Il est souvent défini au moment de l'insertion initiale de l'enregistrement.
2. **Date de modification (updated\_at) :** Cette horodatage est mis à jour chaque fois que l'objet est modifié. Cela inclut tout changement de données associé à cet enregistrement.
3. **Date de suppression (deleted\_at) :** Si vous implémentez la suppression logique (soft delete), c'est-à-dire que vous ne supprimez pas vraiment l'enregistrement de la base de données, mais le marquez comme supprimé, cette horodatage indique quand l'enregistrement a été marqué comme supprimé.
4. Lorsque vous insérez un nouvel enregistrement dans la table, la valeur "created\_at" est définie sur la date et l'heure actuelles. Par exemple, si vous insérez un enregistrement le 4 août 2023 à 14 h 30 min, la colonne "created\_at" de cet enregistrement serait automatiquement remplie avec "2023-08-04 14:30:00".
5. Dans la plupart des systèmes de gestion de bases de données, vous n'avez pas besoin de vous soucier de la valeur à mettre dans la colonne "created\_at". Cela sera géré en interne par la base de données lors de l'insertion des données.

VALEUR :

1. **TIMESTAMP (ou DATETIME) :** Ce type de données stocke à la fois la date et l'heure jusqu'à la fraction de seconde. Il est couramment utilisé pour les horodatages dans de nombreux systèmes de bases de données, y compris MySQL, PostgreSQL, Microsoft SQL Server, etc.
2. **DATE :** Ce type de données stocke uniquement la date (jour, mois, année) sans l'heure. Il est utile si vous n'avez pas besoin de suivre l'heure exacte de création.
3. Haut du formulaire

--------------------------------------------------------------------------------------------------------------------------------------Pour stocker les messages que vous recevez dans une base de données, vous pouvez créer une table dédiée aux messages. Voici comment vous pourriez structurer cette table :

```plaintext

Table : Messages

-------------------------------------------------------------------------------------

| id | expediteur | destinataire | contenu | date\_reception | lu |

-------------------------------------------------------------------------------------

| 1 | Alice | Vous | Bonjour ! | 2023-08-01 12:00:00 | true |

| 2 | Bob | Vous | Besoin d'aide | 2023-08-01 13:30:00 | false |

-------------------------------------------------------------------------------------

TABLE FORMULAIRE DE CONTACT

Dans cet exemple :

- La colonne "expediteur" indique qui a envoyé le message.

- La colonne "destinataire" indique à qui le message est destiné (dans ce cas, "Vous" indique que le message est adressé à vous).

- La colonne "contenu" stocke le contenu textuel du message.

- La colonne "date\_reception" enregistre la date et l'heure de réception du message.

- La colonne "lu" indique si le message a été lu. "true" indique qu'il a été lu, "false" indique qu'il n'a pas encore été lu.

Lorsque vous recevez un nouveau message, vous insérez un nouvel enregistrement dans cette table avec les informations pertinentes. Si vous lisez un message, vous mettez à jour la colonne "lu" en le passant à "true".

La conception exacte de votre table de messages peut varier en fonction des besoins spécifiques de votre application, mais cette structure de base devrait vous donner une idée de la manière dont vous pouvez stocker les messages que vous recevez.

--------------------------------------------------------------------------------------------------------------------------------------SIGNIFICATION DES CASE A COCHER LORS DE CREATION DES TABLES

UN

l'option "UNSIGNED" (ou "UNSIGNÉ" en français) lors de la création d'une colonne dans une table d'une base de données relationnelle.

L'option "UNSIGNED" est utilisée généralement dans le contexte des types de données numériques (comme les entiers) pour indiquer que la colonne ne peut pas contenir de valeurs négatives. Cela permet de garantir que la colonne n'accepte que des valeurs positives ou nulles.

Par exemple, si vous avez une colonne qui stocke des âges, vous pourriez vouloir utiliser un type de données entier non signé ("UNSIGNED INT" ou "UNSIGNED INTEGER" en anglais) pour vous assurer que seules les valeurs positives sont acceptées.

Voici comment vous pourriez l'utiliser lors de la création d'une table (en supposant que vous utilisez MySQL) :

```sql

CREATE TABLE Utilisateurs (

id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

nom VARCHAR(50),

age INT UNSIGNED

);

```

Dans cet exemple, la colonne "age" est définie comme "INT UNSIGNED" pour indiquer que seules les valeurs positives sont acceptées.

Cependant, l'option "UNSIGNED" dépend du système de gestion de bases de données que vous utilisez et de la manière dont il traite les types de données numériques. Assurez-vous de consulter la documentation de votre système de gestion de bases de données pour comprendre comment utiliser correctement cette option lors de la création de vos tables.

NOT NULL

L'expression "NOT NULL" est utilisée lors de la définition d'une colonne dans une table d'une base de données pour indiquer que cette colonne ne peut pas contenir de valeurs nulles. En d'autres termes, chaque enregistrement dans la table doit avoir une valeur valide pour cette colonne, et elle ne peut pas être laissée vide.

VALEUR BOOLEEN

Dans MySQL, le type `TINYINT` est souvent utilisé pour représenter des valeurs booléennes, où 0 représente "faux" (non) et 1 représente "vrai" (oui). MySQL n'a pas de type de données booléen distinct, il utilise généralement `TINYINT(1)` pour stocker des valeurs booléennes.

Lorsque vous utilisez `TINYINT(1)` pour stocker des valeurs booléennes, cela permet d'économiser de l'espace de stockage par rapport à d'autres types de données numériques plus grands.

Par exemple, si vous souhaitez ajouter une colonne "est\_admin" pour indiquer si un utilisateur est administrateur ou non, vous pourriez faire quelque chose comme ceci :

```sql

CREATE TABLE Utilisateurs (

id INT PRIMARY KEY,

nom VARCHAR(50),

est\_admin TINYINT(1)

);

```

Lorsque vous insérez des données dans cette colonne, vous pouvez utiliser 0 pour "faux" et 1 pour "vrai" :

```sql

INSERT INTO Utilisateurs (id, nom, est\_admin) VALUES (1, 'Alice', 1); -- Alice est administratrice

INSERT INTO Utilisateurs (id, nom, est\_admin) VALUES (2, 'Bob', 0); -- Bob n'est pas administrateur

```

Cependant, si vous préférez utiliser des étiquettes plus explicites comme "true" et "false", vous pouvez toujours utiliser le type de données `BOOLEAN` à la place, mais notez que ce type n'est pas toujours pris en charge dans toutes les versions de MySQL.

```sql

CREATE TABLE Utilisateurs (

id INT PRIMARY KEY,

nom VARCHAR(50),

est\_admin BOOLEAN

);

```

Si votre version de MySQL prend en charge `BOOLEAN`, vous pouvez l'utiliser pour représenter des valeurs booléennes. Sinon, utiliser `TINYINT(1)` pour les valeurs booléennes est une approche courante et fonctionnelle.

Phpmyadmin EXPORTER LES TABLE VERS LE SERVEUR

Aller ds configuration sur xampp pour selectionner my.ini

On va exporter le modèle (tables) vers mysql en physique :

Aller dans database ds le menu => forward engineer to database

-Ne rien cocher si pas de mot de passe et faire next

-Pour la 2 ème page, faire next aussi

-affichage des tables : next encore

Les cardinalités en pointillé et en ligne continue sont des notations utilisées dans les schémas de relations pour indiquer les quantités de relations entre différentes entités (tables) dans une base de données. Elles aident à décrire les relations et les associations entre les données dans un modèle de base de données.

Voici ce que signifient les différentes notations de cardinalité :

1. \*\*Cardinalités en ligne continue (trait plein)\*\* :

Les cardinalités en ligne continue sont utilisées pour indiquer les relations obligatoires entre les entités. Elles montrent que chaque instance (ligne) d'une entité doit être associée à une ou plusieurs instances d'une autre entité, en fonction de la cardinalité exacte. Les cardinalités en ligne continue sont généralement indiquées par des lignes droites sans interruption.

Par exemple, si vous avez une relation "un étudiant appartient à une classe", la cardinalité en ligne continue indiquerait que chaque étudiant (entité) doit être associé à une classe (entité).

2. \*\*Cardinalités en pointillé (trait en pointillé)\*\* :

Les cardinalités en pointillé sont utilisées pour indiquer les relations optionnelles entre les entités. Elles montrent que chaque instance d'une entité peut être associée à une ou plusieurs instances d'une autre entité, mais cela n'est pas obligatoire. Les cardinalités en pointillé sont généralement indiquées par des lignes en pointillé.

Par exemple, si vous avez une relation "un client peut avoir plusieurs commandes", la cardinalité en pointillé indiquerait que chaque client (entité) peut être associé à plusieurs commandes (entité), mais il peut également ne pas avoir de commandes du tout.

En résumé, les cardinalités en ligne continue indiquent des relations obligatoires, tandis que les cardinalités en pointillé indiquent des relations optionnelles. Cela permet de spécifier comment les données sont liées entre elles dans un modèle de base de données.