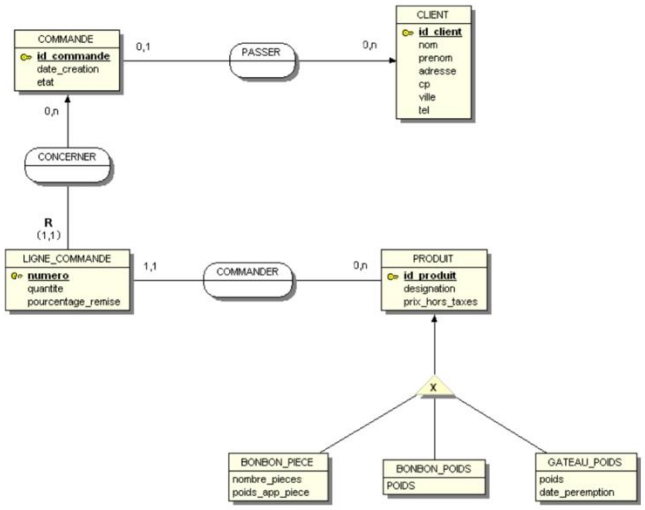
Documentation du TP Sucrerie

MCD du TP :



Objectif du Script
Le script définit une base de données appelée **Sucrerie** pour gérer une boutique de bonbon et de pâtisseries. Il permet de stocker des informations sur les clients, les commandes, les produits, ainsi que des règles spécifiques pour différencier les types de produits.

Structure de la Base de Données
1. Base de données

- **Sucrerie** : Contient les tables pour gérer les clients, les commandes et les produits.
- **Création de la BD : create database Sucrerie ; sur mysql**

```
create database Sucrerie ;
```

2. Tables

a) Client

- Gère les informations des clients.
- **Colonnes :**
  - id_client : Identifiant unique du client (clé primaire).
  - nom : Nom du client.
  - prenom : Prénom du client.
  - adresse : Adresse du client.
  - cp : Code postal.
  - ville : Ville du client.
  - tel : Numéro de téléphone.
- **Contraintes :**
  - pk_client : Définit id_client comme clé primaire.

```sql
create table Client(
    id_client int ,
    nom varchar(250) ,
    prenom varchar(250) ,
    adresse varchar(250) ,
    cp int ,
    ville varchar(250) ,
    tel int,
    constraint pk_client primary key (id_client)
)engine = innodb ;
```

b) Commande

- Gère les commandes effectuées par les clients.
- **Colonnes :**
  - id_commande : Identifiant unique de la commande (clé primaire).
  - date_creation : Date de création de la commande.
  - etat : État de la commande (exemple : 0 pour en cours, 1 pour terminée).
  - id_client : Identifiant du client qui a passé la commande.
- **Contraintes :**
  - pk_commande : Définit id_commande comme clé primaire.
  - fk_commande_client : Clé étrangère reliant id_client à la table Client.

```sql
create table commande(
    id_commande int ,
    date_creation date,
    etat int ,
    id_client int ,
    constraint pk_commande primary key (id_commande),
    constraint fk_commande_client FOREIGN KEY (id_client) REFERENCES Client(id_client)
);
```

c) Ligne_Commande

- Détaille les produits dans une commande.
- **Colonnes :**
  - numero : Numéro de la ligne (clé partielle).
  - id_commande : Identifiant de la commande (clé étrangère).
  - id_produit : Identifiant du produit (clé étrangère).
  - quantite : Quantité commandée.
  - pourcentage_remise : Remise appliquée au produit.
- **Contraintes :**
  - pk_ligne_commande : Définit une clé primaire composée (numero, id_commande) car identifiant relatif.
  - fk_id_commande_ligne : Clé étrangère reliant id_commande à la table Commande.
  - fk_ligne_produit : Clé étrangère reliant id_produit à la table Produit.

```
create table ligne_commande (
    numero int ,
    id_commande int ,
    id_produit int ,
    quantite int,
    pourcentage_remise  float,
    constraint pk_ligne_commande primary key(numero , id_commande) ,
    constraint fk_id_commande_ligne foreign key (id_commande) references commande(id_commande),
    constraint fk_ligne_produit foreign key (id_produit) references produit(id_produit)
);
```

d) Produit

- Stocke les informations générales sur les produits.
- **Colonnes :**
  - id_produit : Identifiant unique du produit (clé primaire).
  - designation : Nom ou description du produit.
  - prix_hors_taxe : Prix HT du produit.
- **Contraintes :**
  - pk_produit : Définit id_produit comme clé primaire.

```
create table produit (
    id_produit int ,
    designation varchar(250) ,
    prix_hors_taxe float,
    constraint pk_produit primary key (id_produit)
);
```

e) Bonbon_Piece

- Détaille les bonbons vendus par pièce.
- **Colonnes :**
  - id_produit : Identifiant du produit (clé étrangère).
  - nombre_pieces : Nombre de pièces par unité.
  - poids_app : Poids approximatif d'une unité.
- **Contraintes :**
  - pk_bonbon_piece : Définit id_produit comme clé primaire.
  - fk_bonbon_produit : Clé étrangère reliant id_produit à la table Produit.

f) Bonbon_Poids

- Détaille les bonbons vendus au poids.

- **Colonnes :**
  - id_produit : Identifiant du produit (clé étrangère).
  - poids : Poids total du produit.
- **Contraintes :**
  - pk_bonbon_piece : Définit id_produit comme clé primaire.
  - fk_poids_produit : Clé étrangère reliant id_produit à la table Produit.

g) Gateau_Poids

- Détaille les gâteaux vendus au poids.
- **Colonnes :**
  - id_produit : Identifiant du produit (clé étrangère).
  - poids : Poids du gâteau.
  - date_peremption : Date de péremption.
- **Contraintes :**
  - pk_bonbon_piece : Définit id_produit comme clé primaire.
  - fk_gateau_produit : Clé étrangère reliant id_produit à la table Produit.

```sql
create table bonbon_piece (
    id_produit int ,
    nombre_pieces int ,
    poids_app float ,
    constraint pk_bonbon_piece primary key (id_produit) ,
    constraint fk_bonbon_produit foreign key (id_produit) references produit(id_produit)

);

create table bonbon_poids (
    id_produit int ,
    poids float,
    constraint pk_bonbon_piece primary key (id_produit) ,
    constraint fk_poids_produit foreign key (id_produit) references produit(id_produit)

);

create table gateau_poids (
    id_produit int ,
    poids float ,
    date_peremption date ,
    constraint pk_bonbon_piece primary key (id_produit) ,
    constraint fk_gateau_produit foreign key (id_produit) references produit(id_produit)

);
```

(j'ai décidé de choisir l'option de créer 3 pour l'héritage avec seulement la clé primaire de l'entité mere.)

3. Triggers gérant la contrainte de partition

Les triggers garantissent qu'un produit ne peut pas appartenir à plusieurs catégories (bonbon_piece, bonbon_poids, gateau_poids).

a) Trigger partitionBonbonpiece

- **Événement :** Avant insertion dans bonbon_piece.
- **Condition :** Vérifie que id_produit n'existe pas déjà dans bonbon_poids ou gateau_poids.
- **Action :** Bloque l'insertion si la condition n'est pas respectée, avec un message d'erreur "Impossible".

b) Trigger partitiongateau

- **Événement :** Avant insertion dans gateau_poids.
- **Condition :** Vérifie que id_produit n'existe pas déjà dans bonbon_piece ou bonbon_poids.
- **Action :** Bloque l'insertion si la condition n'est pas respectée, avec un message d'erreur "Impossible".

c) Trigger partitionpoids

- **Événement :** Avant insertion dans bonbon_poids.
- **Condition :** Vérifie que id_produit n'existe pas déjà dans bonbon_piece ou gateau_poids.
- **Action :** Bloque l'insertion si la condition n'est pas respectée, avec un message d'erreur "Impossible".

```
delimiter //
create trigger partitionBonbonpiece  before insert on bonbon_piece
for each row
begin
if ((select count(*) from bonbon_poids where id_produit = new.id_produit) > 0 OR
(select count(*) from gateau_poids where id_produit = new.id_produit) > 0) then
SIGNAL SQLSTATE  '45000'
    SET MESSAGE_TEXT = "Impossible";
end if;
end //
delimiter;

delimiter //
create trigger partitiongateau  before insert on gateau_poids
for each row
begin
if ((select count(*) from bonbon_poids where id_produit = new.id_produit) > 0 OR
(select count(*) from bonbon_piece where id_produit = new.id_produit) > 0) then
SIGNAL SQLSTATE  '45000'
    SET MESSAGE_TEXT = "Impossible";
end if;
end //
delimiter ;


delimiter //
create trigger partitionpoids  before insert on bonbon_poids
for each row
begin
if ((select count(*) from bonbon_piece where id_produit = new.id_produit) > 0 OR
(select count(*) from gateau_poids where id_produit = new.id_produit) > 0) then
SIGNAL SQLSTATE  '45000'
    SET MESSAGE_TEXT = "Impossible";
end if;
end //
delimiter ;
```