UNIVERSITÉ CÔTE D'AZUR

MIAGE - LICENCE 3

TP d'approfondissement complément BD (MCD/MLD Merise, SQL et PLSQL).

Membre du groupe: NAÏM CHTIOUI, ABDELRAZAK EL GHAZZAZ, VALENTIN BORDY

Base de Données - **Monsieur Mopolo.**

Partiel 2020.

1. La description du sujet

Le système que nous avons décidé de réaliser pour ce projet est une base de données pour le drive d'un supermarché. Cette base de données contiendra tous les produits disponible du drive supermarché, la liste des clients ainsi que la liste des employés avec celle des fournisseurs. Grâce à ce système, nous pourrons récupérer par exemple à l'aide de requêtes, les informations liées à chaque produit ou encore les différentes commandes faite par le client.

Notre base de données sera structurée d'une dizaine de tables, représentant chacune d'elle une caractéristique de notre drive de supermarché.

La table "produit" contiendra tous les tuples représentant chaque produit en spécifiant son id unique en tant que clé primaire, son nom, son poids net, sa description avec ses différents prix (HT et TTC) ainsi que ses stocks.

Un produit est forcément lié à une catégorie qui se retrouve elle-même dans un rayon. Par conséquent, nous avons les tables "CategorieProduit" et "Rayon" possédant comme attributs le nom et la description.

Notre drive possède évidemment des employés, d'où la création de notre table "Employe" ayant pour attributs l'id unique d'un employé en clé primaire, le nom, le prénom, le mail, le téléphone, le salaire, le genre et sa date de naissance.

Nos produits sont issus de différents fournisseurs qui seront renseignés dans la table "Fournisseur" avec comme informations le nom du fournisseur, le mail, le téléphone ainsi que sa description.

Chaque client qui voudra faire une commande à travers le drive de notre supermarché devra s'inscrire. Les données de son inscription seront contenues dans la table "Client" spécifiant les informations personnelles de chaque client tels que son nom, son prénom, son mail, son numéro de téléphone, son mot de passe, son genre et sa date de naissance. Un client sera naturellement caractérisé par sa clé primaire unique.

Pour pouvoir acheter un produit, le client passe une commande par laquelle nous la caractérisons par la table "Commande" contenant la date de la commande ainsi que son statut, si celle-ci est par exemple en "cours de livraison" ou "livré". Cette même table "Commande" est lié à la table "Produit" par une association porteuse de données se prénommant "LigneCommande" qui précise la quantité et le prix de chaque produit de la commande faite par le client.

En s'inscrivant sur notre plateforme, le client possède automatiquement un système de points de fidélité qui sera précisé dans table "CarteFidelite".

Afin de ne pas surcharger les tables “Client”, “Employe” et “Fournisseur”, nous avons choisi de créer la table “Adresse” séparément qui renseigne les adresses postales, la ville, le code postal ainsi que le pays.

2. La description textuelles des requêtes de mise à jour

Requête impliquant une table:

- **Mise à jour du genre et de la date de naissance d’un client (ici client numéro 1) avec les données ‘Autre’ et ‘12-12-1958’**

```
UPDATE client
SET genre = 'Autre', dateNaissance= '12-12-1958'
WHERE idClient = 1;
```

- **Mise à jour du statut (par exemple en ‘Livrée’) d’une commande (par exemple la commande numéro 2)**

```
UPDATE commande
SET statutCommande = 'Livree'
WHERE idCommande = 2;
```

Requête impliquant deux tables:

- **Mise à jour du taux de TVA à 20% de tous les produits ayant pour une catégorie donnée (par exemple ‘Cremerie’)**

```
UPDATE (SELECT p.* FROM produit p
        INNER JOIN categorieProduit cp ON p.idCategorie = cp.idCategorie
        WHERE cp.nomCategorie = 'Cremerie') t
SET t.tauxTVA = 0.2;
```

- **Mise à jour du stock de produit par rapport aux quantités de produit des lignes de commande**

```
CREATE OR REPLACE VIEW quantite_produit AS
SELECT lc.idProduit, SUM(lc.quantite) as qte FROM ligneCommande lc
GROUP BY lc.idProduit;

UPDATE produit pdt
```

SET pdt.stock = pdt.stock-(SELECT quantite_produit.qte FROM quantite_produit WHERE quantite_produit.idProduit = pdt.idProduit)
WHERE EXISTS (SELECT quantite_produit.qte FROM quantite_produit WHERE quantite_produit.idProduit = pdt.idProduit);

Requête impliquant plus de deux tables:

- **Mise à jour du nombre de points de fidélité des clients pour chaque lignes de commandes dont le prix de vente est supérieur à 10 (par exemple ici ajout de 10 points)**

CREATE OR REPLACE VIEW cf_client AS
SELECT cf.idClient, COUNT(cf.idClient) as nb FROM carteFidelite cf
INNER JOIN commande c ON cf.idClient = c.idClient
INNER JOIN ligneCommande lc ON c.idCommande = lc.idCommande
WHERE lc.prixVente > 10
GROUP BY cf.idClient;

UPDATE carteFidelite cf
SET cf.nbPointsFidelite = cf.nbPointsFidelite + (SELECT cf_client.nb FROM cf_client
*WHERE cf_client.idClient = cf.idClient)*10*
WHERE EXISTS (SELECT cf_client.nb FROM cf_client WHERE cf_client.idClient = cf.idClient);

- **Mise à jour du stock minimum des produits dont le rayon est 'Produit Frais'**

UPDATE (SELECT p. FROM produit p*
INNER JOIN categorieproduit cp ON p.idcategorie = cp.idcategorie
INNER JOIN rayon r ON cp.idrayon = r.idrayon
WHERE r.nomrayon = 'Produit Frais') z
SET z.stockmini = 50;

3. La description textuelles des requêtes de suppression

Requête impliquant une table :

1 - Supprime un produit par rapport à son nom “Chips 3D nature”

DELETE FROM produit p
WHERE p.NOMPRODUIT='Chips 3D nature';

2 - Supprime un produit par rapport à son numéro ID

DELETE FROM produit
WHERE idproduit=12;

Requête impliquant deux tables:

- **Supprime tous les clients habitant à Nice**

DELETE FROM client
WHERE idadresse in (select a.idadresse from adresse a where a.ville='NICE');

- **Supprime un fournisseur fournissant un produit en particulier par rapport à son nom**

DELETE FROM fournisseur
WHERE idfournisseur in (select p.idfournisseur from produit p where p.nomproduit='3x Pile AAA');

Requête impliquant plus de deux tables:

- **Supprime le rayon du magasin contenant le nom d'un produit qui est "Chips 3D paprika"**

DELETE FROM rayon
WHERE idrayon in (select idrayon from categorieproduit cp INNER JOIN produit p ON cp.idcategorie=p.idcategorie where p.nomproduit='Chips 3D paprika');

- **Supprime les commandes contenant des produits qui ont un prix inférieur à 3**

DELETE FROM commande
WHERE idcommande in (select idcommande from lignecommande lc INNER JOIN produit p ON lc.idproduit=p.idproduit where p.prixht < 3);

4. La description textuelles des requêtes de consultation

Requête impliquant une table:

- **Sélectionne le nom du produit et son prix TTC et le tri par ordre croissant. (order by)**

select nomProduit, prixTTC from produit order by(prixTTC) ASC;

- **Sélectionne l'id de la commande, la date de la commande et le statut des commandes qui sont livrées.**

select IDCOMMANDE, DATECOMMANDE, STATUTCOMMANDE from COMMANDE where statutCommande = 'Livree';

- **Sélectionne les informations du client(nom, prenom, telephone, mail, genre et date de Naissance).**

select nom,prenom, telephone, mail, genre, dateNaissance FROM CLIENT;

- **Sélectionne le nombre d'employé en fonction du salaire. (group by)**

select count(idemploye) as nbEmploye , salaire from employe group by salaire;

- **Sélectionne le nombre de rayon que nous avons dans le drive.**

select count(idrayon) as nbRayon from rayon;

Requête impliquant deux tables:

- **Sélectionne toutes les commandes passés(id de la commande et la date), ainsi que l'id du client, le nom et le prénom trié par date décroissante. (order by)**

select client.idclient, client.nom,client.prenom,commande.IDCOMMANDE, commande.DATECOMMANDE from client, commande where CLIENT.IDCLIENT = COMMANDE.IDCLIENT order by dateCommande DESC;

- **Sélectionne le nom et le prénom de tous les employés et leurs nombres de commandes assignés. (outer join)**

Select e.nom,e.prenom, count(c.idcommande) From commande c right outer join Employe e on e.idemploye = c.idemploye group by e.nom, e.prenom;

- **Sélectionne les catégories des produits qui ont un total de produits > 3. (group by)**

Select cp.idCategorie, cp.nomCategorie, count(p.idproduit) as NbProduit from CATEGORIEPRODUIT cp, produit p where p.idcategorie = cp.idcategorie group by cp.idCategorie, cp.nomCategorie having count(p.idproduit) > 3;

- **Sélectionne les noms des catégories et leurs descriptions pour le rayon "Epicerie salee".**

Select nomCategorie, descriptionCategorie from categorieproduit, rayon where rayon.idrayon = categorieproduit.idrayon and rayon.nomRayon='Epicerie salee';

- **Sélectionne le nom, prixHT et la description produits vendus par le fournisseur "BERARD"**

Select p.nomproduit, p.prixHT, p.descriptionProduit, f.nomfournisseur from produit p, fournisseur f where f.idfournisseur = p.idfournisseur and f.nomfournisseur = 'BERARD';

Requête impliquant plus de deux tables:

- **Sélectionne le nom, prénom du client ainsi que le prix total de la commande numéro qui a l'id numéro 1. (group by)**

Select CLIENT.nom, prenom, (Select sum(PrixVente) from LIGNECOMMANDE L group by L.idcommande HAVING IDCOMMANDE='1') as prixCommande from client, COMMANDE WHERE COMMANDE.IDCLIENT = CLIENT.IDCLIENT AND COMMANDE.IDCOMMANDE='1' GROUP BY CLIENT.NOM, CLIENT.PRENOM, prenom;

- **Sélectionne les produits(nom, description, prixHT) et leurs rayons(nom, description) comprenant aussi les produits qui n'ont pas de rayon. (jointure externe)**

SELECT PRODUIT.NOMPRODUIT, PRODUIT.DESCRIPTIONPRODUIT, PRODUIT.PRIXHT, RAYON.NOMRAYON, RAYON.DESCRIPTIONRAYON FROM PRODUIT, CATEGORIEPRODUIT, RAYON
WHERE PRODUIT.IDCATEGORIE = CATEGORIEPRODUIT.IDCATEGORIE(+) AND CATEGORIEPRODUIT.IDRAYON = RAYON.IDRAYON(+);

- **Sélectionne le nombre de commande, l'adresse et le nom et prénom des employes qui se sont occupés de 2 commande ou plus et le trier par le nombre de commande de façon croissante.**

select a.ligneadresse1, a.ligneadresse2, a.ville, a.codepostal, a.pays, e.nom as nomEmploye, e.prenom as PrenomEmploye, count(c.IDCOMMANDE) as NbCommandeTraitee from adresse a, commande c, employe e
where a.idadresse = e.idadresse and c.idemploye = e.idemploye GROUP BY e.nom, e.prenom, a.ligneadresse1, a.ligneadresse2, a.ville,
a.codepostal, a.pays HAVING count(c.IDCOMMANDE) >= 2 ORDER BY COUNT(c.IDCOMMANDE) ASC;

- **Sélectionne le nom et prénom des clients qui ont réalisés une commande de 2 produits ou plus ainsi que leurs nombres de points de fidélité.**

select cl.nom,cl.prenom, cf.nbpointsfidelite from client cl , cartefidelite cf, commande c, lignecommande lc
WHERE cl.idcartefidelite = cf.idcartefidelite AND
cl.idclient = c.idclient AND
c.idcommande = lc.idcommande GROUP BY cl.nom, cl.prenom, cf.nbpointsfidelite HAVING COUNT(lc.idcommande) >=2;

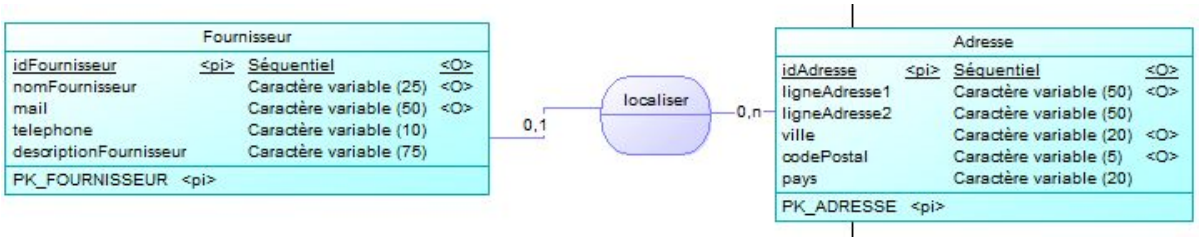
- **Sélectionne les noms, la quantité acheté ainsi que le prix de vente des produits de la commande numéro 1.**

Select p.NOMPRODUIT, p.DESCRIPTIONPRODUIT, lc.quantite, lc.prixvente FROM PRODUIT p , COMMANDE c , LIGNECOMMANDE lc
WHERE p.IDPRODUIT = lc.idproduit AND lc.idcommande = c.idcommande AND c.idcommande = 1;

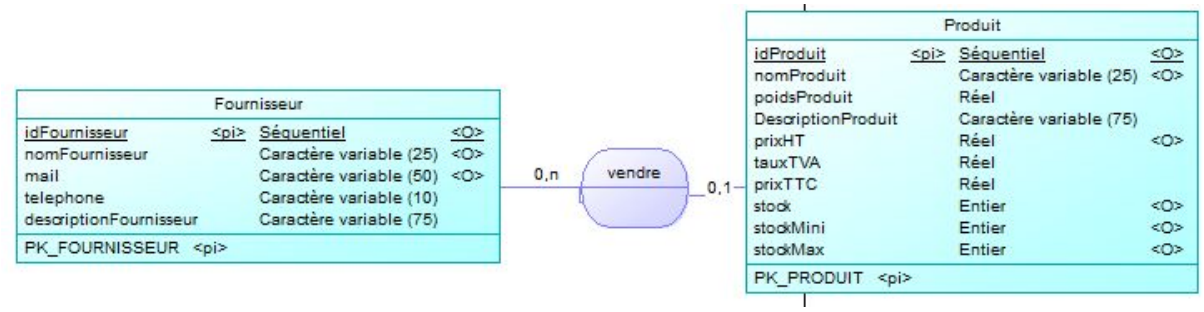
5. Le dictionnaire de données MERISE

(voir fichier joint → Dictionnaire_de_données_BDD_Supermarche.pdf)

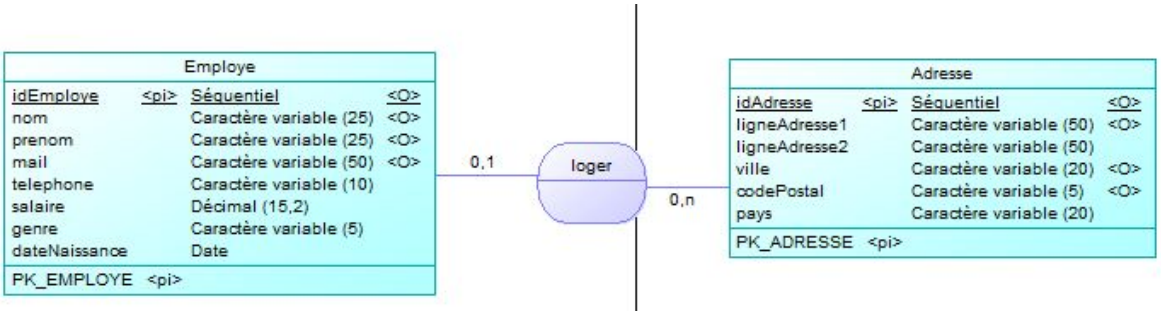
6. La description textuelles des associations



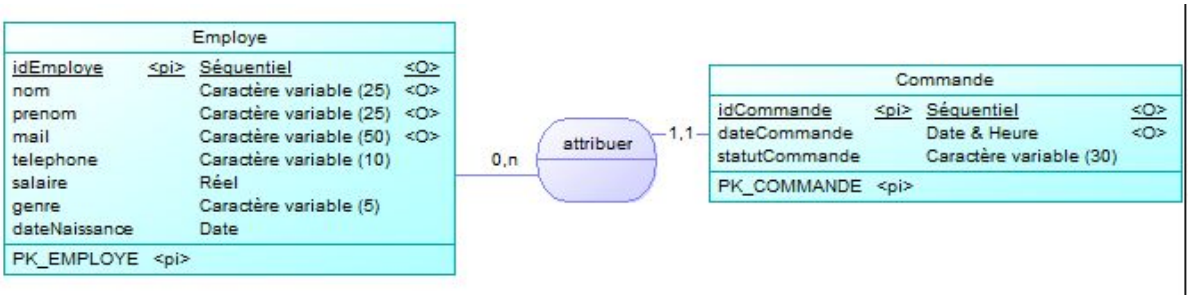
Le fournisseur est localisé à **zéro ou une** adresse mais une adresse peut localiser **zéro ou plusieurs** fournisseurs.



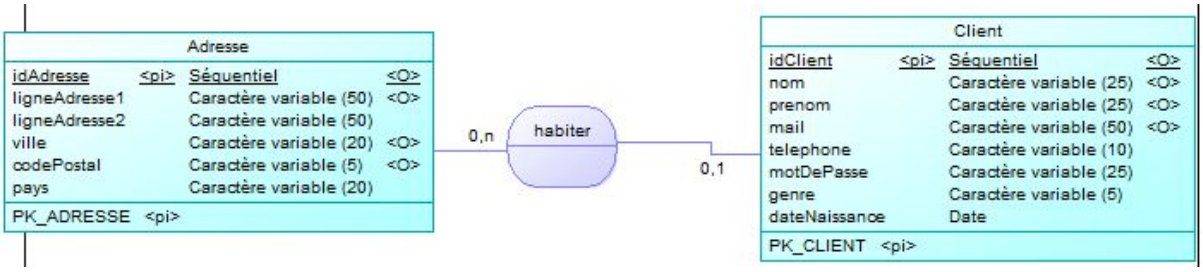
Le fournisseur vend **zéro ou plusieurs** produits mais un produit est vendu par **zéro ou un** fournisseur.



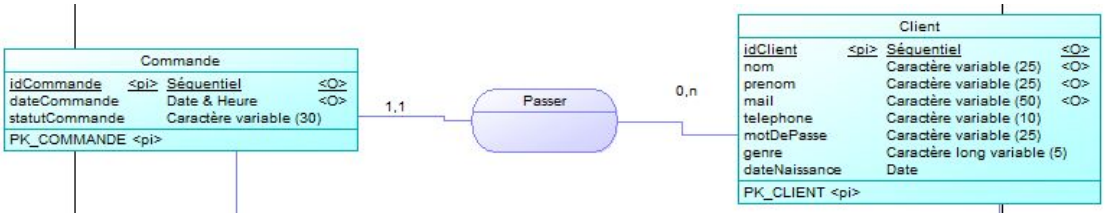
L'employé loge à **zéro ou une** adresse mais l'adresse peut loger **zéro ou plusieurs** employés.



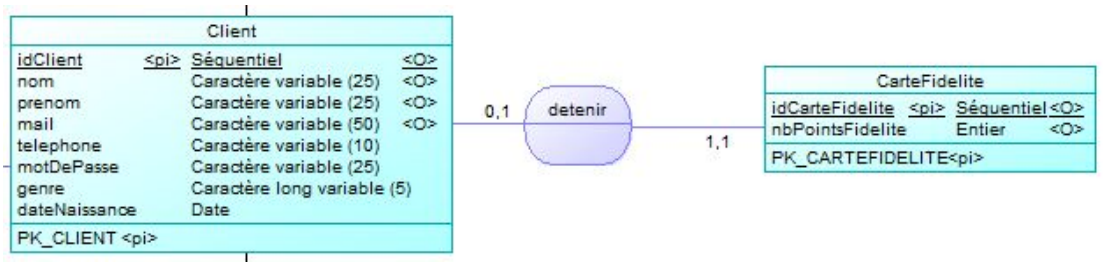
Une Commande est attribuée à **un et un seul** employé mais l'employé peut avoir **zéro ou plusieurs** commandes.



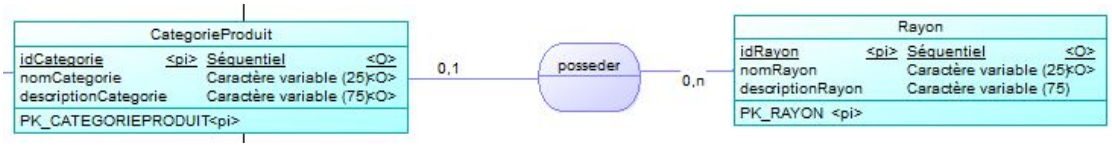
Un Client habite à **zéro ou une** adresse mais une adresse peut habiter **zéro ou plusieurs** clients.



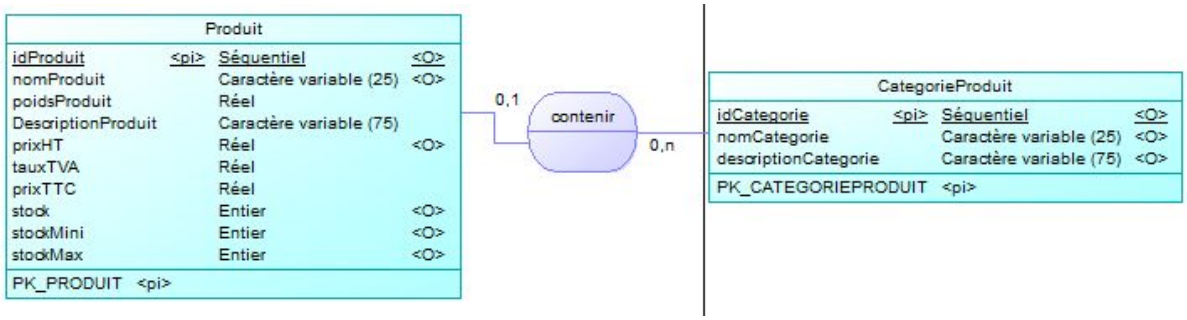
Un client peut passer **zéro ou plusieurs** commandes. Et une commande est passé par **un et un** seul client.



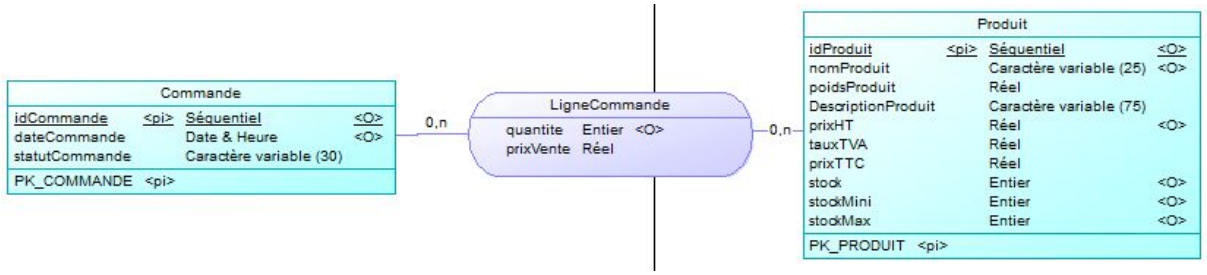
Un client détient **zéro ou une** carte de fidélité mais une carte de fidélité est détenu par **un et un** seul client.



Un rayon possède **zéro ou plusieurs** catégories de produit mais une catégorie de Produit est possédée par **zéro ou un** seul rayon.

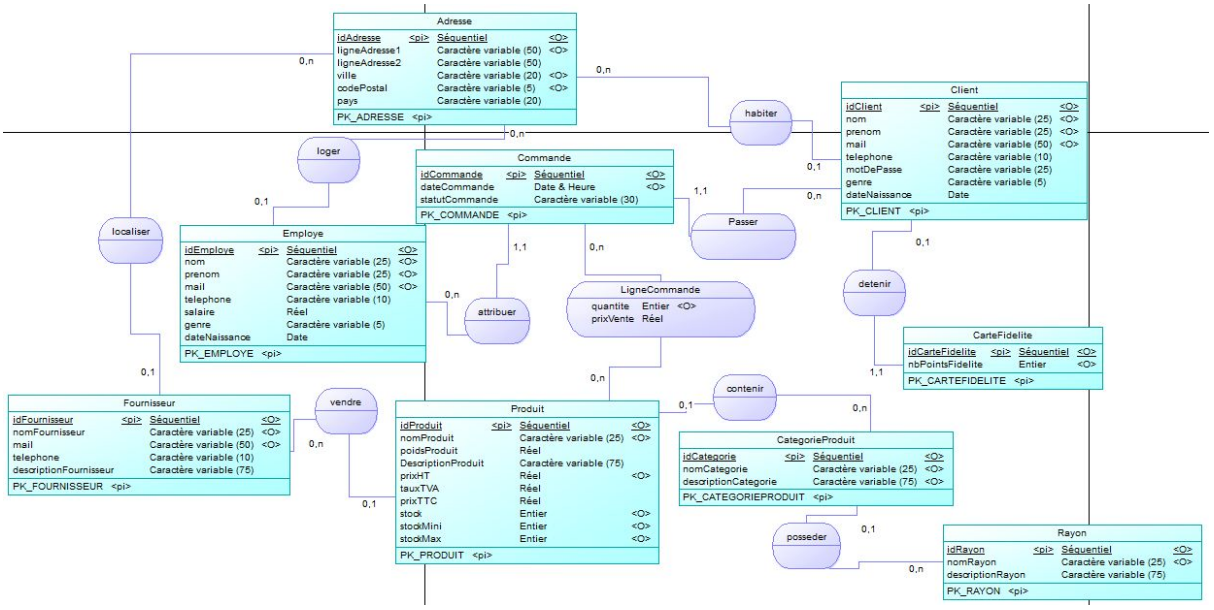


Le produit contient **zéro ou une** catégorie de produit et la catégorie de produit contient **zéro ou plusieurs** produits.



La commande possède **zéro ou plusieurs** produits ainsi que les produits peuvent apparaître dans plusieurs commandes. Pour se faire nous avons créé une association porteuse de données qui possédera la quantité et le prix de vente.

7. La définition du Modèle Entité-Association MERISE



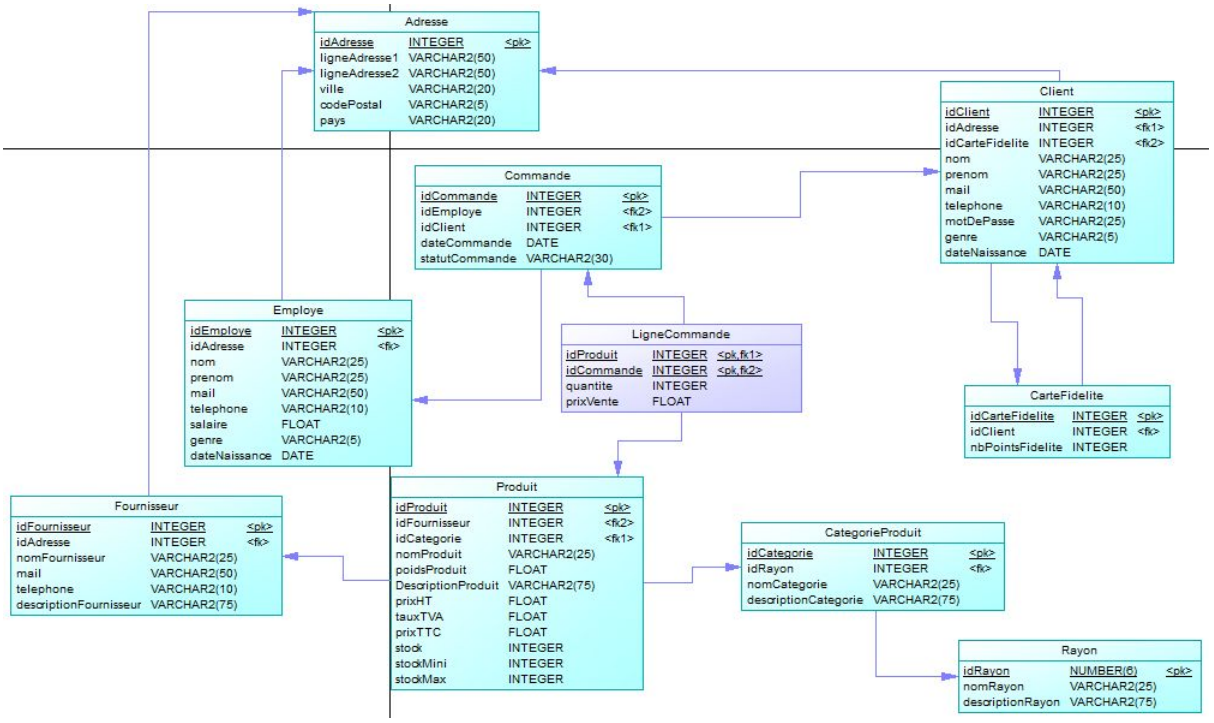
8. La définition du modèle logique de Données ou schéma relationnel

SCHÉMA RELATIONNEL

- **Client** (idClient, nom, prenom, mail, telephone, motDePasse ,genre, dateNaissance, #idAdresse, #idCarteFidelite)
- **Adresse** (idAdresse, ligneAdresse1, ligneAdresse2, ville, codePostal, pays)
- **CarteFidelite** (idCarteFidelite, nbPointsFidelite)
- **Produit** (idProduit, nomProduit, poidsProduit, descriptionProduit, prixHT, tauxTVA, prixTTC, stock, stockMini, stockMax, #idFournisseur, #idCategorieProduit)
- **CategorieProduit** (idCategorie,NomCategorie, descriptionCategorie, #idRayon)
- **Employe** (idEmploye, nom, prenom, mail, telephone, salaire, genre, dateNaissance, #idAdresse)
- **Rayon** (idRayon, nomRayon, descriptionRayon)
- **Fournisseur** (idFournisseur, nomFournisseur, mail, telephone, descriptionFournisseur, #idAdresse)

- **Commande** (idCommande,dateCommande, statutCommande, #idClient, #idEmploye)
- **LigneDeCommande** (#idProduit, #idCommande, quantite, prixVente)

MODÈLE LOGIQUE DE DONNÉES



9. Spécification des traitement avec des packages PLSQL (Modèle de traitements)

Pour le package pack_employe:

function getEmployeByID(idemp IN number) return Employe%rowtype;

-- Cette fonction permet de rechercher un Employe connaissant
-- en cherchant par un idCommande.
function getEmployeByIDComm(idCom IN number) return Employe%rowtype;

-- Cette fonction permet de rechercher tout les employes
function getAllEmploye return pack_employe.refCursorTyp;

-- Cette fonction permet de compter les employes
function getEmployeTotal return number;

-- Cette procedure permet d'inserer un nouveau Employe dans la
-- table Employe.
procedure insertEmploye(lp IN Employe%rowtype) ;

-- Cette procedure permet de modifier le salaire de tous les
-- habitant la meme adresse.
procedure updateEmployeSalaireByAdr(adresse IN number, sal IN number) ;

-- Cette procedure permet de modifier le telephone de
-- l'employe numero x
procedure updateEmployeTelephoneById(id in number, tel IN varchar2) ;

-- Cette fonction permet de rechercher les Employes qui
-- habite a la meme adresse et trié par salaire.
function getEmployeByAdr(adresse IN number) return pack_employe.refCursorTyp ;

-- Cette procedure permet de supprimer un employe en fonction de son id
procedure deleteEmployeById(id IN number);

pour le package pack_Commande :

-- Cette fonction permet de rechercher une commande connaissant
-- son numero.
function getCommandeByID(idcom IN number) return commande%rowtype;

-- Cette fonction permet de rechercher une commande connaissant
-- en cherchant par un idCom.
function getCommandeByidCmp(idCmp IN number) return commande%rowtype;

-- Cette fonction permet de rechercher toutes les commandes
function getAllCommade return pack_Commande.refCursorTyp;

-- Cette fonction permet de compter les commandes
function getCommandeTotal return number;

-- Cette procedure permet d'inserer une nouvelle commande dans la
-- table Commande.
procedure insertCommande(lp IN Commande%rowtype) ;

-- Cette procedure permet de modifier l'état d'une commande
procedure updateCommandeEtatById(id in number, etat IN varchar2) ;

-- Cette procedure permet de modifier la date de
-- la commande
procedure updateCommandeDateById(id in number, dateC IN date) ;

-- Cette fonction permet de rechercher les Commandes qui ont
-- pour etat livree.
function getCommandeByEtat(etat IN varchar2) return pack_Commande.refCursorTyp ;

-- Cette procedure permet de supprimer une commande en fonction de son id
procedure deleteCommandeById(id IN number);

10. Spécification des triggers

Pour ce qui est des triggers nous avons décidé d'en créer trois. Les triggers sont ajoutés à la fin du fichier *Schema_physique_Gestion_Supermarche_BORDY_CHTOUI_EL-GHAZZAZ.sql*
Le premier trigger est le trigger qui se nomme : trig_PrixTTC.
Il se déclenche avant l'insertion ou la modification d'une ligne dans la table produit.
Ce trigger va permettre de calculer le prix TTC d'un produit soit PRIXHT + (PRIXHT * TAUXTVA).

Le second trigger est le trigger qui se nomme : trig_PrixVente.
Il se déclenche avant l'insertion ou la modification d'une ligne dans la table ligneCommande.
Ce trigger va permettre de calculer le prix de vente d'une ligne produit soit la quantité du produit * prixTTC du produit.

Le dernier trigger est le trigger qui se nomme : trig_CarteFidelite.
Il se déclenche après l'insertion d'une ligne dans la table CarteFidelite.
Ce trigger va permettre de modifier le champ IdCarteFidelite du client dont son id est renseigné dans l'insertion de CarteFidelite.