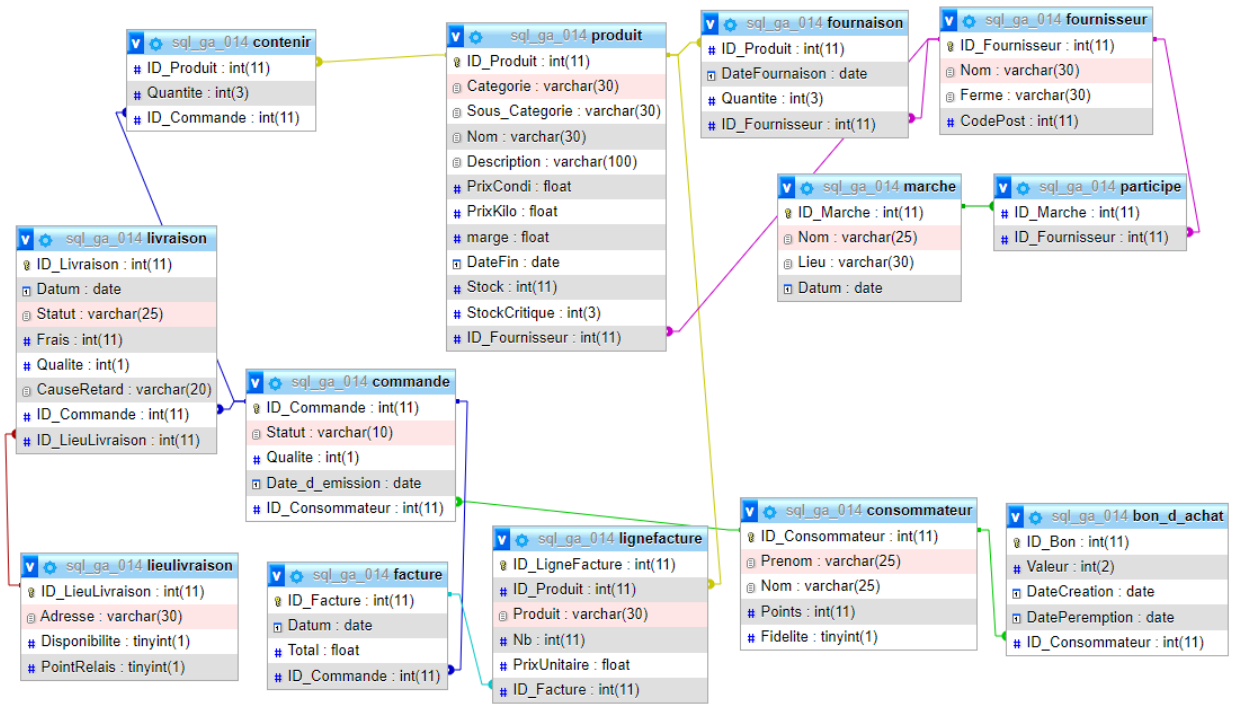
ADDE Mathis
AMON Pauline
AUBRY Gatien
ARNAUDO Alexandre
BABOK Julien
ENGEL Victoria

Guide de fonctionnement SQL

Projet Base de données VBA/SQL



Visuel graphique des composants et des différentes tables de la base de données ainsi que de leurs interactions

Table des matières

1) Gestion du catalogue produits et des fournisseurs

a. Ajout, modification ou suppression d'un produit ou d'un fournisseur

Ajout d'un produit

Nom de la fonction : PRODUIT_ADD

Arguments :

Nom	Type	Taille/Valeurs*
categorie	VARCH	30
sous_cate	VARCH	30
nom	VARCH	30
descriptior	VARCH	100
prixcondi	FLOAT	
prixkilo	FLOAT	
dateperem	DATE	---
quantite	INT	11
idfourni	INT	11

Code :

```
INSERT INTO `produit` (`ID_Produit`, `Categorie`, `Sous_Categorie`, `Nom`, `Description`, `PrixCondi`, `PrixKilo`, `DateFin`,
`StockCritique`, `ID_Fournisseur`)
VALUES (0, categorie, sous_categorie,nom,description,prixcondi,prixkilo, dateperemption, quantite, idfourni)
```

Explications :

Afin de pouvoir ajouter un produit, la procédure PRODUIT_ADD a été créée. Elle prend en entrée tous les éléments qui composent un produit (sauf l'id qui s'incrémente automatiquement et le stock de 0 à l'origine) et les insèrent dans la table des produits.

Modification d'un produit

Nom de la fonction : PRODUIT_UPDATE

Arguments :

Nom	Type	Taille/Valeurs*
idprod	INT	10
idfourni	INT	10
cat	VARCHAR	30
sous_cat	VARCHAR	30
nommage	VARCHAR	30
descrip	VARCHAR	100
prixcond	FLOAT	
prixk	FLOAT	
dateperempti	DATE	---
quantite	INT	11

Code :

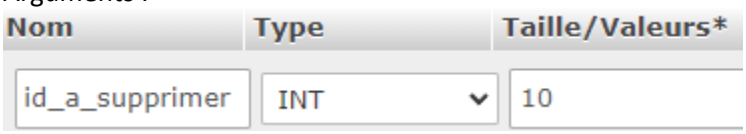
```
Begin -- Début de la fonction d'update
#Changement de l'id du fournisseur du produit
if idfourni <> 0 then
update `produit`
SET `ID_Fournisseur` = idfourni
WHERE produit.ID_Produit = idprod;
END if;

#Changement de la catégorie du produit
if cat <> '' then
UPDATE `produit`
SET `Categorie` = cat
WHERE produit.ID_Produit = idprod;
END if;

#Changement de la sous-catégorie du produit
if sous_cat <> '' then
UPDATE `produit`
SET `Sous_categorie` = sous_cat
WHERE produit.ID_Produit = idprod;
END if;

#Changement du nom du produit
if nommage <> '' then
UPDATE `produit`
SET `Nom` = nommage
WHERE produit.ID_Produit = idprod;
END if;

#Changement de la description du produit
if descrip <> '' then
UPDATE `produit`
SET `Description` = descrip
WHERE produit.ID_Produit = idprod;
END if;

#Changement du prix conditionnel du produit
if prixcond <> 0 then
UPDATE `produit`
SET `PrixCondi` = prixcond
WHERE produit.ID_Produit = idprod;
END if;

#Changement du prix au kg du produit
if prixk <> 0 then
UPDATE `produit`
SET `PrixKilo` = prixk
WHERE produit.ID_Produit = idprod;
END if;

#Changement de la date de préremption du produit
if dateperemption <> 0 then
UPDATE `produit`
SET `DateFin` = dateperemption
WHERE produit.ID_Produit = idprod;
END if;

#Changement de la quantite du produit
if quantite <> 0 then
UPDATE `produit`
SET `StockCritique` = quantite
WHERE produit.ID_Produit = idprod;
END if;
-- Fin de la fonction d'update
END
```

Explications :

Afin de pouvoir modifier un produit, la procédure PRODUIT_UPDATE a été créée. Elle prend en entrée tous les éléments qui composent un produit (sauf la quantité). En fonction de la présence de valeurs en entrée (nulles si laissées vides lors de l'appel de la fonction), elle modifie les valeurs du produit dont l'id correspond à celui donné en entrée.

Suppression d'un produit

Nom de la fonction : PRODUIT_ERASE
Arguments :



Code :

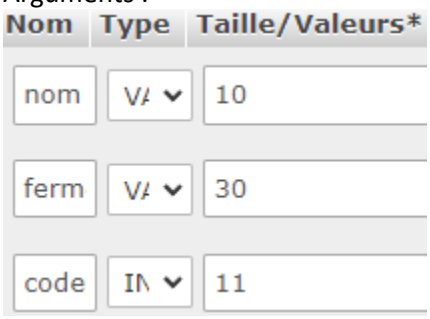
```
DELETE FROM `produit` WHERE produit.ID_Produit = id_a_supprimer
```

Explications :

Pour supprimer un produit, on utilise la fonction PRODUIT_ERASE. Cette fonction prend en entrée l'id du produit à supprimer et supprime la ligne de la table des produits dont l'id correspond à celui entré.

Ajout d'un fournisseur

Nom de la fonction : FOURNISSEUR_ADD
Arguments :



Code :

```
INSERT INTO `fournisseur` (`Nom`, `Ferme`, `CodePost`)
VALUES (nom, ferme, codepostal)
```

Explications :

Pour ajouter un fournisseur, on utilise la fonction FOURNISSEUR_ADD. Cette fonction prend en entrée tous les éléments qui composent un fournisseur (sauf l'id qui s'incrémente automatiquement) et les insèrent dans la table des fournisseurs.

Modification d'un fournisseur

Nom de la fonction : FOURNISSEUR_UPDATE

Arguments :

Nom	Type	Taille/Valeurs*
id	INT	11
name	VARCHAR	30
ferm	VARCHAR	30
codepostal	INT	11

Code :

```
Begin

#Changement de la ferme du fournisseur
if ferm <> '' then
update `fournisseur`
SET `Ferme` = ferm
WHERE fournisseur.id_fournisseur = id;
END if;

#Changement du code postal du fournisseur
if codepostal <> 0 then
update `fournisseur`
SET `CodePost` = codepostal
WHERE fournisseur.id_fournisseur = id;
END if;

#Changement du nom du fournisseur
if name <> '' then
update `fournisseur`
SET `Nom` = name
WHERE fournisseur.id_fournisseur = id;
END if;

end
```

Explications :

Pour modifier un fournisseur, on utilise la fonction FOURNISSEUR_UPDATE. Cette fonction prend en entrée tous les éléments qui composent un fournisseur et modifie les valeurs du fournisseur correspond à l'id quand les entrées ne sont pas nulles.

Suppression d'un fournisseur

Nom de la fonction : FOURNISSEUR_ERASE
Arguments :

Nom	Type	Taille/Valeurs*
id_a_supr	INT	10

Code :

```
DELETE FROM fournisseur WHERE fournisseur.ID_Fournisseur = id_a_supr
```

Explications :

Pour supprimer un fournisseur, on utilise la fonction FOURNISSEUR_ERASE. Cette fonction prend en entrée l'id du produit à supprimer et supprime la ligne de la table des produits dont l'id correspond à celui entré.

b. Affichage d'une fiche produit (avec toutes ses caractéristiques) associé à son producteur et au stock en cours pour le site de vente en ligne.

Note : La recherche se fait par la catégorie du produit

Nom de la fonction : FICHE_PRODUITS_CATEGORIE
Arguments :

Direction	Nom	Type	Taille/Valeurs*
IN	cat	VARCHAR	30

Code :

```
1 SELECT *, fournisseur.Nom
2 FROM produit
3 INNER JOIN  fournisseur on produit.ID_Fournisseur = fournisseur.ID_Fournisseur
4 where produit.Categorie = cat
```

Explications :

Pour afficher une fiche produite, on utilise la fonction FICHE_PRODUITS_CATEGORIE qui prend en entrée la catégorie du produit. Cette fonction va ensuite renvoyer tous les produits (et le nom des fournisseurs) dont la catégorie correspond à la variable d'entrée.

c. Affichage d'une fiche fournisseur (avec ses caractéristiques principales) associé à son historique d'approvisionnement

Nom de la fonction : FICHE_FOURNISSEUR
Arguments :

Nom	Type	Taille/Valeurs*
id	INT	10

Code :

```
SELECT fournisseur.Nom, fournisseur.Ferme, fournisseur.CodePost, fournaison.ID_Produit,
fournaison.Quantite, fournaison.DateFournaison
from fournisseur
INNER JOIN fournaison on fournaison.ID_Fournisseur = fournisseur.ID_Fournisseur
where fournisseur.ID_Fournisseur=id
```

Explications :

Pour obtenir les informations liées à un fournisseur et son historique d'approvisionnement, on utilise la fonction FICHE_FOURNISSEUR avec en entrée l'id du fournisseur que l'on cherche. La fonction renvoie ensuite tout l'historique d'approvisionnement de ce fournisseur avec ses informations (nom, ferme, code postal).

d. Bilan de stocks (affichage des stocks critiques) et saisie du seuil critique

Affichage des stocks critiques

Note : Chaque produit a un stock critique fixé à 10 par défaut

Nom de la fonction : BILAN_STOCK_DEFAUT
Arguments :
Aucun

Code :

```
SELECT produit.ID_Produit,produit.Nom,produit.Stock
FROM produit
where produit.Stock<produit.StockCritique
```

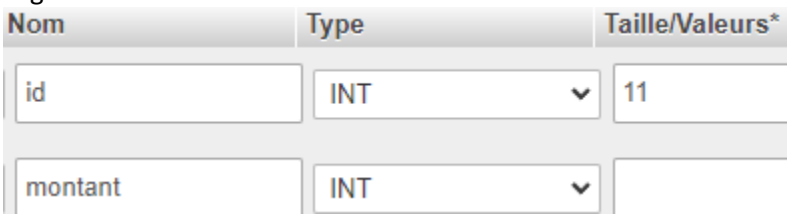
Explications :

La fonction BILAN_STOCK_DEFAUT ne prend aucun argument en entrée et sort une liste de tous les produits dont le stock actuel est inférieur au seuil critique de ce produit.

Saisie du seuil critique

Nom de la fonction : PRODUIT_SEUIL

Arguments :



Code :

```
update `produit`
SET `StockCritique` = montant
WHERE produit.ID_Produit = id
```

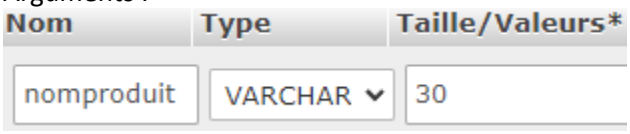
Explications :

Cette fonction permet, en rentrant un id et un montant, de remplacer la valeur du produit dont l'id correspond par le montant rentré.

e. Le prix moyen d'un produit donné sur l'ensemble des fournisseurs proposant ce produit

Nom de la fonction : PRIXMOYEN_PRODUIT

Arguments :



Code :

```
SELECT produit.Nom, AVG(produit.PrixKilo) FROM produit WHERE produit.Nom = nomproduit
```

Explications :

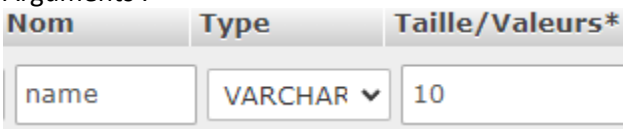
PRIXMOYEN_PRODUIT est une fonction qui renvoie le prix au kg moyen des produits dont le nom est égal à la variable d'entrée de la fonction.

2) Gestion des commandes clients

a. Affichage d'une fiche consommateur (avec ses caractéristiques principales) associé avec son historique commande et les factures associées.

Note : La recherche se fait sur le nom du consommateur ou sur la première lettre de son nom.

Nom de la fonction : FICHE_CONSOMMATEUR
Arguments :



Code :

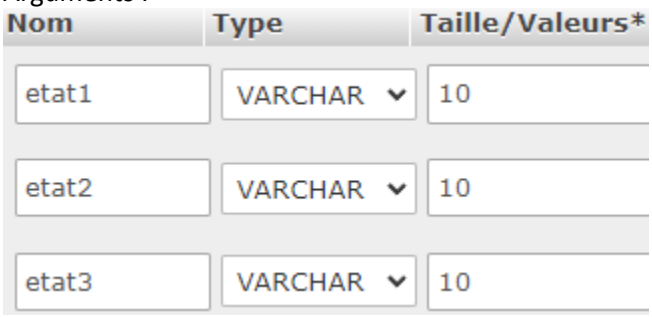
```sql
SELECT *, commande.ID_Commande, livraison.*
FROM consommateur
INNER JOIN commande on commande.ID_Consommateur = consommateur.ID_Consommateur
INNER JOIN livraison on livraison.ID_Commande = commande.ID_Commande
WHERE consommateur.Nom like concat(name,'%')
```

Explications :

La fonction FICHE_CONSOMMATEUR prend une séquence de caractères en entrée et retourne toutes les informations des consommateurs, l'id de ses commandes associées et les informations des livraisons des consommateurs dont le nom commence par la séquence de caractères.

b. La liste des commandes pour un ou plusieurs statuts de commande donné(s) avec pour chaque commande le nom des consommateurs associés

Nom de la fonction : LISTE_COMMANDE_STATUTS
Arguments :



Code :

```sql
SELECT commande.ID_Commande, commande.Date_d_emission, consommateur.Nom
FROM commande
inner JOIN consommateur on commande.ID_Consommateur=consommateur.ID_Consommateur
where commande.Statut IN(etat1, etat2, etat3)
```

Explications :

Pour La fonction FICHE_CONSOMMATEUR prend jusqu'à trois séquences de caractères en entrée. Elle revoit toutes les commandes (avec le nom du consommateur associé) dont les statuts sont les mêmes qu'au moins une des variables d'entrée.

c. Les commandes avec les frais de livraison offerts avec le nom du consommateur

Nom de la fonction : LIVRAISON_OFFERTE
Arguments :
Aucun

Code :

```
SELECT livraison.Frais, consommateur.Nom, commande.ID_Commande, commande.Statut, commande.Date_d_emission
FROM livraison
INNER join commande on livraison.ID_Commande = commande.ID_Commande
inner JOIN consommateur on commande.id_consommateur = consommateur.ID_Consommateur
where livraison.Frais = 0
```

Explications :

Pour afficher les livraisons dont les frais sont gratuits, on utilise la fonction LIVRAISON_OFFERTE qui retourne toutes les livraisons (et le nom du consommateur associé) des livraisons dont les frais sont égaux à 0.

d. Indicateur de qualité des livraisons et des commandes

Variable des tables :

Les variables Qualite des tables commande et livraisons sont des int de taille 2 permettant de stocker des notes de 1 à 10.

Nom des fonctions : QUALITE_COMMANDE et QUALITE_LIVRAISON
Arguments : Aucun
Codes :

```
SELECT avg(commande.Qualite) from commande commande
WHERE commande.Qualite
SELECT avg(livraison.Qualite) from livraison WHERE
livraison.Qualite
```

Explications :

On calcule la valeur moyennes des valeurs existantes de note de qualité sur l'ensemble des commandes/livraisons.

3) Gestion commerciale

a. Bilan des ventes annuelles (chiffre d'affaires et marge commerciale) et comparaison de l'année n-1

Nom de la fonction : BILAN_VENTES_ANNUELLES
Arguments :

Nom	Type	Taille/Valeurs*
an	YEAR	

Code :

```
SELECT year(commande.Date_d_emission) as Année, sum(produit.PrixCondi*contenir.Quantite) as CA,
sum(produit.marge*contenir.Quantite) as Marge
FROM commande
INNER join contenir on commande.ID_Commande = contenir.ID_Commande
INNER join produit on contenir.ID_Produit = produit.ID_Produit
where year(commande.Date_d_emission) = an or year(commande.Date_d_emission) = an-1
group by year(commande.Date_d_emission)
```

Explications :

Pour obtenir le bilan des ventes annules, on utilise la fonction BILAN_VENTES_ANNUELLES avec en argument l'année que l'on souhaite observer. Dans la procédure l'on calcul le bilan pour chaque produit intégré dans une commande, puis l'on regroupe les résultats par année pour l'année en entrée et l'année n-1.

b. Bilan des ventes (chiffre d'affaires) par catégories de produits

Nom de la fonction : BILAN_VENTES_CATEGORIE
Arguments : Aucun
Code :

```
SELECT produit.Categorie, sum(lignefacture.Nb*lignefacture.PrixUnitaire) as MONTANT
FROM produit
InNeR jOiN lignefacture on produit.ID_Produit = lignefacture.ID_Produit
group by produit.Categorie
```

Explications :

BILAN_VENTES_CATEGORIE est une procédure qui retourne les catégories et la somme des ventes correspondantes (en regardant les factures) puis en les regroupant par catégorie.

c. Obtenir le nombre de produits vendus par producteur pour les producteurs d'un département donné

Nom de la fonction : PRODUITS_VENDUS
Arguments :

Nom	Type	Taille/Valeurs*
reg	INT	2

Code :

```
SELECT sum(lignefacture.Nb) as Quantite, fournisseur.nom, fournisseur.CodePost
from fournisseur
INNER join produit on fournisseur.ID_Fournisseur = produit.ID_Fournisseur
inner JOIN lignefacture on produit.ID_Produit = lignefacture.ID_Produit
where fournisseur.CodePost LIKE concat(reg,'%')
GROUP by fournisseur.ID_Fournisseur
```

Explications :

Pour obtenir le nombre de produits vendus des fournisseurs d'un département donné, on entre en argument de la fonction PRODUITS_VENDUS les 2 premiers chiffres du numéro de département. La procédure retournera la somme des quantités des lignes de facturation, le nom de chaque fournisseur et leurs code postaux respectifs.

d. Mise en place d'alerte des livraisons en retard

Nom de la fonction : ALERTE_RETARD
Arguments : Aucun
Code :

```
SELECT livraison.ID_Livraison,  commande.ID_Commande, commande.Date_d_emission
FROM commande
inner join livraison on commande.ID_Commande = livraison.ID_Commande
where livraison.Statut = 'En cours'  and datediff(now(),commande.Date_d_emission) > 10
```

Explications :

La fonction ALERTE_RETARD retourne les informations principales de toutes les commandes dont la date d'émission de la livraison est passée de 10 jours.

e. Mise en place de statistiques sur les causes des retards

Nom de la fonction : STATS_CAUSE_RETARDS
Arguments : Aucun
Code :

```
SELECT livraison.CauseRetard as Cause,
count(livraison.CauseRetard) as NombreDeFois
FROM livraison
wHeRe livraison.CauseRetard is not NULL
GROUP By livraison.CauseRetard
```

Explications :

La fonction STATS_CAUSE_RETARDS renvoie le nombre d'occurrence de chaque cause de retard de la table des livraisons

e. Mise en place de statistiques sur la satisfaction des consommateurs

Nom de la fonction : STATS_SATISFACTION
Arguments : Aucun
Code :

```
1 SELECT (avg(livraison.Qualite) + avg(commande.Qualite))/2 as
  NoteGlobaleDesUtilisateurs from livraison, commande
2 WHERE livraison.Qualite or commande.Qualite
```

Explications :

STATS_SATISFACTION permet d'obtenir la moyenne des notes données par les utilisateurs sur les qualités des commandes et des livraisons quand des notes ont étés données.