RAPPORT : projet BDD

Un projet fait par : ***BREGER Aymeric, BURY Jean-Baptiste, BENYEMNA Hamza***

I. Modèle E/A :

Nous avons globalement respecté le cahier des charges de VéloMax. Cependant, nous avons pris des libertés sur les zones d'ombres afin de trouver une solution pour les gérer au mieux. Notre modèle E/A est conforme avec notre base SQL car nous avons utilisé *JMerise* pour vérifier notre modèle et créer une base à partir de celui-ci.

Bicyclettes :

Chaque bicyclette doit être montée par VéloMax. Ainsi, il existe en plus des attributs, nous avons ajouté un **id_bicyclette** afin de l'utiliser comme clé primaire. Ainsi, il existera plusieurs bicyclette avec le même numéro avec lequelle nous pourrons les trier mais chaque bicyclette aura sa propre id.

Pièces :

Nous avons décider que le numéro de produit et le numéro de fournisseurs seraient les mêmes pour un soucis de cohérence avec le cahier des charges lors des commandes auprès du fournisseur pour plusieurs pièces du même numéro de produit. Cependant, celui-ci est le même pour un ensemble de pièces. C'est pourquoi nous avons créer un **id_produit_piece** qui permet de distinguer chaque pièce comme les vélos.
En plus de cela, nous avons ajouté la clé primaire de la bicyclette en clé étrangère de la pièce pour distinguer les pièces qui servent au montage des bicyclettes des pièces de rechange.

Fournisseurs :

Respecter le cahier des charges nous a suffit, seulement une précision déjà mentionnée a été apportée concerant le numéro de fournisseur qui est le même que le numéro de produit. Un fournisseur ne peut fournir que des pièces.

Fournir :

Cette table a été ajouté après traduction du modèle relationnel. Elle permet de lier le fourniseur et les picèes qu'il fourni. Elle permet de préciser le **délai d'approvisionnement** en plus du numéro de produit de la pièce.

Clients :

Nous avons décider de distinguer 2 entités de clients : les entreprises et les particuliers. Ainsi, l'entreprise aura un **numéro de siret** unique, quant au particulier, il aura un **id_particulier**. De plus , cela nous permet une meilleure gestion des promotions. En ce qui concerne les entreprises, elles ont un attributs **pourcentage_remise_promotion** qui permet une remise sur leur commande. Les particuliers auront un **numéro de programme Fidelo** qui correspond au programme Fidelio auquel ils sont inscrits. Si ces attributs de promotions sont nuls, cela indique qu'il n'y a pas de promotion pour le client.

Commandes :
La commande relie les produits (pièces et bicyclettes) aux clients. Pour éviter toute ambiguïté, chaque commande possède un **num_commande** unique. Si un produit est dans une commande, il aura en attribut le numéro de celle-ci. De cette manière, si le numéro de commande est nul, un produit n'est pas encore commandé. De même, dans la commande, on retrouve le **num_siret_entreprise** si c'est une entreprise, **l'id_particulier** si c'est un particulier. Mais une commande ne peut ces deux numéros en même temps car une commande est attribuée à un unique client.
En ce qui concerne les datas, nous avons trouvé de réels fournisseurs, de réel bicyclette, … afin de garantir une meilleure BDD.

II. Gestion des tables :

Affichage :
Nous avons implémenté les diverses fonctions d'affichages selon les critères demandés pour les tables demandée. Nous n'avons noté aucune erreur liée à la conception du modèle. De plus, pour gérer les **dépendances entre tables** lors d'une requête, nous avons procédé à des **triggers**.

Création :
La création ne pose acun problème. Cependant, nous l'avons sécurisé en empechant l'intégration d'un produit à une commande dès sa création.

Suppression :
En ce qui concerne les suppression, cela nous a demandé beaucoup de triggers. Une pièce faisant partie d'une bicyclette ne peut pas être supprimée. La suppression d'une bicyclette transforme les pièces qui le composaient en pièces de rechange à nouveau disponibles. La suppression d'un client entraine la suppression de ses commandes. La suppression d'une commande remet les produits dans les stocks.

Mise à jour :
La gestion des dépendances à été traitées de la même manière que les suppressions. Un produit peut être supprimé d'une commande ou ajouté en mettant à jour son numéro de commande. Cela aura pour effet de modifier la quantité de produit en conséquence. De même, une pièce peut être retirée d'une bicyclette ou la mettre dans une autre en mettant à jour son **id_bicyclette** en clé étrangère.

Stock :
On peut afficher les stocks des produit selon la marque, le numéro de protui, nom de fournisseur, … comme demandé.

III. Modules statistiques :

Nous avons tous les modules statistiques sauf la gestion de la date d'expiration des adhésions au programmes Fidelio car nous n'avons pas pris en compte le date d'adhésion. Nous avons plusieurs requêtes spéciales comme vous verrez ci-après.

Jointure :
select nom_particulier as 'nom',count(*) as 'nombre de commandes' from client_particulier **join** commande using(id_particulier) group by id_particulier;Clients :

Cette requête utilise une **jointure naturelle** pour avoir le cumul des commandes par clients.

Union et requête synchronisée :

```
select * from ( select num_produit_piece as num_produit,count(*) as nb_stock_piece from Piece
group by num_produit having nb_stock_piece<=2
union all select num_modele_bicyclette,count(*) as nb_stock_bicyclette from Bicyclette group by
num_modele_bicyclette having nb_stock_bicyclette<=2 ) as nb_stock;
```

Cette requête permet d'avoir le nombre de produit en stocks (bicyclettes et pièces) selon le numéro de produit.

Prix:

```
select sum(prix) as total from ( select num_commande,sum(number) as prix from (
select num_commande,sum(prix_unitaire_bicyclette) as number
from commande join bicyclette using(num_commande) group by num_commande
union all select num_commande,sum(prix_unitaire_piece) as number
from commande join piece using(num_commande) group by num_commande ) as a group by
num_commande ) as b;
```

Cette commande affiche le cumul du prix des commandes par clients.

Export :

JSON exporte toute la BDD dans un seul fichier.
XML exporte la BDD table par table.

IV. Visuel et mode démo :

Nous avons opté pour un **WPF**. Nous avons un mode démo comme demandé avec les utilités exigées.

Pour commencer, nous avons une page de login avec les connexions du type **lecture uniquement** avec (id : user-reader ; mdp : mdp-reader) et en **lecture et écriture** avec (id : user-writer ; mdp : mdp-writer). Pour les connexions, nous utilisons un fichier txt qui récupère les identifiant et les mdp insérés. Dans le **Menue**, il y a les affichages de base, avec des accès aux gestions que l'on nous demande. Une fois dans les pages de **Gestion** (Gestion de clients, gestions de commandes, gestion de fournisseurs, gestions de pièces, gestions de bicyclettes, gestion des stocks) on peut créer, mettre à jour, ou supprimer. Pour la **Gestion des clients**, nous avons le choix d'accéder à celle des **entreprises**, ou bien à celle des **particuliers**.

En implémentant tous les triggers crées, les problèmes liés aux dépendances ou aux problèmes de cohérence seront évités autant que possible.

Nous avons créé un bouton démo en jaune pour l'évaluateur et un bouton démo normal pour une démo complète pour modifier toutes les tables et qui complète tout le cahier des charges.