INF403 : PROJET SQL

Introduction :

Dans ce pré-rapport du projet SQL vous trouverez une description du sujet choisi puis le modèle UML correspondant et enfin le modèle relationnel avec les spécifications, les contraintes et les règles de traduction de noms.

Description du projet :

Une entreprise vendant des téléphones reconditionnés souhaite développer un système d'information pour gérer ses opérations. Les activités de l'entreprise impliquent plusieurs domaines :

Téléphone : Chaque téléphone est identifié par un numéro unique. Les détails tels que le modèle, la marque et la capacité de stockage sont enregistrés.

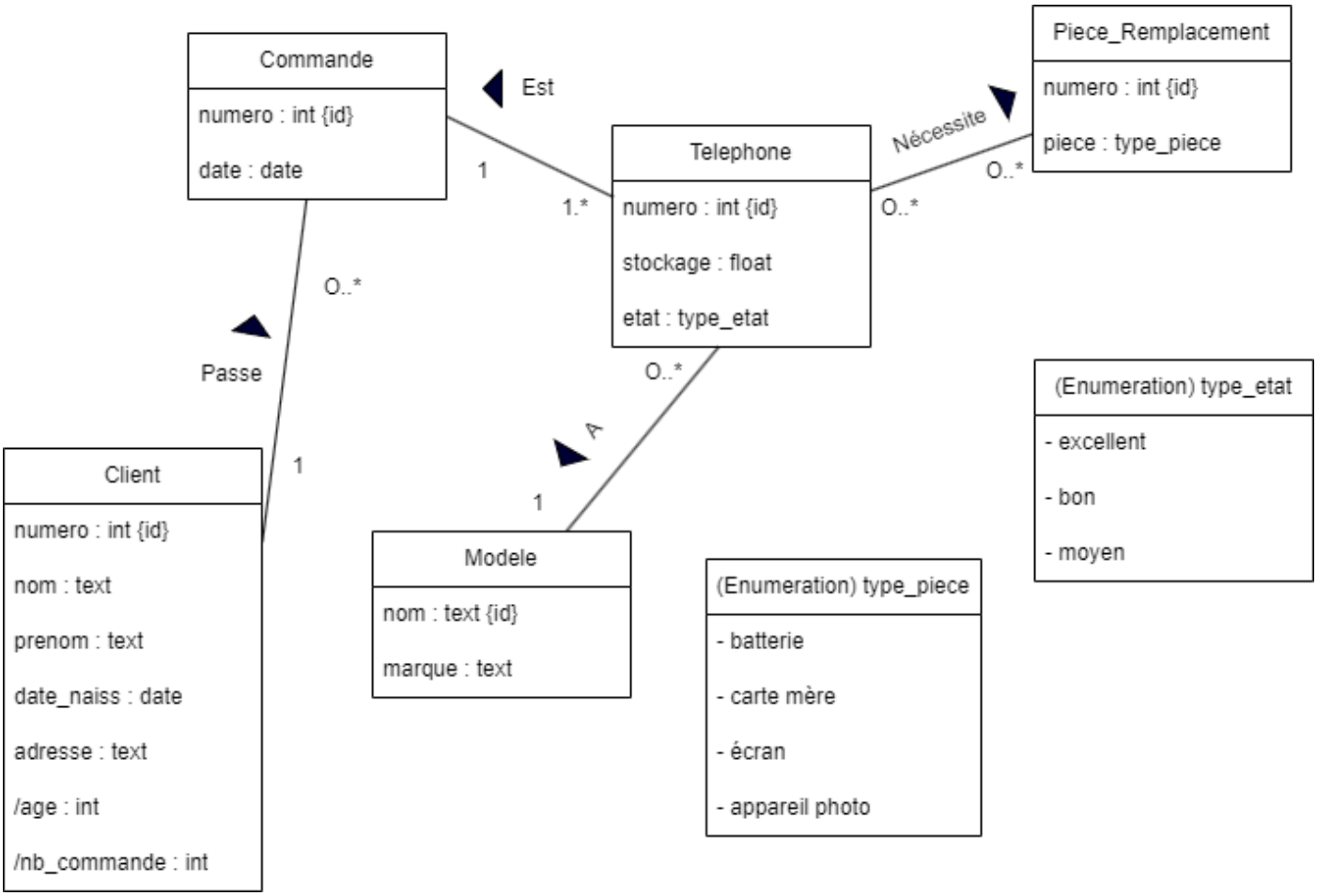
Client : Les clients de l'entreprise sont identifiés par leur numéro de client. Les informations de base telles que le nom, le prénom, la date de naissance et l'adresse sont enregistrées.

Commande : Chaque transaction de vente est enregistrée sous forme de commande. Une commande peut inclure un ou plusieurs téléphones reconditionnés.

Modèle : Chaque téléphone est associé à un modèle spécifique, qui est défini par un nom unique.

Pièce de Remplacement : Lorsqu'un téléphone est reconditionné, des pièces de remplacement peuvent être utilisées. Chaque pièce de remplacement est identifiée par un numéro unique et son type.

Modèle UML :



INF403 : PROJET SQL

Modèle relationnel :

Telephones(numero_telephone, stockage_telephone, etat_type_etat, nom_modele (fk), numero_commande (fk))

Modeles(nom_modele, marque_modele)

Clients_base(numero_client, nom_client, prenom_client, date_naiss_client, adresse_client)

Commandes(numero_commande, date_commande, numero_client (fk))

Pieces(numero_piece, piece_type_piece)

Reparations(numero_telephone, numero_piece)

(View) Clients(numero_client, nom_client, prenom_client, date_naiss_client, adresse_client, age_client, nb_commande_client)

Spécification :

Soit R(numero_telephone, stockage_telephone, etat_type_etat, nom_modele, numero_commande) la relation construite par la requête <n1, s, e, n2, n3 > ∈ R ⇔ Le téléphone identifié par le numéro n1, de stockage s, d'état e, de modèle n2 associé à la commande de numéro n3.

Soit R(nom_modele, marque_modele) la relation construite par la requête <n, m> ∈ R ⇔ Le téléphone de la marque m porte le nom n.

Soit R(numero_client, nom_client, prenom_client, date_naiss_client, adresse_client) la relation construite par la requête <n1,n2,p,d,a > ∈ R ⇔ Le client identifié par le numéro n1 porte le nom n2, le prenom p. Il est né à la date d et habite à l'adresse a.

Soit R(numero_commande, date_commande, numero_client) la relation construite par la requête <n,d, c> ∈ R ⇔ La commande de numéro n est passé à la date d par le client c.

Soit R(numero_piece, piece_type_piece) la relation construite par la requête <n,p> ∈ R ⇔ La pièce désignée par la numéro n est de type p.

Soit R(numero_telephone, numero_piece) la relation construite par la requête <t,p> ∈ R ⇔ Le téléphone t nécessite la pièce p pour être réparé.

Contraintes d'intégrité et domaines :

Reparations[numero_telephone] ⊆ Telephones[numero_telephone]

Reparations[numero_piece] ⊆ Pieces[numero_piece]

Telephones[nom_modele] ⊆ Modeles[nom_modele]

Telephones[numero_commande] ⊆ Commandes[numero_commande]

Commandes[numero_client] ⊆ Clients_base[numero_client]

Domaine(Etat_type_etat) ∈ {excellent, bon, moyen}

Domaine(Piece_type_piece) ∈ {batterie, carte mère, écran, appareil photo}

Domaine(numero_telephone) = domaine(stockage_telephone) = domaine(numero_commande) = domaine(numero_client) = domaine(numero_piece) = entiers ≥ 0

INF403 : PROJET SQL

Domaine(nom_modele) = domaine(marque_modele) = domaine(nom_client) = domaine(prenom_client) = domaine(adresse_client) = chaine de caractères

Domaine(date_naiss_client) = domaine(date_commande) = date

Règles de traduction :

Chaque attribut de classe en modèle logique est nommé de la sorte : attribut_(nom_de_la_classe_au_singulier)

Donc pour les attributs de Telephones on trouve par exemple 'numero_telephone' mais aussi 'etat_type_etat' qui doit prendre une valeur spécifique listé dans l'énumération type_etat ou encore 'nom_modele' puisque la classe Telephone doit tenir compte du modèle associé au téléphone se trouvant dans la classe Modeles.

Guide d'exécution du programme :

A l'exécution du programme se font la création de la base de données et l'initialisation des premières valeurs définies dans le fichier Donnees_OK.sql.

Ensuite un menu s'affiche, indiquant à l'utilisateur quelles requêtes il peut faire. Il faut taper le numéro de la requête pour qu'elle commence à s'exécuter. Pour la requête 1 taper 1, pour la requête 2 taper 2…et pour quitter taper q. Pour toute autre entrée que l'un des chiffres associés aux requêtes ou q, un message d'erreur s'affiche et le menu se recharge.

La première requête demande à l'utilisateur de rentrer le nom de la table qu'il veut rentrer, si le nom n'est pas correct on retourne au menu avec un message d'erreur explicatif.

La seconde requête affiche toutes les tables sans que l'utilisateur ait besoin de faire quoique ce soit.

Dans la troisième requête on offre à l'utilisateur la possibilité de supprimer des données en notifiant à l'utilisateur l'ordre à respecter pour ne pas violer les contraintes d'intégrité. L'utilisateur est guidé par le programme qui lui demande en fonction de la table voulue les paramètres qu'il doit rentrer. Une gestion du type et des erreurs est effectuée pour avertir l'utilisateur du format à respecter, cela sert aussi à éviter les injections SQL et donc à protéger notre code.

Pour la quatrième requête l'utilisateur peut choisir des données à ajouter dans la table qu'il veut, il faut néanmoins respecter les contraintes d'intégrité décrites plus haut sous peine d'erreur. Une nouvelle fois, à chaque valeur entrée par l'utilisateur une gestion des erreurs et du type est effectuée pour éviter les erreurs et protéger le code.

La cinquième requête permet d'afficher les informations d'un client et du téléphone qu'il a acheté en plus de fournir le nombre de réparations que le téléphone a reçu.

La sixième requête permet à l'utilisateur d'afficher les informations des clients en fonction du nombre de réparations dont a eu leur téléphone. Cette requête offre la possibilité à l'utilisateur de choisir l'opérateur binaire qu'il veut pour effectuer la comparaison.

La septième requête affiche à l'écran les informations des clients qui ont acheté un téléphone qui n'a pas eu besoin d'être réparé.

La huitième et dernière requête crée une vue calculant dynamiquement l'âge et le nombre de commandes passées par un client.

Enfin, pour quitter le programme il suffit de taper q.

A la fin de chaque requête on retourne au menu pour pouvoir effectuer autant de requête que voulu.

Les différents fichiers fournis peuvent être un peu long et compliqués, j'ai fait le choix de gérer tous les cas d'erreur possibles pour avoir une bonne interface et un code robuste ce qui allonge et complique le code. Cependant le fait de diviser les fonctions en plusieurs fichiers rend le tout plus clair et plus facile à modifier et lire. De plus le code est commenté pour essayer de rendre le tout facilement compréhensible.