Elise METAYER
Fanny SHEHABI

ESIR1 - INFO

ECOLE SUPERIEURE D'INGENIEURS DE RENNES

TP4 - BASES DE DONNEES

Le but de ce TP est de se familiariser avec la conception de schémas de base de données, nous allons notamment créer des modèles conceptuels des données (MCD) et des modèles logiques des données (MLD) afin de bien identifier et comprendre les liens entre les différentes tables.

I. EXERCICE 1

Dans ce premier exercice nous sommes chargées de concevoir la base de données d'une entreprise de vente d'électronique en ligne. Dans cette base de données, nous allons ajouter les quatre tables suivantes :

1. La table « produits » avec comme attributs : l'ID du produit, la marque, le prix unitaire, la description, les caractéristiques techniques et la catégorie.

2. La table « client » avec comme attributs : l'ID du client, le nom, la ville, l'adresse, le numéro de téléphone et l'adresse mail.

3. La table « commande » avec comme attributs : l'ID de la commande, le prix total, la date de commande et le statut de la commande.

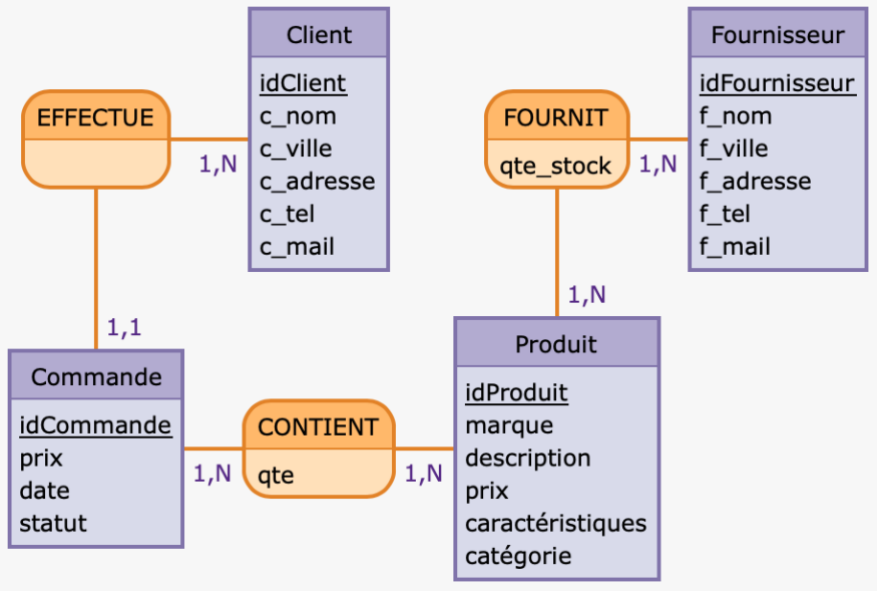
4. La table « fournisseur » avec comme attributs : l'ID du fournisseur, le nom, la ville, l'adresse, le numéro de téléphone et le mail.

Nous avons donc le schéma MCD suivant, avec 3 associations :

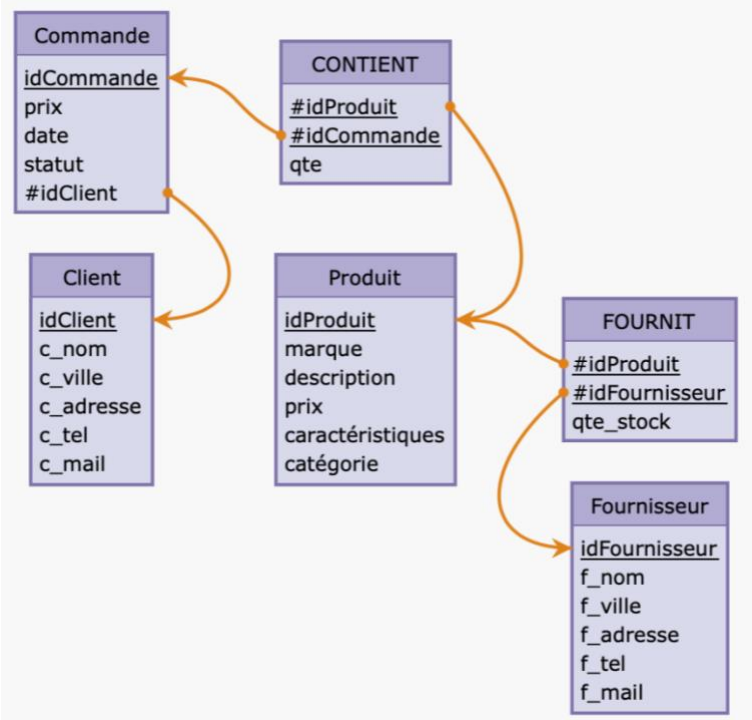
‘EFFECTUE’ entre client et commande. Client doit effecteur entre 1 et N commande(s) et commande ne peut être effectuée que par 1 unique client.

‘CONTIENT’ entre commande et produit. Commande doit contenir entre 1 et N produit(s) et produit doit être contenu dans 1 ou plus.

‘FOURNIT’ entre fournisseur et produit. Fournisseur doit fournir entre 1 et N produits et produit doit être fournit par 1 fournisseur ou plus.



De ce schéma MCD découle un schéma relationnel (MLD) :



Nous avons donc 2 nouvelles tables pour faire les liens :

La table ‘CONTENT’ avec comme attributs : la quantité commandée, la clé étrangère ID produit et la clé étrangère ID commande.

La table ‘FOURNIT avec comme attributs : la quantité en stock, la clé étrangère ID produit et la clé étrangère ID fournisseur.

Pas de table pour l'association EFFECTUE mais ajout de la clé primaire de client ‘ID client’ en tant que clé étrangère dans commande.

Nous avons utilisé un ‘UPDATE’ afin de ne pas avoir à rentrer manuellement le prix total de la commande et de ce fait éviter les erreurs humaines. Ce prix sera calculé en multipliant le prix unitaire de chaque produit avec sa quantité.

Nous aurions pu créer un Trigger afin d’actualiser les stocks de la base de données dès qu’une commande est effectuée, mais nous n’en avons pas eu le temps.

II. EXERCICE 2 : PREMIÈRE VERSION

Dans ce second exercice nous sommes chargées de concevoir une base de données pour une entreprise de location de voitures qui opère dans plusieurs pays. Dans cette base de données, nous allons ajouter les quatre tables suivantes :

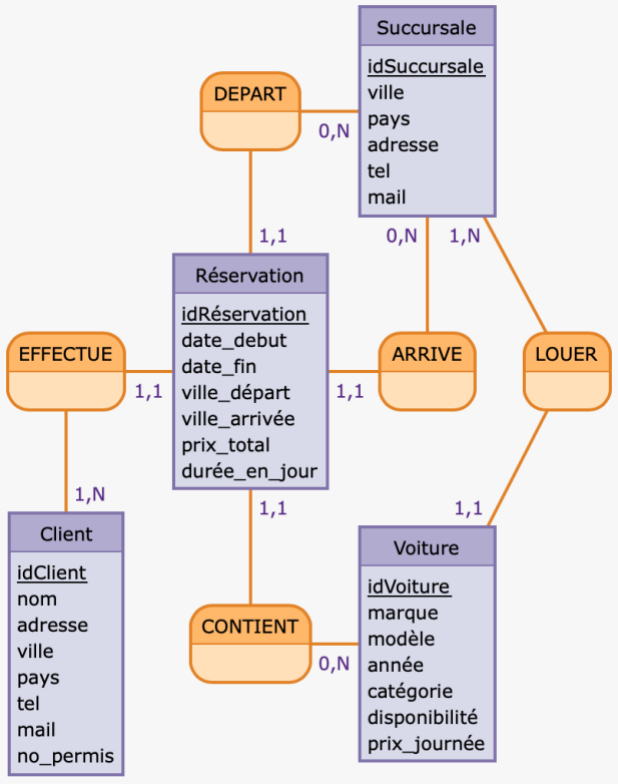
1. La table « Succursale » avec comme attributs : l'ID de la succursale, le nom de la ville dans laquelle se trouve la succursale, le pays dans laquelle elle se trouve, son adresse, son code postal, et son numéro de téléphone.

2. La table « Voiture » avec comme attributs : l'ID de la voiture, la marque, le modèle, l'année, la catégorie, la disponibilité, le prix de location à la journée. On a également l'ID de la succursale en tant que clé étrangère.

3. La table « Réservation » avec comme attributs : l'ID de la réservation, la date de début de location, la date de fin, la ville de location, le pays de location, la ville retour, le pays retour, la voiture louée, la durée en jour et le prix total. La table contient aussi les clés étrangères contenant l'ID client, l'ID de la voiture, l'ID de la succursale de départ et l'ID de la succursale d'arrivée.

4. La table « Client » avec comme attributs : l'ID client, son nom, son adresse, sa ville, son pays, son numéro de téléphone, son mail et son numéro de permis de conduire.

Nous avons donc effectué un modèle conceptuel de données (MCD) à l'aide de Mocodo :



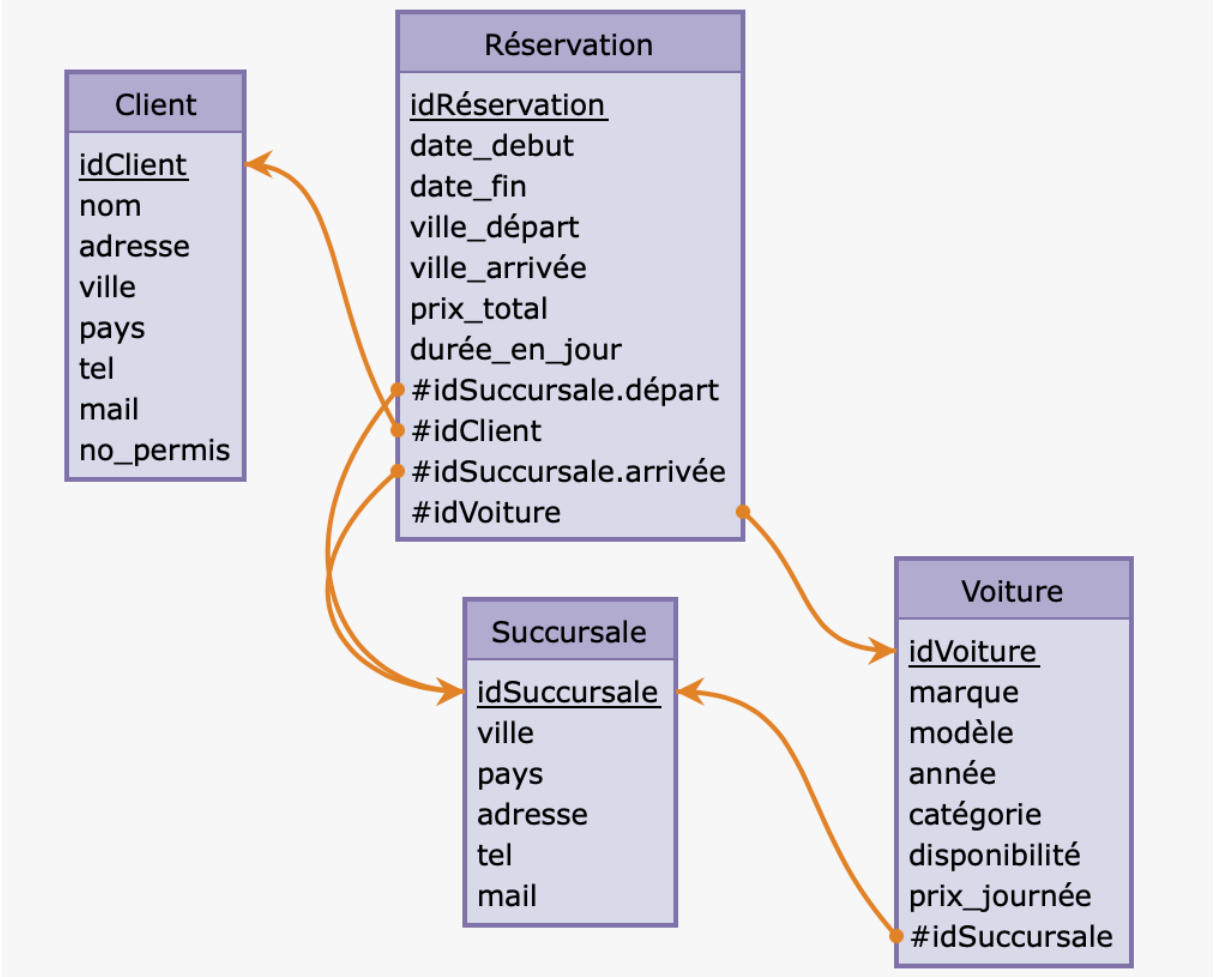
Une succursale possède au moins une voiture à mettre en location. En revanche, une voiture est mise à la location par une seule succursale.

Une voiture peut ne pas être réservée, mais elle peut aussi être réservée plusieurs fois (sur des périodes différentes) et donc être reliée à plusieurs réservations. Toutefois, une réservation est reliée à une seule voiture.

Un client peut effectuer une ou plusieurs réservation (sinon il n'est pas dans la base de données) mais une réservation est affectée à un unique client.

Une réservation est associée à une seule succursale de départ ou d'arrivée mais une succursale peut être associée à 0 ou plus réservations.

On déduit de ce MCD, un modèle logique de données :



Puisqu'une réservation est reliée à une unique voiture, nous avons ajouté aux requêtes de création de la table un UPDATE. Il permet de faire en sorte que le prix de location de la voiture soit égal au prix total.

Nous aurions pu créer un Trigger afin d'actualiser les disponibilités des voitures de la base de données dès qu'une commande est effectuée, mais nous n'en avons pas eu le temps.

Il reste cependant quelques ambiguïtés :

1. Comme dans le TP2 (ristourne), nous voulons pouvoir appliquer un tarif flexible. Nous pouvons modéliser cette politique de tarif flexible à l'aide d'un trigger.

2. Afin de prendre en compte les préférences spéciales des clients en matière de voitures, nous avons ajouter des attributs à la table voiture ('couleur' et 'transmission') et des attributs à la table client ('couleur_fav' , 'transmission_fav' et 'catégorie_fav').

3. Afin de modéliser le profil d'un client (entreprise ou particulier), nous avons ajouté l'attribut 'profil' à la table client.

4. Afin de pouvoir appliquer différentes règles pour la location de voiture suivant les différentes succursales (certaines pouvant exiger des cautions). Nous avons donc ajouté un attribut 'caution' de type float et initialisé par défaut à 0 dans la classe succursale. Si cet attribut est différent de 0, alors une caution s'applique.

III. Conclusion

Lors de ce TP, nous avons pu expérimenter la conception des modèles des données. Ils permettent, en effet, d'expliciter les liens entre les différentes tables. Ces schémas permettent aussi de mettre en évidence les problèmes d'une base de données. Nous avons pu nous rendre compte d'une chose : la conception de base de données est une discipline complexe. Effectivement, la base de données parfaite n'existe pas. Lorsque nous essayons de régler une ambiguïté, d'autres surviennent. Il faut donc essayer de minimiser les ambiguïtés.