- TP projet -
Base de données
Système de gestion et réservation d’un aéroclub

par Édouard Lumet

Sommaire

1. Présentation générale, description du projet

Ce projet consiste en la création d'une base de données qui assure la gestion d'un aéro-club. Le but est de répertorier les membres, les avions, disponibles, les réservations par membre et par avion et les comptes de chaque membre au sein de l'association. Par exemple, cette base de données doit pouvoir nous renseigner sur les membres inscrits ainsi que leurs coordonnées et leur(s) fonction(s) mais aussi sur les avions du club ou sur les réservations enregistrées pour tel avion à telles dates, etc.

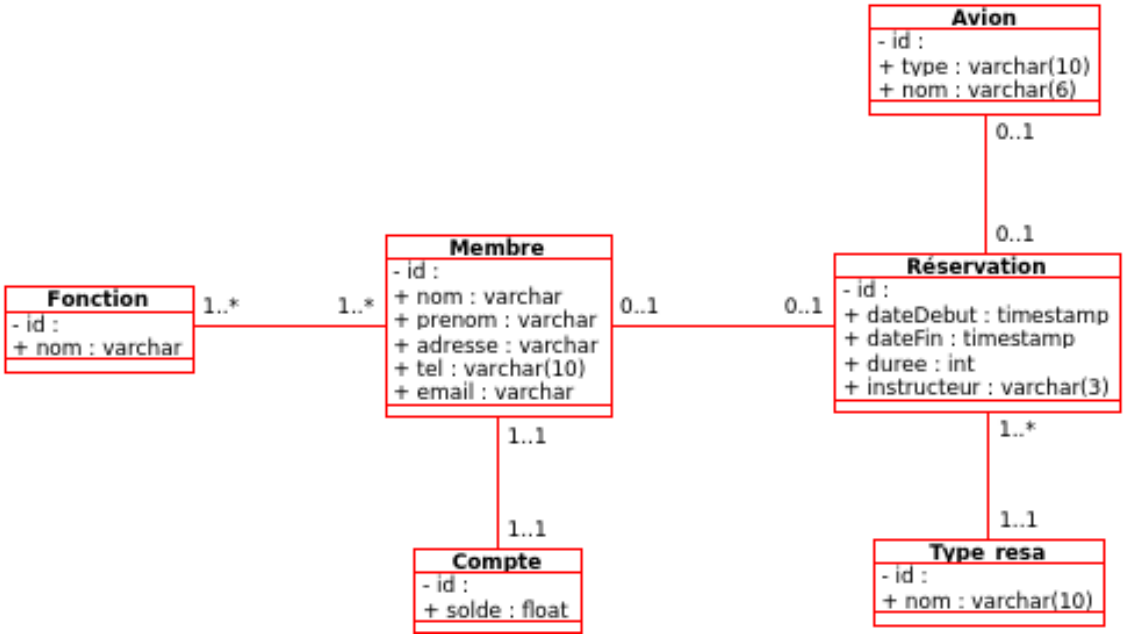
On peut donc facilement imaginer par la suite (dans les prochains modules de DESR par exemple…) que cette base de données puisse être implémentée sous la forme de service web avec une interface web à l'aide de PHP ou autre. La base de données est donc fondée sur un modèle synchronique, permettant à un instant T une réservations par avion et un avion par réservation.

Les données à créer/compléter sont les suivantes :

- les membres : nom, prénom, adresse, numéro de téléphone, adresse email et leur(s) fonction(s),
- les avions : nom de l'appareil (son immatriculation) et type,
- les réservations : dates de début et de fin, durée de la réservation, instructeur si vol d'instruction et le type de réservation.

2. Schéma conceptuel : diagramme UML

D'après le cahier des charges précédent et la description du projet, on peut alors établir le diagramme UML suivant qui correspond au niveau conceptuel de la programmation de notre base de données.



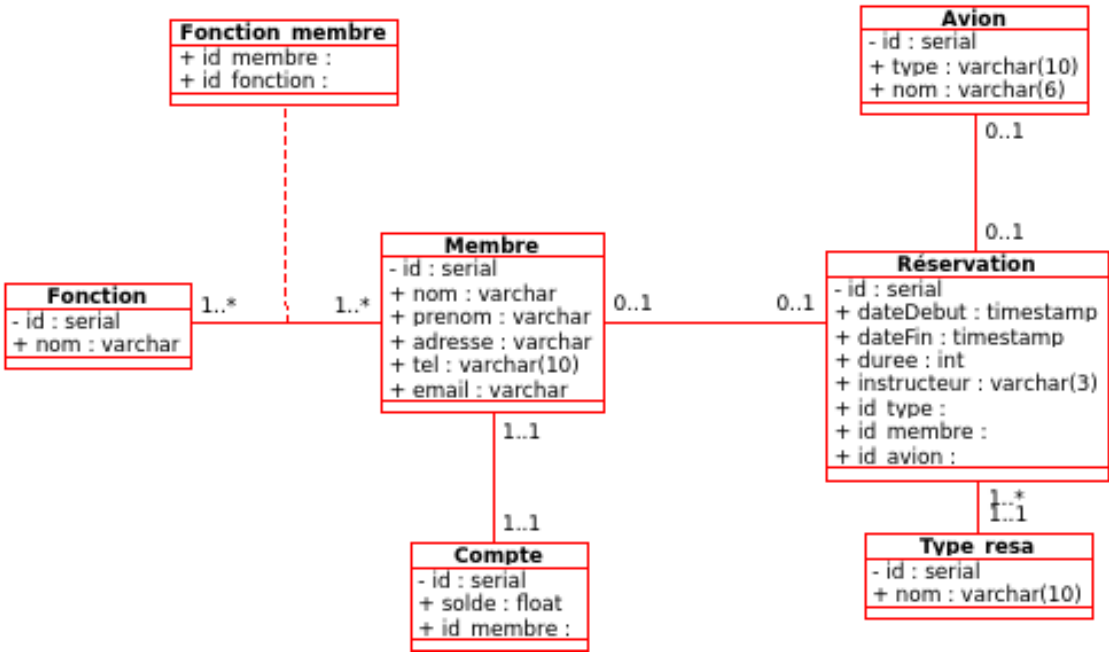
On retrouve alors 6 classes dans notre diagramme :

- Fonction : regroupe toutes les fonctions au sein de l'association,
- Membre : contient tous les membres de l'aéro-club,
- Compte : regroupe les soldes des comptes membres au sein de l'association,
- Réservation : liste l'ensemble des réservations d'un membre pour un avion,
- Avion : contient tous les avions du club,
- Type_resa : regroupe les types de réservations possibles (école, navigation, …)

Enfin, les cardinalités correspondent au cahier des charges et au modèle synchronique comme décrit dans la description du projet.

3. Schéma logique

Ensuite, nous déterminons le schéma logique en fonction des cardinalités mises en évidence dans le diagramme UML. Il permet de voir toutes les tables qui seront à créer par la suite ainsi que les contraintes liées à celles-ci.



Nous avons maintenant 6 classes qui correspondront à autant de tables dans notre base de données, plus une autre classe particulière qui correspondra à une table d'association.

Les cardinalités entre les classes *Fonction* et *Membre* sont 1..* / 1..* ce qui nous oblige à avoir recours à une classe associant ces deux dernières. En effet, un membre peut avoir plusieurs fonctions et une fonction peut être assignée à plusieurs membres. Ensuite, entre les classes *Compte* et *Membre* on repère 1..1 / 1..1 donc il faut intégrer une clé étrangère dans la table *Compte*. De même pour la table *Réservation* qui contiendra trois clés étrangères faisant référence aux clés primaires de *Membre, Avion* et *Type_resa*.

4. Niveau physique : langage SQL

Conformément au schéma précédent, nous pouvons désormais créer nos tables en langage SQL sur PostgreSQL puis les peupler avec des données.

Les requêtes SQL permettant la création de tables se situent dans la première partie du fichier SQL joint. Les requêtes concernant le peuplement se trouvent dans la seconde partie de ce même fichier.

L'ordre de création et de peuplement des tables est très important à cause des contraintes telles que les les clés étrangères. En effet, l'ordre de création de nos tables est le suivant :

1. membre
2. fonction
3. fonction_membre
4. compte
5. avion
6. type_resa
7. reservation

De même pour le peuplement des tables avec les données voulues.

5. Interrogation de la base de données : requêtes SQL

Afin de concrétiser notre projet, nous établissons une liste de questions/requêtes que nous traduisons ensuite en langage SQL pour interroger notre base de données. Les questions et requêtes que j'ai choisi sont les suivantes :

- trouver la liste des membres par ordre alphabétique des noms (nom, prénom, adresse, téléphone, email),
- afficher les membres ainsi que leur(s) fonction(s) dans l'association (nom, prénom, nom fonction),
- trouver les coordonnées de tous les instructeurs de l'aéro-club (nom, prénom, adresse, téléphone, email),
- quels sont les élèves parmi les membres de l'association ? par ordre alphabétique (nom, prénom),
- trouver la liste des comptes par membre en ordre alphabétique (nom, prénom, solde),
- quels sont les membres ayant un solde inférieur à 130€ par ordre croissant ? (nom, prénom, solde),
- quels sont les réservations enregistrées dans la base de données par ordre chronologique ? (nom, prénom, datedebut, datefin, durée, nom avion, type réservation),
- trouver la liste des réservations de type école par ordre chronologique (nom, prénom, datedebut, datefin, duree, nom avion, type réservation),
- afficher les trois statistiques suivantes sur les réservations en 2015 :
  - durée moyenne d'une réservation en minutes,
  - durée totale de l'ensemble des réservations (somme) en heures,
  - nombre total de réservations,
- quels sont les avions ayant été réservés plus de 100h ? (non avion, somme des durées par avions en heure).

Les requêtes en SQL se trouvent dans la dernière partie du fichier SQL joint.