Ahma Diallo
Florian Caillol
Téo Lambert
Thomas Laurent

L3 MIAGE

Projet Base de Données 2020

I- Introduction

Le but de ce projet est de réaliser une application Java accédant via le protocole JDBC à une base de données sur un serveur distant permettant de gérer la planification et la réservation d'une compagnie aérienne. Ainsi nous allons dans un premier temps indiquer les modifications appliquées sur le schéma conceptuel, ensuite nous ferons une description de l'état d'avancement du projet et puis nous finirons par expliquer les opérations de l'application.

II- Modification du schéma conceptuel

Nos contraintes :

- Le numéro de vol pouvant être réutilisé sur plusieurs vols, on choisira d'utiliser un identifiant de vol arbitraire en guise de clé primaire, et le numéro de vol sera stocké en clé étrangère.

- La terminaison du vol sera gérée par un boolean, qui sera automatiquement actualisé une fois le vol terminé.
- Les places d'un avion sont gérées dans une table dédiée "Place ", qui renseigne notamment a quelle avion elles appartiennent, leur classe ainsi que la position. Le nombre de place par classe sera facilement calculable grâce à un count. L'intérêt de gérer les places dans une table à part qui renseigne le numéro d'avion est que les places d'un même modèle d'avion peuvent varier sans impacter les autres avions de même modèle.

- Concernant le numéro de place, on utilisera le numéro d'allée ainsi qu'une lettre pour désigner la position dans la rangée (2A pour la place située deuxième allée côté hublot rangée de gauche, 2B pour la place située au milieu rangée gauche, etc…)

- L'identifiant du client sera son identifiant en tant que personne, le numéro de passeport sera lui stocké dans la table "Client ".
- Le client peut réserver plusieurs places pour plusieurs vols donnés, cela sera stocké dans la table "Réservation " ou les identifiants du vol, de la place et du client seront stockés, en plus de la date de réservation ainsi que le numéro de réservation généré.

- Le cumul des heures de vol d'un client seront stocké dans la table du client, et actualisé automatiquement à l'aide d'un trigger lorsque le vol concerné sera marqué comme terminé. Les bons de réductions seront stockés dans une table à part pour pouvoir les cumuler et un champ "utilise " permettra de savoir s'il a déjà été utilisé.

- Le fait qu'il faille vérifier que le nombre de pilotes et d'hôtesse soit respecté se déroulera du coté application.

- La mise à jour des heures de vol des pilotes et des hôtesses dès que le vol est terminé sera géré par un trigger déclenché lorsque le vol sera terminé.

- La pause d'une durée de la moitié du vol sera gérée niveau application : l'affectation sera impossible si le délai n'est pas respecté.

- De même pour la ville d'affectation d'un membre du personnel de l'avion, si le départ de l'avion n'est pas au même endroit que l'aéroport d'arrivée du dernier avion pris par le membre du personnel, l'application signifiera une erreur.

Modèle:

https://www.lucidchart.com/documents/edit/a21ec5c2-acea-4779-be3f-73c7ae2d7280/0_0?shared=true

Explications:

Lors du rendu de notre pré-rapport du projet, nous avons eu un comme retour que nos contraintes sont bien précisées.Cependant, au niveau de la gestion du vol en termes de capacité et rayon d'action nous avons utilisés une table place dans laquelle on a les numéros de places disponibles permettant la gestion.

III- Gestion de Projet

Dans la réalisation de notre projet nous avons en place un patron DAO qui permet de faire la liaison entre une couche d'accès aux données et une couche métier permettant de manipuler des données. Après nous avons établit la connexion avec la base de données.Nous avons ensuite créer des triggers afin de permettre de gérer certaines mis à jour automatique d'attribut.

Un pseudo ORM a été mis en place pour faciliter la gestion des objets.
Il est presque générique bien qu'il à dû nécessité des spécificité pour gérer l'héritage et des classe particulière.
On utilise les réflections du langage java pour traiter la liste des attributs comme un tableau, auquel on attribue les valeurs que la base de données nous a renvoyé.
On dispose donc de fonction comme .save() qui permet de sauvegarder un objet dans la base de données et .findOne(int id) qui nous renvoie un objet déjà construit à partir de la base de donnés. Tout ce passe dans le fichier DAOHelper.java, il est un peu dense car il nécessite beaucoup de cas spéciaux, mais si vous avez une question nous restons à disposition pour y répondre .

-un trigger permettant de gérer chaque fois l'ajout d'un vol.

```
CREATE FUNCTION public.ajoutvol(numvol character varying, dep character varying, arr
character varying, hdep timestamp without time zone, distance integer, numavion integer,
duree integer) RETURNS integer
  LANGUAGE plpgsql
  AS $$
DECLARE
  idV integer;
  ret integer := 0;
BEGIN
  idV = CAST(CAST(numvol AS text)||CAST(nextval('seq_vol') AS text) AS integer);
  insert into Vol(idvol, numvol, aeroportdepart, aeroportarrivee, horairedepart, distance,
numavion, termine, duree)
  values(idV, numvol, dep, arr, hdep, distance, numavion, false, duree);

  -- Test si fonctionne (affecte 1 a la valeur de retour si le nouvel id est bien dans la base
  select 1 into ret where idV in(select idvol from vol);

  return ret;
END;
$$;
CREATE TRIGGER actuvolclient AFTER UPDATE ON public.vol FOR EACH ROW
EXECUTE PROCEDURE public.actuvolclient();
```

- un trigger permettant de gérer le nombre d'heure totale effectuée par le personnel

```
CREATE FUNCTION public.actuheurepers() RETURNS trigger
  LANGUAGE plpgsql
  AS $$
DECLARE
  listePers record;
BEGIN
  if old.termine=false and new.termine=true then
     for listePers in select * from VolPersonnel v where v.numvol = new.numvol LOOP
        update Personnel p
        set nbHeureCumul = nbHeureCumul + new.duree
        where p.numPersonnel = ligne.numPersonnel;
     END LOOP;
  END IF;
  return new;
END
$$;
-
```

```
CREATE TRIGGER actuheurepers AFTER UPDATE ON public.vol FOR EACH ROW
EXECUTE PROCEDURE public.actuheurepers();
```

Ensuite nous avons mis en place une interface qui permet de gérer en fonction de la saisie d'un utilisateur de pouvoir réserver un vol, modifier une réservation etc…. . A chaque fois que l'utilisateur rentre le numéro qui n'est pas dans l'intervalle, ce dernier sera appelé à ressaisir le bon numéro.

IV- Protocoles des Tests

Au niveau des protocoles de test nous avons eu à créer des fichiers sql qui nous permet à chaque fois d'effectuer des tests. Au fur et mesure de l'avancement de notre projet on s'assure du fait que nos jeux de test marche.

V-Conclusion

Ainsi le projet nous a permis de mieux comprendre les différents rôles des technologies notamment JDBC et Java pour la partie applicative et Oracle pour la partie de la donnée. Cependant au niveau de l'architecture logicielle, le patron composant DAO nous a permis durant la réalisation de notre projet de comprendre la liaison entre la couche métier à l'action aux données. Les notions acquises en système d'information nous ont beaucoup aidé dans le projet avec la réalisation des diagrammes UML permettant de comprendre les différentes dépendances de nos modèles relationnels.