NGUYEN Duc Hau
ROBERT Quentin

BDD : Travail 2

Table des matières

Introduction :

Dans ce tp nous allons écrire la base de données correspondant au tp précédent, puis y ajouter des données, et enfin faire des requêtes.

Create Table

La création des tables se trouve dans le fichier **ddl.sql**. Dans un souci de lisibilité nous avons renommé les contraintes, et mis ces dernières en fin de fichiers.

Pour le développement, on a mis DROP TABLE en premier pour faciliter les mises à jours notre requête.

On rapporte dans cette partie l'explication globale de notre choix des contraintes:

- Lors de la création des tables, seules les NOT NULL et des clés primaires sont définies.
- Les clés étrangères sont déclarée lors que tous les tables sont initialisées afin d'éviter l'erreur qu'un table référençant à une non existante.
- Les not null sont mis pour les champs qu'on suppose descriptifs pour le système. Par exemple le nom de pays, type de personne, etc. Les informations concernant client sont facultatives.
- Pour d'autre contrainte suivie par l'explication:

```
ALTER TABLE Vol ADD CONSTRAINT Vol_dateorder_ck  CHECK (DEPART <  ARRIVEE);
ALTER  TABLE  Infos_Escale  ADD  CONSTRAINT  Infos_escale_dateorder_ck  CHECK
(DEPART <  ARRIVEE);
```

Assurer que la date de départ soit avant la date d'arrivée.

```
ALTER TABLE Compte ADD CONSTRAINT Compte_courriel_syntax_ck CHECK (COURRIEL
LIKE '%_@__%._%');
```

Assurer la bonne syntax d'un adresse mail

```
ALTER TABLE Passport ADD CONSTRAINT Passport_numero_syntax_ck CHECK (NUMERO
LIKE '%[^A-Z0-9]%');
ALTER  TABLE  Adresse  ADD  CONSTRAINT  Adress_codepostal_syntax_ck  CHECK
(CODE_POSTAL LIKE '%[^A-Z0-9]%');
```

Numéro de passeport et code postal ne contient pas des caractères spéciaux.

```
ALTER TABLE Place ADD CONSTRAINT Place_price_positive_ck CHECK (PRIX > 0);
```

Le prix ne peut pas être négatif.

```
ALTER TABLE Infos_Place ADD CONSTRAINT Infos_place_rangee_ck CHECK (RANGEE
LIKE '[A-H]');
ALTER TABLE Infos_Place ADD CONSTRAINT Infos_place_siege_ck CHECK (SIEGE >
0);
```

Par la description de Travail TP1, les rangée sont de A à H et les siège ne sont que les numéro (pas de signe négative)

Insert Values

Pour pouvoir tester les requêtes que nous avons eu à faire il est important d'avoir un jeu de données correspondant à la requête. Pour cela il suffit de regarder ces dernières et d'établir les besoins. Par exemple tout ce qui est en relation avec les numéros de téléphone n'est pas utilisé, donc pas de besoin spécifique de ce côté-là. Alors qu'en revanche les requêtes indiquent des villes précises, qu'il faut donc ajouter.
Encore une fois il existe des logiciels qui peuvent aider, en créant des jeux de données, auquel il suffit d'ajouter les spécifiques, tel que les noms de villes.

Requête 1

➔ Quels sont tous les vols ouverts de Montréal à Paris entre le 25 décembre 2017 et le 12 janvier 2018 ?

```
SELECT numero, depart
FROM vol, aeroport a1, aeroport a2, ville_desservie v1, ville_desservie v2,
ville vi1, ville vi2
WHERE vol.depart>='2017/12/25' AND vol.depart<='2018/01/12'
and a1.id=vol.aeroport_depart and a2.id=vol.aeroport_arrive
and v1.aeroport = a1.id and v2.aeroport = a2.id and v1.ville=
vi1.id and v2.ville=vi2.id and vi1.nom='Montréal' and vi2.nom='Paris';
```

Requête qui est assez basique. Il faut en revanche de multiples jointures pour obtenir les aéroports correspondants aux villes de départ et d'arrivé. Le problème de la date peut se résoudre avec des comparateurs numériques, ou encore avec between. On choisit d'afficher le numéro du vol, ainsi que la date de départ, afin que le retour donne quelques informations basiques à l'utilisation. Sans quoi on va probablement chercher par la suite la date… Mais dans l'absolu le numéro seul aurait suffi, afficher l'id n'est pas pertinent pour l'utilisateur, puisque c'est une valeur utile à la bdd mais qui ne sert pas à grand-chose pour un humain…

Requête 2

➔ Quels sont les vols ouverts et sans escales entre Vancouver et New York dont la date de départ est avant le 1 janvier 2018 ?

```
SELECT numero, depart
FROM vol, aeroport a1, aeroport a2, ville_desservie v1, ville_desservie v2,
ville vi1, ville vi2, Statut
WHERE vol.depart<'2018/01/01'
and a1.id=vol.aeroport_depart and a2.id=vol.aeroport_arrive
and v1.aeroport = a1.id and v2.aeroport = a2.id and v1.ville=vi1.id and
v2.ville=vi2.id and ((vi1.nom='Vancouver' and vi2.nom='New York City') or
(vi1.nom='New York City' and vi2.nom='Vancouver'))
and vol.id not in(Select vol from Escale) and statut.id = vol.statut and
statut.libelle = 'Ouvert Reservation';
```

Deux 'difficultés' : l'utilisation des and et or afin de permettre que les villes soient en départ ou en arrivé, et vérifié qu'un vol n'a pas d'escale. Ici nous avons utilisé l'opérateur not in. La sous-requête n'étant pas synchronisé le sgbd ne va calculé la liste des vol avec escale qu'une fois, et il vérifie simplement ensuite que l'id du vol n'est pas dedans. La solution est donc assez performante niveau temps. On aurait aussi pu utiliser un Minus ou une jointure gauche.

Requête 3

➔ Quels sont les vols ouverts, du moins cher au plus cher, pour lesquels il est possible de réserver une place en classe business entre New York et Montréal ?

```
SELECT numero, compagnie_aerienne.nom, depart, P.prix
FROM vol, aeroport a1, aeroport a2, compagnie_aerienne,
ville_desservie v1, ville_desservie v2,
ville vi1, ville vi2, PLACE P, Classe c, infos_place i, statut s
WHERE a1.id=vol.aeroport_depart and a2.id=vol.aeroport_arrive
and v1.aeroport = a1.id and v2.aeroport = a2.id and v1.ville=
vi1.id and v2.ville=vi2.id and ((vi1.nom='Montréal' and vi2.nom='New York
City') or (vi1.nom='New York City' and vi2.nom='Montréal'))
and s.id = vol.statut and s.libelle='Ouvert Reservation' and
P.vol = vol.id and P.classe = C.id and C.libelle = 'Business'
and vol.compagnie=compagnie_aerienne.id
and P.place = i.id and i.disponible = 'T'
ORDER BY P.prix;
```

Ici on vérifie que la classe business est disponible pour au moins un siège par vol entre les deux villes. C'est ce qui explique le nombre de table impliquées dans la requête. L'élément exclusif de cette requête est la clause order by, pour trier les valeurs de retour comme souhaité, par défaut il s'agit d'un ordre croissant, donc pas besoin de préciser 'ASC'. On affiche également la compagnie, qui est à prendre en compte pour le client si il a des réductions ou autre.

Requête 4

➔ Quels sont les clients qui ont réservé un vol entre Paris et New York en première classe depuis 2015 ?

```
CREATE TABLE Reservation_Place (
    VOL         INTEGER,
    CLASSE      INTEGER,
    PLACE       INTEGER,
    RESERVATION INTEGER,

    CONSTRAINT Reservation_Place_pk PRIMARY KEY (VOL, CLASSE, PLACE,
RESERVATION)
);

ALTER TABLE Reservation_Place ADD CONSTRAINT Reservation_Place_Place_fk
FOREIGN KEY (VOL, CLASSE, PLACE)  REFERENCES Place(VOL, CLASSE, PLACE);
ALTER TABLE Reservation_Place ADD CONSTRAINT
Reservation_Place_Reservation_fk FOREIGN KEY (RESERVATION)
REFERENCES Reservation(ID);

INSERT INTO Reservation_Place VALUES(16, 2, 3, 3);
INSERT INTO Reservation_Place VALUES(16, 1, 2, 4);

SELECT P.nom, P.prenom, pass.numero
FROM vol, aeroport a1, aeroport a2, ville_desservie v1, ville_desservie v2,
ville vi1, ville vi2, Place pl, Reservation_Place rp, Reservation r,
Reservation_Personne rpp,
Personne p, passport pass, Classe c, type_personne tpp
WHERE r.date>='2015/01/01' and pl.vol=vol.id and pl.classe = c.id and
c.libelle='Première classe'
and rp.vol=pl.vol and pl.classe = rp.classe and rp.reservation = r.id and
rpp.reservation=r.id
and rpp.personne = p.id and pass.id = p.passeport
and a1.id=vol.aeroport_depart and a2.id=vol.aeroport_arrive
and v1.aeroport = a1.id and v2.aeroport = a2.id and v1.ville=
vi1.id and v2.ville=vi2.id and (vi1.nom='Paris' and vi2.nom='New York City')
or (vi1.nom='New York City' and vi2.nom='Paris') and tpp.type='Client' and
tpp.id=p.type_personne;

DROP TABLE Reservation_Place;
```

La question du sujet n'est en réalité par réalisable avec la bdd que nous utilisons, en effet si on garde bien le vol correspondant à une réservation on ne garde pas du tout la place correspondant à la réservation, et on ne sait donc pas quelle est la classe de la réservation, sauf s'il n'y qu'une seule classe sur le vol... Ce qui n'est pas très pertinent dans la réalité.
Une solution simple serait d'employer une classe reservation_place, similaire à réservation_vol qui serait d'ailleurs à ce moment supprimé car redondante. Une fois la modification réalisé la requête est de fonctionnement similaire aux précédente,

sans difficultés à noter. On affiche finalement le nom et le prénom de la personne, ainsi que son numéro de passeport, afin d'identifié la personne de manière unique, tout en gardant une information simple à traiter par un humain.

Requête 5

➔ Combien de places sont déjà réservées pour le vol numéro AC1520 à destination de La Havane le 25 décembre 2017 ?

```
SELECT numero, COUNT(RS.VOL) AS Solution
FROM vol, aeroport a2, ville_desservie v2, ville vi2,
reservation_vol RS
WHERE vol.depart='2017/12/25' AND vol.numero='AC1520'
and a2.id=vol.aeroport_arrive
and v2.aeroport = a2.id and v2.ville=vi2.id
and vi2.nom='La Havane' and RS.vol = vol.id
GROUP BY numero;
```

Ici le point à souligner est l'utilisation de l'opérateur COUNT. De par le fait que l'on affiche également le numéro du vol, pour confirmer que c'est le bon vol, il faut utiliser Group By sur tous les attributs du select qui ne sont pas dans le COUNT, uniquement numero dans notre cas.

Requête 6

➔ Quels sont les clients qui ont une adresse aux USA ?

```
SELECT P.nom, P.prenom, pass.numero
FROM pays pa, Province_Pays pp, Personne P, Passport pass, adresse_Personne
ap, adresse a, Type_Personne tp
WHERE P.passeport = pass.id and p.id = ap.personne and ap.adresse
=a.id and a.pays = pp.pays and a.province=pp.province
and pp.pays = pa.id and pa.nom='USA' and tp.id = P.type_personne and
tp.type='Client'
GROUP BY P.nom, P.prenom, pass.numero;
```

Ici rien de compliquer, il s'agit de simples jointures. En revanche on utilise le Group by pour ne pas avoir plusieurs fois le même client qui s'affiche s'il a plusieurs adresses aux USA.

Requête 7

➔ Quel est le prix et la classe de la place la plus chère à bord du vol AF2306 du 12 février 2018 ?

```
SELECT prix, libelle
FROM vol, PLACE P, Classe c
WHERE P.vol = vol.id and P.classe = C.id
and vol.numero = 'AF2306' and vol.depart='2018/02/12'
and prix = (Select max(prix) from vol, place
            where vol.id = place.vol and vol.numero='AF2306');
```

Pour cette requête on applique la même structure que pour la requête numéro 2. Sauf que cette fois la sous-requête calcul un maximum. L'avantage est qu'ainsi si plusieurs places ont le prix le plus chère elles seront toutes retournées.

Requête 8

➔ Quels sont tous les clients de France ayant une réservation sur un vol pour Paris ?

```
SELECT P.nom, prenom, pass.numero
FROM vol, aeroport a2, ville_desservie v2, ville vi2,
Reservation_Personne rp, Reservation r, reservation_vol rv, type_personne tp,
pays pa, Province_Pays pp, Personne P, Passport pass, adresse_Personne ap,
adresse a
WHERE P.passeport = pass.id and p.id = ap.personne and ap.adresse=a.id
and a.pays = pp.pays and a.province=pp.province
and pp.pays = pa.id and pa.nom='France'
and rp.personne = p.id and rp.reservation = r.id
and r.id = rv.reservation and rv.vol = vol.id
and a2.id=vol.aeroport_arrive and vol.AEROPORT_ARRIVE=a2.id
and v2.aeroport = a2.id and v2.ville=vi2.id
and vi2.nom='Paris' and tp.id = P.type_personne and tp.type='Client'
GROUP BY nom, prenom, pass.numero;
```

On part du principe qu'un client de France est un client ayant au moins une adresse en France. Cette requête est une suite de jointure qui 'synthétise' la requête 4 et 6. Aucune difficulté nouvelle à signaler.

Requête 9

➔ Quels sont tous les aéroports du Canada ?

```
SELECT CODE_IATA, ae.nom
FROM aeroport ae, Ville_desservie vd, ville v, pays p
WHERE ae.id = vd.aeroport AND vd.ville=v.id AND v.Pays=P.id
and p.nom='Canada';
```

Ici une simple jointure. On choisit d'afficher le nom de l'aéroport ainsi que son code afin que le retour soit facilement compréhensible et réutilisable si besoin.

Requête 10

➔ Quels sont les compagnies aériennes qui proposent des vols avec le moins d'escales ?

```
SELECT c.nom, vv.numero, SUM((Select count(*) from escale e, vol v where
vv.id=v.id and e.vol=v.id and c.id=v.compagnie)) as nbEscale
FROM Compagnie_Aerienne c, Vol vv
WHERE c.id=vv.compagnie
GROUP BY c.nom, vv.numero
ORDER BY nbEscale
```

Requête un peu plus compliquée que les précédentes, en effet il faut pouvoir retourner le nombre d'escale par vol… Or il y a des vols sans escale, ce qui veut dire qu'une simple jointure n'est pas suffisante. Nous avons employé une sous requête synchronisé pour y répondre, qui dans le cadre de notre petite base de données n'est pas un problème niveau performance, et a l'avantage d'être simple à comprendre. Dans le cas où on serait en recherche de performance on envisager une jointure droite.

Conclusion

Travail qui permet de bien prendre en main le sql. Il peut sembler un peu fastidieux par les nombreuses jointures des requêtes mais ce n'est pas vraiment un problème en étant rigoureux lors de leurs rédactions. Toutes nos requêtes sont, après tests, fonctionnelles, et retournent des valeurs cohérentes vis-à-vis de notre bdd.