Université de Lille

TP 3
Le DML sous PostgreSQL

CORRECTION + PÉDAGOGIE

M3104 - WEB-BDD
DUT Info – N3
Philippe Mathieu
2019–2020

Objectifs

Savoir créer et manipuler les structures de tables en SQL. Appréhender les vues et leur manipulation.

1 Création de tables

Afin de nous entraîner à la création de tables, recopiez le script `tp02_tables.sql` précédemment étudié en un fichier `tp03_tables.sql` que l'on modifiera et complètera avec toutes les réponses de ce sujet. Ce fichier doit à tout moment pouvoir etre relancé complètement.

avion(ano, type, places, compagnie)
pilote(pno, nom, prenom, adresse)
ligne(lno, depart, arrivee)
vol(ano, pno, lno, hdep, harr)

1. Créez les 4 tables citées, en appliquant les contraintes suivantes :
   - les clés des tables issues des entités doivent être gérées par des numéros automatiques `serial`.
   - Les avions ont au minimum 100 places et au maximum 500 places.
   - L'adresse par défaut d'un pilote sera Lille.
   - La ville de départ doit être différente de la ville d'arrivée
   - L'heure d'arrivée est toujours postérieure d'au minimum 1/2h à l'heure de départ
   - Les clés étrangères seront en `cascade` pour les mises à jour et `restrict` pour les effacements.

```
-------------------
-- Suppression des différents objets
-- ATTENTION : l'ordre est important
-------------------

DROP TABLE IF EXISTS vol;
DROP TABLE IF EXISTS avion CASCADE;
DROP TABLE IF EXISTS pilote CASCADE;
DROP TABLE IF EXISTS ligne CASCADE;

-------------------
-- La table AVION
-------------------

CREATE TABLE avion
(
    ano serial,
    TYPE text,
    places integer CHECK (places BETWEEN 100 AND 500),
    compagnie text,
    CONSTRAINT pk_avion PRIMARY KEY(ano)
);

-------------------
-- La table PILOTE
-------------------

CREATE TABLE pilote
(
    pno serial,
    nom text,
    prenom text,
    adresse text DEFAULT 'Lille',
    CONSTRAINT pk_pilote PRIMARY KEY(pno)
);

-------------------
-- La table LIGNE
-------------------

CREATE TABLE ligne
(
    lno serial,
    depart text,
    arrivee text CHECK (arrivee <> depart),
    CONSTRAINT pk_ligne PRIMARY KEY(lno)
);

-------------------
-- La table VOL
-------------------

CREATE TABLE vol
(
    ano integer,
    pno integer,
    lno integer,
    hdep time,
    harr time CHECK (harr > hdep + '00:30:00') ,
    CONSTRAINT pk_vol PRIMARY KEY(ano,pno,lno),
    CONSTRAINT fk_avion FOREIGN KEY(ano) REFERENCES avion(ano)
                        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_pilote FOREIGN KEY(pno) REFERENCES pilote(pno)
                        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_ligne FOREIGN KEY(lno) REFERENCES ligne(lno)
                        ON DELETE RESTRICT ON UPDATE CASCADE
);
```

2. Insérez un pilote, un avion, une ligne puis un vol concernant ce pilote et cet avion dans les tables.

```
INSERT INTO avion(TYPE,places,compagnie) VALUES('A320',250,'Air France');
INSERT INTO avion(TYPE,places,compagnie) VALUES('A320',250,'Air Liberte');
INSERT INTO avion(TYPE,places,compagnie) VALUES('A350',400,'British Airways');

INSERT INTO pilote(nom,prenom,adresse) VALUES('Durand','Pierre','Lille');
INSERT INTO pilote(nom,prenom,adresse) VALUES('Lefebvre','Jean','Douai');
INSERT INTO pilote(nom,prenom,adresse) VALUES('Dupond','Louis','Lens');

INSERT INTO ligne(depart,arrivee) VALUES('Lille','Ajaccio');
INSERT INTO ligne(depart,arrivee) VALUES('Ajaccio','Toulouse');
INSERT INTO ligne(depart,arrivee) VALUES('Toulouse','Lille');
```

3. Tentez d'insérer un nouveau pilote avec le même identifiant. Est-ce accepté ?

```
impossible ... C'est une clé, donc UNIQUE
```

4. Modifiez le numéro d'identifiant du pilote? Que se passe t-il ?

```
ça se modifie en CASCADE dans vol
```

5. Supprimez l'avion. Que se passe t-il ?

```
Impossible de supprimer si l'avion est référencé dans vol
```

6. Créez en une seule commande SQL une table temporaire `copievol` qui contient l'ensemble des données de la table vol.

```
CREATE temp TABLE copievol
   AS SELECT * FROM vol;
```

7. Effacez le contenu de la table vol.

```
DELETE FROM vol;
```

8. Assurez vous maintetant d'avoir 3 pilotes, 3 lignes et 3 avions. Remplir la table vol avec toutes les manières possibles d'associer un pilote, un avion et une ligne (il y en a donc 27), en mettant pour chaque vol une heure de dep à 12:00 et une heure d'arrivée à 14:00 , le tout en une seule requête.

```
INSERT INTO vol
       SELECT ano,pno,lno,'12:00','14:00' FROM avion, pilote,ligne ;
```

9. Comment afficher une table (vol par ex) avec les lignes mélangées ?

```
SELECT * FROM vol ORDER BY random() ;
```

10. Comment afficher approximativement 20% des lignes d'une table (vol par ex) prises au hasard.

```
SELECT * FROM vol WHERE random()<0.2 ;

et si on souhaite 20% exactement

SELECT * FROM vol ORDER BY random() LIMIT (SELECT COUNT(*)*0.2  FROM vol);
```

2 Modification de structures

1. Ajoutez une colonne `couleur` à la table `avion`

```
ALTER TABLE avion ADD couleur int;
```

2. Ajoutez quelques couleurs à certains avions par l'ordre `update`

```
UPDATE avion SET couleur='rouge' WHERE TYPE='A320';
UPDATE avion SET couler='bleu' WHERE TYPE='A350';
```

3. Ajoutez une colonne `email` à la table `pilote`
4. Supprimez la colonne `couleur` précédemment ajoutée. Que se passe t-il pour les données de cette colonne ?

```
ALTER TABLE avion DROP couleur;

Il ne se passe rien de particulier, les données sont perdues
```

3 Non redondance

Dans la table ligne, les aeroports sont codés par une simple chaine de caractère. On souhaite maintenant stocker un peu plus d'information concernant les aeroports.

1. Créez une nouvelle table `aeroport(ano,ville,IATA)` avec comme clé un numéro automatique

```
CREATE TABLE aeroport
(ano serial PRIMARY KEY,
 ville text,
 IATA text UNIQUE
);
```

2. Insérez en une seule requete SQL l'ensemble des aéroports existant dans la table `ligne` dans cette table `aeroport` (en mettant l'IATA à null). Assurez vous qu'il n'y ait pas de doublon.

```
INSERT INTO aeroport(ville)
    SELECT depart FROM ligne
    UNION
    SELECT arrivee FROM ligne;
```

3. Donnez la suite des ordres SQL permettant de transformer (donc sans la détruire) la table ligne pour que les deux colonnes depart et arrivee contiennent la bonne clé issue de la table aeroport

```
-- on ajoute deux colonnes pour les numeros
ALTER TABLE ligne ADD dano int;
ALTER TABLE ligne ADD aano int;
-- on va chercher les numeros dans Aeroport
UPDATE ligne AS l
   SET dano = (SELECT ano FROM aeroport AS a WHERE a.ville=l.depart) ,
       aano = (SELECT ano FROM aeroport AS a WHERE a.ville=l.arrivee) ;
-- on peut virer les anciens textes
ALTER TABLE ligne DROP depart;
ALTER TABLE ligne DROP arrivee;
-- et renommer les colonnes pour etre comme avant
ALTER TABLE ligne RENAME dano TO depart;
ALTER TABLE ligne RENAME aano TO arrivee;
```

4. Ecrire la requête qui affiche les lignes avec les aéroports en clair (3 colonnes), comme auparavant.

```
SELECT  lno, a1.ville AS depart , a2.ville AS arrivee
FROM ligne AS l , aeroport AS a1, aeroport AS a2
WHERE l.depart=a1.ano AND l.arrivee=a2.ano ;
```

4 Création de vues

1. Créez une vue `petitavion` qui contient les informations sur les avions qui ont entre 100 et 200 places

2. Effectuez une sélection sur cette vue.
3. Créez une vue `volclair` qui affiche les informations sur les vols en y ajoutant les noms et prénoms des pilotes ainsi que les types et compagnies des avions correspondants et les départs et arrivée de la ligne (11 colonnes)

```
-------------------
-- La vue VOLCLAIR
-------------------

CREATE OR REPLACE VIEW volclair
       AS SELECT v.ano, TYPE, v.pno, nom, prenom,
                 v.lno, depart, hdep, arrivee, harr
          FROM avion a, pilote p , vol v, ligne l
          WHERE v.ano=a.ano AND v.pno=p.pno AND v.lno=l.lno;
```

4. Effectuez une sélection sur cette vue.

```
Tout se fait comme si C'était une TABLE :
SELECT * FROM volclair;
```

5. Testez les ordres `insert`, `update` et `delete` sur cette vue. Sont-ils possibles ?
6. Créez une `rule` permettant de faire un `delete` sur `volclair` en effaçant le vol correspondant de la table `vol`

```
CREATE RULE r1 AS ON DELETE TO volclair
DO INSTEAD DELETE FROM vol AS v
           WHERE v.ano=OLD.ano AND v.pno=OLD.pno AND v.lno=OLD.lno;
```

7. La table vol est considérée stratégique. Créez une table `espion(nom,dat,action)` permettant de stocker les informations d'insertion ou d'effacement sur la table `vol`

```
CREATE TABLE espion(nom text, dat TIMESTAMP, action text);
CREATE RULE r2 AS ON INSERT TO vol
DO also INSERT INTO espion VALUES(USER,CURRENT_TIMESTAMP,'insert');
CREATE RULE r3 AS ON DELETE TO vol
DO also INSERT INTO espion VALUES(USER,CURRENT_TIMESTAMP,'delete');
```