TP-1 : Introduction (rappel) au langage SQL avec PostgreSQL

Exercice 1 : Mes premières requêtes et création de tables

Accédez à l'interface d'édition des requêtes, dans un premier temps nous allons créer des tables dans la base de données portant votre nom. Nous n'utiliserons pas l'interface graphique pour créer les tables mais uniquement l'outil d'édition de scripts.

Dans cet exercice nous allons créer une base de données jouet de vendeurs de voitures, chaque vendeur (concessionnaire) dispose de plusieurs modèles de voitures avec éventuellement plusieurs fois le même modèle. Avant de commencer, il vous faudra créer un script en cliquant du bouton droit sur votre schémas de la base de données BDA (en local le schéma public, sur la plateforme le schéma correspondant à votre nom). Au début de votre script (si vous n'êtes pas sur le schéma public), vous pouvez ajouter l'instruction suivante :

```
SET search_path TO mon_schema;
```

Question 1 :

Créez la relation `modele` : modele (*m_id* : **`int`**, *m_constructeur* : **`str`**, *m_type* : **`str`**, *m_version* : **`str`**) Qui contient les informations sur les modèles des voitures.

Solution :

```
-- On spécifie le schéma sur lequel on travaille
SET search_path TO t_gerald;
-- On détruit les tables si elles existent
DROP TABLE IF EXISTS concessionaire;
DROP TABLE IF EXISTS modele;

CREATE TABLE modele(
  m_id SERIAL,
  m_constructeur VARCHAR(255),
  m_type VARCHAR(255),
  m_version VARCHAR(255),
  CONSTRAINT m_pk PRIMARY KEY(m_id)
);
```

Question 2 :

Ajoutez les données suivantes dans la table

Modèle			
vid	constructeur	type	version
1	Toyota	Yaris	CY-2018
2	Toyota	Prius	CP-2005
3	Toyota	Prius	CP-2012
4	Renault	Twingo	RM-25A8

Solution :

```
INSERT INTO modele (constructeur, type, version) VALUES
  ('Toyota', 'Yaris', 'CY-2018'),
  ('Toyota', 'Prius', 'CP-2005'),
  ('Toyota', 'Prius', 'CP-2012'),
  ('Renault', 'Twingo', 'RM-25A8');
```

Question 3 :

Si vous ajoutez l'enregistrement suivant : (5, Renault, Twingo , RM-25A8) Que se passe t-il ? Comment l'éviter ? Supprimez si nécessaire l'élément et **ajoutez** la contrainte adéquate !

Solution : L'élément sera ajouter, nous aurons donc deux enregistrements identique. Il faudrait donc ajouter une contrainte d'unicité sur le triplet (constructeur,modele, version)

```
INSERT INTO modele (constructeur, type, version) VALUES
    ('Renault', 'Twingo', 'RM-25A8');
DELETE FROM modele WHERE m_id=5;
ALTER TABLE modele ADD CONSTRAINT u_constructeur_type_version
    UNIQUE(constructeur, type, version);
INSERT INTO modele (constructeur, type, version) VALUES
    ('Renault', 'Twingo', 'RM-25A8'); -- Ne devrait pas fonctionner
```

Question 4 :

Créer la relation `concessionnaire` suivante :
concessionnaire (*c_id* : **`int`**, *c_nom* : **`string`**, *c_note* : **`float`**, *c_adresse* : **`str`**)

Solution :

```
CREATE TABLE concessionnaire (
  c_id INTEGER,
  c_nom VARCHAR(64),
  c_note FLOAT,
  c_adresse VARCHAR(128)
);
```

Question 5 :

Ajoutez les éléments afin d'obtenir la table suivante :

Concessionnaire			
id	nom	note	adresse
1	Auto-2000	4	155 avenue du général Leclerc, Orsay
2	Renault-Palaiseau	4,5	48 rue Gambetta, Palaiseau
3	Auto-moto	3	1 Avenue Jean Moulin, Gif-sur-Yvette

Solution :

```
INSERT INTO concessionnaire (c_id, nom, note, adresse) VALUES
    (1, 'Auto-2000', 4, '155 avenue du général Leclerc, Orsay'),
    (2, 'Renault-Palaiseau', 4.5, '48 rue Gambetta, Palaiseau'),
    (3, 'Auto-moto', 3, '1 Avenue Jean Moulin, Gif-sur-Yvette');
```

Question 6 :

Comment définir la table concessionnaire pour que la requête suivante :

```
INSERT INTO concessionnaire (nom, adresse)
   VALUES ('Toyota+', '13 rue des champs, Massy')
```

ajoute l'enregistrement (4, 'Toyota+', .0, '13 rue des champs, Massy') et que les requêtes suivantes :

```
INSERT INTO concessionnaire (c_nom, c_note, c_adresse)
   VALUES ('Toyota+', 5.1, '13 rue des champs, Massy');
INSERT INTO concessionnaire (c_nom, c_note, c_adresse)
   VALUES ('Toyota+', -1, '13 rue des champs, Massy');
INSERT INTO concessionnaire (c_nom, c_adresse)
   VALUES ('Auto-2000', '155 avenue du général Leclerc, Orsay');
INSERT INTO concessionnaire (c_id, c_nom, c_adresse)
   VALUES (1, 'Toyota++', '54 rue des champs, Massy');
```

Retournent des erreurs. **NB : Si vous avez redéfini la table n'oubliez pas d'ajouter les enregistrements de la question précédente**

Solution : 1. Contrainte de valeur par défaut + serial pour c_id 2. Contrainte de valeur maximum (CHECK) 3. Contrainte de valeur minimum 4. Contrainte d'unicité (c_nom, c_adresse) ou adresse seulement (à eux de choisir) 5 Contrainte de clef primaire ou unicité sur c_id Si la correction est faite au tableau commenter la solution 2 ci-dessous

```
INSERT INTO concessionnaire (c_nom, c_adresse)
   VALUES ('Toyota+', '13 rue des champs, Massy');
SELECT * FROM concessionnaire; -- Ajoute (null, 'Toyota+',
   -- null, '13 rue des champs, Massy')
INSERT INTO concessionnaire (c_nom, note, c_adresse)
   VALUES ('Toyota+', 5.1, '13 rue des champs, Massy');
INSERT INTO concessionnaire (c_nom, note, c_adresse)
   VALUES ('Toyota+', -1, '13 rue des champs, Massy');
INSERT INTO concessionnaire (c_nom, c_adresse)
   VALUES ('Auto-2000 ', '155 avenue du général Leclerc, Orsay');
INSERT INTO concessionnaire (c_id, c_nom, c_adresse)
   VALUES (1, 'Toyota++', '54 rue des champs, Massy')
SELECT * FROM concessionnaire; -- les requêtes
   -- fonctionnent (donc il faut ajouter des contraintes)
-- On supprime
SELECT ctid FROM concessionnaire OFFSET 3 ;
DELETE FROM concessionnaire
   WHERE ctid IN (SELECT ctid FROM concessionnaire OFFSET 3 );
SELECT * FROM concessionnaire;

------ SOLUTION 1
-- On ajoute les contraintes
ALTER TABLE concessionnaire ALTER COLUMN note SET DEFAULT 0.;
ALTER TABLE concessionnaire DROP COLUMN c_id;
ALTER TABLE concessionnaire ADD COLUMN c_id SERIAL;
SELECT * FROM concessionnaire;
ALTER TABLE concessionnaire ADD CONSTRAINT c_note
   CHECK(note >= 0 and note <= 5);
ALTER TABLE concessionnaire ADD CONSTRAINT u_adresse
   UNIQUE (c_adresse);
ALTER TABLE concessionnaire ADD CONSTRAINT pk_concessionnaire
   PRIMARY KEY(c_id);
-- ON re-test
INSERT INTO concessionnaire (c_nom, c_adresse)
   VALUES ('Toyota+', '13 rue des champs, Massy'); -- OK
SELECT * FROM concessionnaire; -- Ajoute (null, 'Toyota+',
   -- null, '13 rue des champs, Massy') -- OK
INSERT INTO concessionnaire (c_nom, note, c_adresse)
   VALUES ('Toyota+', 5.1, '13 rue des champs, Massy'); -- KO
INSERT INTO concessionnaire (c_nom, note, c_adresse)
   VALUES ('Toyota+', -1, '13 rue des champs, Massy'); -- KO
INSERT INTO concessionnaire (c_nom, c_adresse)
   VALUES ('Auto-2000 ', '155 avenue du général Leclerc, Orsay'); -- KO
INSERT INTO concessionnaire (c_id, c_nom, c_adresse)
   VALUES (1, 'Toyota++', '54 rue des champs, Massy'); --KO
```

```sql
----- SOLUTION 2
DROP TABLE IF EXISTS concessionnaire;
CREATE TABLE concessionnaire (
  c_id SERIAL,
  c_nom VARCHAR(64),
  c_note FLOAT DEFAULT 0.,
  c_adresse VARCHAR(128),
  CONSTRAINT c_note CHECK(c_note >= 0 and c_note <= 5),
  CONSTRAINT u_adresse UNIQUE (c_adresse),
  CONSTRAINT pk_concessionnaire PRIMARY KEY(c_id)
);
INSERT INTO concessionnaire (c_nom, c_note, c_adresse) VALUES
  ('Auto-2000', 4, '155 avenue du général Leclerc, Orsay'),
  ('Renault-Palaiseau', 4.5, '48 rue Gambetta, Palaiseau'),
  ('Auto-moto', 3, '1 Avenue Jean Moulin, Gif-sur-Yvette');
SELECT * FROM concessionnaire

-- ON re-test
INSERT INTO concessionnaire (c_nom, c_adresse)
   VALUES ('Toyota+', '13 rue des champs, Massy'); -- OK
SELECT * FROM concessionnaire; -- Ajoute (null, 'Toyota+',
   -- null, '13 rue des champs, Massy') -- OK
INSERT INTO concessionnaire (c_nom, c_note, c_adresse)
   VALUES ('Toyota+', 5.1, '13 rue des champs, Massy'); -- KO
INSERT INTO concessionnaire (c_nom, c_note, c_adresse)
   VALUES ('Toyota+', -1, '13 rue des champs, Massy'); -- KO
INSERT INTO concessionnaire (c_nom, c_adresse)
   VALUES ('Auto-2000', '155 avenue du général Leclerc, Orsay'); -- KO
INSERT INTO concessionnaire (c_id, c_nom, c_adresse)
   VALUES (1, 'Toyota++', '54 rue des champs, Massy'); --KO
```

Question 7 :

Créez la relation `concessionnaire_modèle` qui associe les voitures disponibles chez un concessionnaire :
concessionnaire_modèle :(*c_id* : **integer**, *m_id* : **integer**, *cm_nserie* : **integer**, *cm_kilometres* : **integer**) où `c_id` et `m_id` sont des clefs étrangères sur concessionnaire et modèle. Notons aussi que le numéro de série est spécifique au modèle et que deux constructeurs peuvent avoir un numéro de série identique. Enfin, vous y insérerez les éléments suivants :

concessionnaire_modele			
c_id	m_id	nserie	kilometres
1	1	2978415	0
1	1	2777123	3000
1	3	2978415	0
3	1	2978689	1500

Solution :

```
CREATE TABLE concessionnaire_modele (
  c_id INTEGER,
  m_id INTEGER,
  cm\_nserie INTEGER,
  cm\_kilometres INTEGER,
  CONSTRAINT fk_cm_concessionnaire FOREIGN KEY(c_id)
       REFERENCES concessionnaire(c_id),
  CONSTRAINT fk_cm_modele FOREIGN KEY(m_id)
       REFERENCES modele(m_id)
);

INSERT INTO concessionnaire_modele (c_id, m_id, cm_nserie, cm\_kilometres)
  VALUES (1, 1, 2978415, 0),
    (1, 1, 2777123, 3000),
    (1, 3, 2978415, 0),
    (3, 1, 2978689, 1500);
```

Question 8 :

Proposez une clef primaire qui vous semble pertinente et ajoutez la contrainte adéquate.

Solution : Comme le numéro de série est unique au modèle seulement, alors, l'utilisation du numéro de série seul est insuffisant. On peut donc choisir comme clef primaire le couple m_id, nserie.

```
ALTER TABLE concessionnaire_modele
   ADD CONSTRAINT pk_concessionaire_modele PRIMARY KEY (m_id, nserie)
```

Question 9 :

Retrouver tous les constructeur et type de véhicules vendu par "Auto-2000" en utilisant des jointures naturelles (attention un modèle ne doit apparaitre qu'une seule fois).

Solution :

```
SELECT DISTINCT constructeur, type FROM modele M
   NATURAL JOIN concessionnaire_modele CM
   NATURAL JOIN concessionnaire C
   WHERE c_nom = 'Auto-2000';
```

Question 10 :

Afficher les concessionnaires et les véhicules des concessionnaires (nom concessionnaire, nserie, constructeur, type, kilometres), si le concessionnaire ne vend pas de véhicules neufs nous aurons une unique ligne pour le concessionnaire et les autres champs seront NULL.

Solution :

```
SELECT c_nom, cm_nserie, c_constructeur, c_type, cm_kilometres
   FROM concessionnaire
   NATURAL LEFT JOIN concessionnaire_modele
   NATURAL LEFT JOIN modele;
```

Résultat :

nom	nserie	constructeur	type	kilometres
Auto-2000	2978415	Toyota	Prius	0
Auto-2000	2777123	Toyota	Yaris	3000
Auto-2000	2978415	Toyota	Yaris	0
Renault-Palaiseau	NULL	NULL	NULL	NULL
Auto-moto	2978689	Toyota	Yaris	1500
Toyota+	NULL	NULL	NULL	NULL

Question 11 :

Afficher les concessionnaires et tout ses véhicules (nom concessionnaire, nserie, constructeur, type, kilometre) si le concessionnaire vend au moins un véhicule d'occasion (le nombre de kilomètres est supérieur à 0).

Solution :

```
SELECT nom, nserie, constructeur, type, kilometres
  FROM concessionnaire C
  NATURAL JOIN concessionnaire_modele
  NATURAL JOIN modele
  WHERE C.c_id IN (
    SELECT DISTINCT c_id FROM concessionnaire
    NATURAL JOIN concessionnaire_modele
    WHERE cm_kilometres > 0
  );
```

Question 12 :

En terme de colonnes retournées, quelle est la différence entre l'instruction **A NATURAL JOIN B** et **A JOIN B ON ...**. Explicitez cette différence avec deux requêtes et copiez le résultat dans votre fichier.

Solution : Lorsque nous utilisons la jointure naturelle, les colonnes de jointures sont fusionnées, ce qui n'est pas le cas avec l'utilisation de la jointure en précisant les colonnes de jointure. Ainsi dans le premier résultat (ointure naturelle) ci-dessous nous avons 7 colonnes alors que dans le second (jointure avec champ) nous en avons 8.

```
SELECT DISTINCT * FROM modele M NATURAL JOIN concessionnaire_modele CM ;
```

Résultat :

m_id	constructeur	type	version	c_id	nserie	kilometres
1	Toyota	Yaris	CY-2018	1	2777123	3000
1	Toyota	Yaris	CY-2018	1	2978415	0
1	Toyota	Yaris	CY-2018	3	2978689	1500
3	Toyota	Prius	CP-2012	1	2978415	0

```
SELECT DISTINCT * FROM modele M JOIN concessionnaire_modele CM
   ON M.m_id= CM.m_id;
```

Résultat :

m_id	constructeur	type	version	c_id	m_id-2	nserie	kilometres
1	Toyota	Yaris	CY-2018	1	1	2777123	3000
1	Toyota	Yaris	CY-2018	1	1	2978415	0
1	Toyota	Yaris	CY-2018	3	1	2978689	1500
3	Toyota	Prius	CP-2012	1	3	2978415	0

⚠ Enregistrez vos ou votre script en cliquant sur l'icône 💾 et téléchargez le(s) fichier(s) sur votre machine (`Tools→Storage Manager...`). Vous enverrez à l'adresse mail thomas.gerald@universite-paris-saclay.fr un rapport nommé tp1-exo1-[nom]-[prenom].pdf. Le code doit s'exécuter ! Les sorties et les essais seront expliqués

Exercice 2 : Sélection et requêtes imbriquées

Nous allons maintenant considérer la base de données "movie" dans cet exercice, accessible via le schéma movie.

Question 1 :

Que contient la relation `movies`, décrivez la relation sous la forme Table(nom-colonne type, nom-colonne type) en soulignant la clef primaire.

Solution : La requête :

```
SELECT * FROM movies;
```

permet de répondre à la question, nous pouvons donc donner le schéma suivant :
movies(*id* : **int**, *title* : **str**, *release_date* : **str**,*budget* : **bigint**, *revenue* : **bigint**, *popularity* : **float**, *runtime* : **int**, *rating* : **float**, *original_language* : **int**, *belongs_to_collection* : **int**,*overview* : **str**)

Question 2 :

Faites la même chose pour les relations `actors` et `persons`.

Solution : actors :

```sql
SELECT * FROM actors LIMIT 10;
```

actors(*id* : `int`, *person_id* : `int`, *movie_id* : `int`, *order_id* : `int`)

persons :

```sql
SELECT * FROM persons LIMIT 10;
```

actors(*id* : `int`, *name* : `str`)

Question 3 :

Combien d'enregistrements ont les relations précédentes ? Quelles requêtes avez vous utilisée ?

Solution :

- movies : 45433
- persons : 353343
- actors : 562044

```sql
SELECT COUNT(*) FROM movies;
SELECT COUNT(*) FROM persons;
SELECT COUNT(*) FROM actors;
```

Question 4 :

Décrivez la relation actors, quelle est sa différence avec la relation persons ?

Solution : La relation actors référence pour chaque film les acteurs, il peut y avoir plusieurs fois la même personnne dans cette table (un acteur à joué dans plusieurs films)

Question 5 :

Déterminez les trois films avec le plus grand budget

Solution :

title	budget
Pirates of the Caribbean : On Stranger Tides	380000000
Pirates of the Caribbean : At World's End	300000000
Avengers : Age of Ultron	280000000

```sql
SELECT title, budget FROM movies WHERE budget IS NOT NULL
   ORDER BY budget DESC LIMIT 3 ;
```

⚠ Attention si le NOT NULL n'est pas donnée alors les premier résultats sont les films avec un budget inconnu

Question 6 :

Donnez les trois films ayant eu les bénéfices les plus importants, vous retournerez la valeur du bénéfice et le titre des films.

Solution :

title	benefice
Avatar	2550965087
Star Wars : The Force Awakens	1823223624
Titanic	1645034188

```
SELECT title, (revenue-budget) as benefice FROM movies
   WHERE budget IS NOT NULL and revenue is not null
   ORDER BY (revenue-budget) DESC LIMIT 3 ;
```

Question 7 :

À l'aide de l'opération de jointure, sélectionnez le nom des acteurs ayant un rôle dans le film "Mars Attacks !", combien d'enregistrements avez vous récupérés ?

Solution :

```
SELECT movie.persons.name
FROM
  movie.persons, movie.movies, movie.actors
WHERE
  title = 'Mars Attacks!'
  AND movie.movies.id=movie.actors.movie_id
  AND movie.actors.person_id=movie.persons.id;
```

Question 8 :

À l'aide d'une requête imbriquée, sélectionnez le nom des films dans lesquels ces acteurs ont joués. Attention le nom des films ne doit apparaître qu'une seule fois dans le résultat de la requête !!!

Solution :

```
SELECT DISTINCT movie.movies.title FROM movie.movies, movie.actors WHERE
movie.actors.person_id IN (
SELECT movie.actors.person_id
FROM
  movie.movies, movie.actors
WHERE
  title = 'Mars Attacks!'
  AND movie.movies.id=movie.actors.movie_id) AND
  movie.actors.movie_id=movie.movies.id;
```

Question 9 :

Même question, mais cette fois-ci on veut retrouver les films où au moins **deux** acteurs de "Mars Attacks" ont joué.

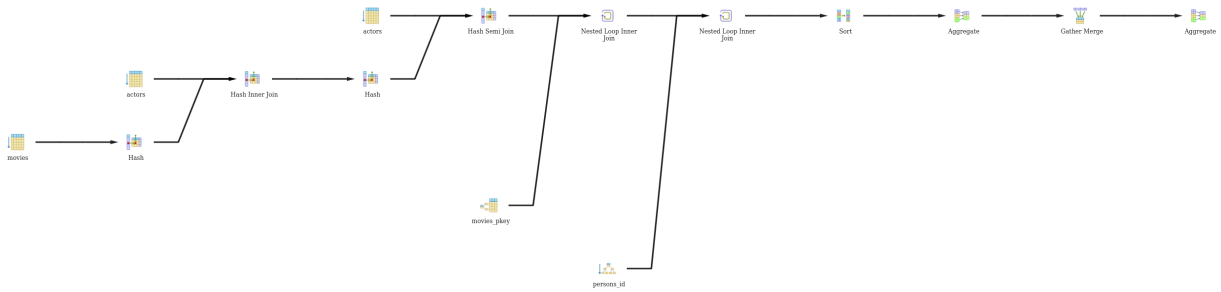
Solution :

```
SELECT movie.movies.title, STRING_AGG(movie.persons.name, ', ') FROM
movie.movies, movie.actors, movie.persons WHERE
movie.actors.person_id IN (
SELECT movie.actors.person_id
FROM
  movie.movies, movie.actors
WHERE
  title = 'Mars Attacks'
  AND movie.movies.id=movie.actors.movie_id) AND
  movie.actors.movie_id=movie.movies.id AND
  movie.actors.person_id=movie.persons.id
  GROUP BY movie.movies.title HAVING COUNT(movie.actors.id)>1;
```

Question 10 :

Pour la requête précédente, utilisez l'icône . Vous devriez voir un diagramme s'afficher, expliquez (avec vos connaissances et votre intuition) de quoi s'agit-il ? (ce diagramme sera plus clair à la fin de la deuxième partie du cours !!!)

Solution :



Il s'agit du plan d'exécution, c'est à dire le graphe des opérations exécutées pour obtenir le résultat de la requête !

⚠ Comme pour l'exercice précédant vous sauvegarderez et téléchargerez votre script SQL.

Rendu

Vous devez déposer un rapport au format pdf contenant les réponses aux deux exercices intutilé **tp1-[nom]-[prenom].pdf** sur la plateforme e-campus.