Aujourd'hui, nous allons utiliser un autre schéma, le schéma `sakila` qui contient des informations utilisées par un magasin de location de DVD. Le schéma est visible à cette adresse `http://stephane-v-boucheron.fr/WordPress3/wp-content/uploads/2016/01/sakila-schema.png`. Vous pouvez aussi inspecter les tables comme d'habitude avec

```
bd_2016=# \d sakila.film
bd_2016=# \d sakila.actor
```

1 Requêtes SQL (encore plus) avancées

1.1 Requêtes imbriquées

Les requêtes imbriquées permettent d'utiliser le resultat d'une requête dans la clause `WHERE` d'une requête. On utilisera essentiellement les deux connecteurs suivants : `IN, EXISTS`. `IN` permet de tester la présence d'une valeur dans le resultat d'une requête. `EXISTS` renvoie `True` si la requête donnée est non-vide et `False` sinon. On peut les combiner avec `NOT` pour inverser leur comportement : `NOT IN` et `NOT EXISTS`. Par exemple, pour connaître les films disponibles en allemand, on pourra écrire :

```
SELECT * FROM sakila.film
WHERE language_id IN (SELECT language_id
                      FROM sakila.language
                      WHERE name='German');
```

Pour connaître les acteurs qui ont joué dans au moins un film, on pourra écrire :

```
SELECT * FROM sakila.actor AS ac
WHERE EXISTS (SELECT *
              FROM sakila.film_actor as fa
              WHERE fa.actor_id = ac.actor_id);
```

1.2 Jointure externe

La jointure externe est une jointure un peu particulière. On a vu la semaine dernière que lorsqu'on faisait une jointure, les lignes de la table de droit étaient recollées aux lignes de la table de gauche. Si une ligne a gauche ne pouvaient pas être recollée, elle disparaissait de la jointure. La jointure extérieure permet de garder ces lignes-là malgré tout.

On utilisera `LEFT JOIN` et `RIGHT JOIN`. Par exemple, la requête suivante renvoie la liste des pays et leur langages. Les pays qui ne se trouvent pas dans la table `countrylanguage` (il y en a, l'antarctique par exemple) seront listés quand même et les informations manquantes seront remplies avec des valeurs `NULL`.

```
SELECT * FROM world.country as p
         LEFT JOIN world.countrylanguage as l
         ON p.countrycode = l.countrycode;
```

On peut utiliser cette requête pour trouver les pays qui n'ont pas de langue officielle par exemple :

```
SELECT * FROM world.country as p
         LEFT JOIN world.countrylanguage as l
         ON p.countrycode = l.countrycode AND l.isofficial
         WHERE l.countrycode IS NULL;
```

2 Exercices

2.1 Requêtes

1. Quels sont les langues qui ne sont officielles dans aucun pays ? Écrivez une version avec `NOT IN` et une autre avec `LEFT JOIN`.
2. Quels sont les films qui n'ont jamais été loués ?
3. Quels sont les acteurs qui ont joués dans toutes les catégories de film ?
4. Existe-t-il des acteurs qui ne jouent avec aucun autre acteur ?
5. Nom, prénom des clients installés dans des villes sans magasins ?
6. Lister les pays pour lesquels toutes les villes ont au moins un magasin.
7. Déterminer la liste des films disponibles dans toutes les langues.
8. Un même DVD (inventory_id) peut bien sûr être loué plusieurs fois, mais pas simultanément. Proposer une requête qui vérifie que les dates de location d'un DVD donné sont compatibles.

2.2 Fonctions SQL

Dans votre schéma personnel (qui porte le nom de votre identifiant ENT), écrire une fonction SQL `film_id_cat` qui prend en paramètre une chaîne de caractère *s* et renvoie la liste des films de catégorie *s*. On rappelle la syntaxe :

```
CREATE OR REPLACE FUNCTION entid.film_id_cat(s TEXT)
RETURNS TABLE(film_id INTEGER)
LANGUAGE 'sql' AS
$$
requete
$$
```

Utilisez votre fonction pour écrire les requêtes suivantes :

1. Quels sont les acteurs qui ont déjà joué dans un film d'horreur (catégorie 'Horror') ?
2. Quels sont les acteur qui n'ont jamais joué dans une comédie ('Comedy') ?
3. Quels sont les acteurs qui ont joué dans un film d'horreur ('Horror') et dans un film pour enfant ('Children') ?

2.3 Devoir maison "hello world"

Pour la semaine prochaine, créez dans votre schéma personnel les fonctions SQL correspondant aux questions suivantes.

1. Ecrire une fonction actor_category(nom character varying(49), prenom character varying(49))qui prend en argument le nom et le prénom d'un acteur (d'une actrice) et renvoie la liste des noms des catégories de films dans lesquels il/elle a joué.
2. Ecrire une fonction co_actors(nom character varying(49), prenom character varying(49))qui renvoie les noms et prénoms des acteurs qui jouent dans un film où apparaît un acteur ou une actrice dont le nom et le prénom sont donnés en argument.