Tutoriel - interrogation d'une base de données en SQL (PostgreSQL + `pgweb`)

Fabien Duchateau (Université Claude Bernard Lyon 1) - 2025

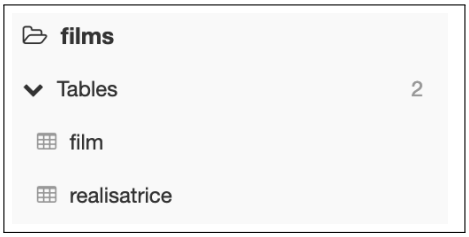
1. Pour interagir avec votre base de données PostgreSQL, nous allons **utiliser l'interface graphique `pgweb`**. Rendez-vous sur https://http://bdw.univ-lyon1.fr/ et remplissez le formulaire d'authentification pour vous connecter à votre BD.

Une fois connecté·e, vous voyez à gauche vos répertoires schémas (notamment *public*).

Host : `bd-pedago.univ-lyon1.fr`
Username : `[votre num. étu.]`
Password : sur Tomuss
(**pas** votre mot de passe UCBL !)
Database : `[votre num. étu.]`

2. Dans ce tutoriel, nous utilisons des **données sur des films et leur réalisatrice** (schéma relationnel ci-contre).
Copiez-collez le code de création des tables (voir en fin de ce document) dans l'onglet `query` et cliquer sur le bouton `Run query` pour l'exécuter : un répertoire schéma `films` devrait apparaitre.

FILM (idF, *titre*, *genre*, *annéeF*, *#idR*)
RÉALISATRICE (idR, *nom*, *annéeR*, *pays*)



3. Commençons par une première requête SQL pour récupérer **toutes les informations sur les films**. La clause `SELECT` indique les attributs attendus dans le résultat (opération de projection), et le symbole `*` signifie *tous les attributs*. La clause `FROM` indique les tables à utiliser.
Copiez-collez et exécutez la requête SQL pour visualiser le résultat.

```sql
SELECT *  -- tous les attributs
FROM film;  -- de la table film
```

idf	titre	genre	anneef	idr
1	Persepolis	animation	2007	1
2	The voices	comédie	2015	1
3	Mustang	drame	2015	2
4	Polisse	drame	2015	3
5	Gabrielle	drame	2013	4
6	Holy grail	comédie	1975	5
7	Cloud atlas	SF	2012	6
8	Koyaanisqatsi	essai	1983	NULL

4. Si nous voulons la liste des genres (de film), on précise le nom de l'attribut dans la clause `SELECT`. Pour **enlever les doublons du résultat**, c'est le mot-clé `DISTINCT`.

Testez les deux requêtes.
Notez les `--` pour ajouter un commentaire dans une requête.

```sql
SELECT genre  -- tous les genres
FROM film;  -- de la table film
```

```sql
SELECT DISTINCT(genre)  -- tous les
    genres, sans doublon
FROM film;  -- de la table film
```

genre
animation
comédie
drame
SF
essai

5. Récupérons le titre et l'identifiant de la réalisatrice pour les films sortis 2015 et après. Ici, il faut **filtrer les instances** avec la clause `WHERE` (opération de sélection). Cette clause permet d'exprimer une condition booléenne que chaque instance doit satisfaire pour être conservée dans le résultat. Essayez d'autres filtres : les réalisatrices nées avant 1960 ou celles dont le nom commence par *M* !

```sql
SELECT titre, idr  -- titre et id réalisatrice
FROM film  -- de la table film
WHERE anneef >= 2015;  -- sortis en 2015 et après
```

titre	idr
The voices	1
Mustang	2
Polisse	3

6. Dans la requête précédente, ce serait plus compréhensible si on affichait le nom de la réalisatrice (plutôt que son identifiant). Il faut mettre en correspondance 2 tables, c'est une **opération de jointure**. Elle consiste à assembler 2 instances (en une seule nouvelle instance) selon une condition.
Ici, on veut que l'identifiant réalisatrice de la table FILM corresponde à celui de la table RÉALISATRICE : c'est notre condition. Remarquez que l'on ne peut pas écrire `idr = idr` (ambiguité), donc on précise le nom de la table devant l'attribut.

```sql
SELECT titre, nom  -- titre et nom
FROM film JOIN realisatrice ON
    film.idr = realisatrice.idr  -- de la jointure entre les 2 tables
```

titre	nom
The voices	Marjane Satrapi
Mustang	Deniz Erguven
Polisse	Maiwenn

Jointure complète entre les 2 tables :

```sql
SELECT *
FROM film JOIN realisatrice ON
    film.idr = realisatrice.idr;
```

idf	titre	genre	anneef	idr	idr	nom	anneer	pays
1	Persepolis	animation	2007	1	1	Marjane Satrapi	1969	IR,FR
2	The voices	comédie	2015	1	1	Marjane Satrapi	1969	IR,FR
3	Mustang	drame	2015	2	2	Deniz Erguven	1978	TR,FR
4	Polisse	drame	2015	3	3	Maiwenn	1976	FR
5	Gabrielle	drame	2013	4	4	Louise Archambault	NULL	CA
6	Holy grail	comédie	1975	5	5	Terry Jones	1942	GB
7	Cloud atlas	SF	2012	6	6	Lana Wachowski	1965	US

7. Les **fonctions d'agrégation** produisent une seule valeur à partir des valeurs d'un attribut. Par exemple, `COUNT(idr)` va compter le nombre de réalisatrices.
La première requête retourne l'année la plus récente grâce à la fonction `MAX`. Notez le renommage de l'attribut pour l'affichage.
On peut utiliser ces fonctions d'agrégation dans des **sous-requêtes**, par exemple pour retourner les films sortis le plus récemment.

```sql
SELECT MAX(anneef) AS
    annee_plus_recente
FROM film;
```

annee_plus_recente
2015

```sql
SELECT *
FROM film
-- la sous-requête (entre parenthèses) retourne 2015
-- similaire à WHERE anneef = 2015
WHERE anneef = (SELECT MAX(anneef)
                FROM film);
```

idf	titre	genre	anneef	idr
2	The voices	comédie	2015	1
3	Mustang	drame	2015	2
4	Polisse	drame	2015	3

8. Pour certaines requêtes, il peut être nécessaire de **regrouper des instances**. Dans ce cas, la requête retourne qu'une seule instance par groupe formé.
Comptons le nombre de films pour chaque genre : avec la clause `GROUP BY`, on forme des groupes en fonction du genre, et on retourne pour chacun des groupes le genre et le nombre de films (`COUNT(*)`, renommé en *nb_films*).
Le `COUNT(*)` compte le nombre de lignes (dans le groupe), mais on peut aussi compter le nombre de valeurs d'un attribut (`COUNT(idf)`). Attention, les NULL ne sont pas pris en compte.

```sql
SELECT genre, COUNT(*) AS nb_films
    -- une instance par groupe !
FROM film
GROUP BY genre;  -- 2 instances avec le même genre sont dans le même groupe
```

genre	nb_films
animation	1
comédie	2
drame	3
SF	1
essai	1

```sql
-- même résultat qu'avec COUNT(*)
SELECT genre, COUNT(idf) AS
    nb_films
FROM film
GROUP BY genre;
```

9. On peut **filtrer des groupes** avec la clause `HAVING`. Similaire à la clause `WHERE`, elle exprime une condition booléenne qu'un groupe doit satisfaire pour être gardé.
Cherchons les réalisatrices qui ont réalisé plus d'un film. On regroupe les réalisatrices par `idR`, chaque groupe contient donc les instances (films) d'une réalisatrice. Puis on ne conserve que ceux avec plus d'un film.
Modifiez la requête pour afficher le nom des réalisatrices au lieu de leur identifiant.

```sql
SELECT idr, COUNT(*) AS nb_films
FROM film
GROUP BY idr
HAVING COUNT(*) > 1;
```

idr	nb_films
1	2

nom	nb_films
Marjane Satrapi	2

10. Enfin, la clause`ORDER BY` permet de **trier les résultats**, selon l'ordre naturel d'un attribut (e.g., alphabétique pour une chaîne). Par défaut, le tri est croissant, mais le mot-clé `DESC` spécifie un ordre décroissant. Dans cet exemple, les instances sont d'abord triées par année décroissante. En cas d'égalité sur l'année (e.g., *2015*), le tri se fait par ordre alphabétique (croissant) sur le titre.

```sql
SELECT titre, anneef
FROM film
-- tri décroissant sur le nombre de
    films, et en cas d'égalité sur
    le titre
ORDER BY anneef DESC, titre;
```

titre	anneef
Mustang	2015
Polisse	2015
The voices	2015
Gabrielle	2013
Cloud atlas	2012
Persepolis	2007
Koyaanisqatsi	1983
Holy grail	1975

Conclusion

Ces instructions SQL (pour PostgreSQL) permettent d'interroger une base de données. Ci-dessous le code de création de la base de données.
Ce tutoriel est volontairement limité pour en faciliter la prise en main. Des détails supplémentaires sont donnés dans les diapositives de cours (https://perso.liris.cnrs.fr/fabien.duchateau/) ou sur d'autres ressources comme sql.sh ou le livre ~~Not~~ Only SQL.

Code SQL de création des tables

```sql
DROP SCHEMA IF EXISTS films CASCADE;
CREATE SCHEMA films;

-- la ligne suivante permet de spécifier qu'on utilise le schéma film
-- en cas d'erreur "relation does not exist" quand vous exécutez une requête, écrivez cette ligne avant votre requête SQL
SET SEARCH_PATH TO films;

CREATE TABLE Film (
  PRIMARY KEY (idF),
  idF    INTEGER NOT NULL,
  titre  VARCHAR(200),
  genre  VARCHAR(30),
  anneeF INTEGER,
  idR    INTEGER NULL
);

CREATE TABLE Realisatrice (
  PRIMARY KEY (idR),
  idR    INTEGER NOT NULL,
  nom    VARCHAR(100),
  anneeR INTEGER,
  pays   VARCHAR(30)
);

ALTER TABLE Film ADD FOREIGN KEY (idR) REFERENCES Realisatrice (idR);

INSERT INTO Realisatrice VALUES(1, 'Marjane Satrapi', 1969, 'IR,FR');
INSERT INTO Realisatrice VALUES(2, 'Deniz Erguven', 1978, 'TR,FR');
INSERT INTO Realisatrice VALUES(3, 'Maiwenn', 1976, 'FR');
INSERT INTO Realisatrice VALUES(4, 'Louise Archambault', NULL, 'CA');
INSERT INTO Realisatrice VALUES(5, 'Terry Jones', 1942, 'GB');
INSERT INTO Realisatrice VALUES(6, 'Lana Wachowski', 1965, 'US');
INSERT INTO Realisatrice VALUES(7, 'Catherine Hardwicke', 1955, 'US');
INSERT INTO Film VALUES(1, 'Persepolis', 'animation', 2007, 1);
INSERT INTO Film VALUES(2, 'The voices', 'comédie', 2015, 1);
INSERT INTO Film VALUES(3, 'Mustang', 'drame', 2015, 2);
INSERT INTO Film VALUES(4, 'Polisse', 'drame', 2015, 3);
INSERT INTO Film VALUES(5, 'Gabrielle', 'drame', 2013, 4);
INSERT INTO Film VALUES(6, 'Holy grail', 'comédie', 1975, 5);
INSERT INTO Film VALUES(7, 'Cloud atlas', 'SF', 2012, 6);
INSERT INTO Film VALUES(8, 'Koyaanisqatsi', 'essai', 1983, NULL);
```