- **L3 MIASHS/Ingémath/METIS**
- **Université Paris Cité**
- Année 2024-2025
- Course Homepage

- Moodle

⚠ Toutes les questions portent sur le schéma `pagila` rappelé ci-dessous.
Pour chaque question, proposer une requête écrite en algèbre relationnelle *OU* en `SQL`.

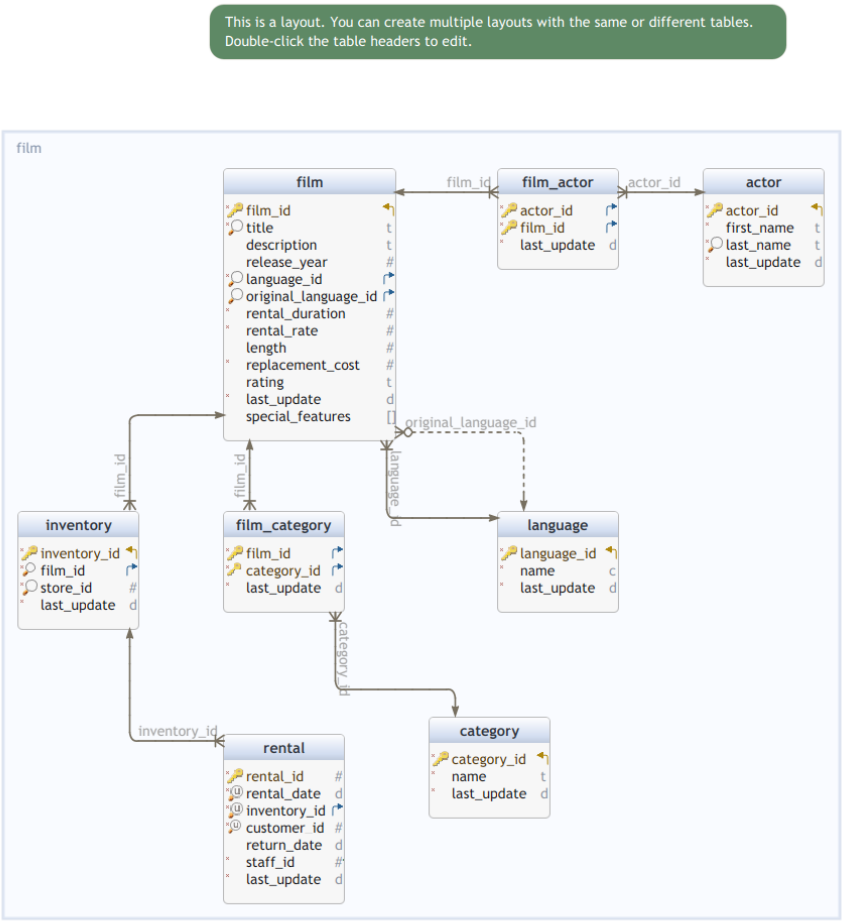

FIG. 1 : Schéma `pagila`, films

1. Donner pour chaque film au catalogue, le titre et les prénoms des acteurs qui jouent dans ce film.

Solution

```
SELECT fi.title, string_agg(DISTINCT a.first_name, ', ') AS casting
FROM
   pagila.film fi
JOIN
   pagila.film_actor fa ON (fi.film_id=fa.film_id)
JOIN
   pagila.actor a ON (fa.actor_id=a.actor_id)
GROUP BY fi.film_id;
```

👉 Il n'est pas nécessaire d'effectuer un groupement et une aggrégation, mais c'est une occasion d'illustrer cette possibilité. `string_agg()` est une fonction proposée par `PostgreSQL`, analogue à `paste()` ou `stringr::str_c()` en `R`, ou à une invocation de `functools.reduce()` en `Python` :

```
>>> foo = lambda x,y : x + ', ' + y if x else y
>>> reduce(foo, ['Brad', 'Angelina', 'Romy', 'Ava'], '')
'Brad, Angelina, Romy, Ava'
```

Des fonctions d'aggrégations comme `SUM(), MAX(), COUNT()` sont de cette forme. Des fonctions d'aggrégations comme `AVG()`, `VAR()`, `COUNT(DISTINCT ...)` sont un peu plus compliquées à réaliser mais relève du même genre de calcul.

2. Lister les titres de films qui ne sont disponibles que dans une seule langue

Solution

```
SELECT
  fi.title                                                    ③
FROM
  pagila.film fi
GROUP BY fi.title                                             ①
HAVING COUNT (DISTINCT fi.language_id)=1 ;                    ②
```

① Partitionnement sur `title`
② Filtration des classes de la partition à partir d'une condition sur l'aggrégat
③ Projection sur l'attribut de partitionnement (pas d'aggrégat utile ici)
Il n'est pas indispensable de passer par un groupement/aggrégation. On peut obtenir le résultat en utilisant simplement sélection, projection, jointure, opérations ensemblistes $(\cap, \cup, \setminus)$.

```
(
 SELECT
    DISTINCT title
 FROM
    pagila.film
)
EXCEPT
(
 SELECT
     DISTINCT f1.title
 FROM
     pagila.film f1
 JOIN
     pagila.film f2
 ON (f1.title = f2.title AND f1.language_id < f2.language_id)
)
```

👉 on est parfois tenté de choisir comme condition de jointure dans la seconde sous-requête `f1.film_id = f2.film_id AND f1.language_id < f2.language_id`. Cela conduira a un résultat sans intérêt. L'attribut `film_id` est clé primaire (`primary key`) de `film` (cela fait partie du schéma, cette contrainte s'impose à toutes les instances du schéma). Quelle que soit l'instance, on ne pourra pas trouver dans la table `film`, deux tuples qui coïncident sur `film_id` et diffèrent sur `language_id`.

3. Pour chaque langue originale (`original_language_id`), chaque langue de diffusion (`language_id`), compter le nombre de titres tournés dans la langue originale et traduits dans la langue de diffusion (sans doublons).

Solution

```
SELECT
  original_language_id, language_id,                    ②
  count(title)                                          ③
FROM
    pagila.film
GROUP BY
    original_language_id, language_id                   ①
;
```

① Partitionnement sur le couple d'attributs `original_language_id, language_id`
② Projection sur les attributs de partitionnement et
③ sur la/les aggrégations

1. Donner pour chaque catégorie (de film), les noms des acteurs qui apparaissent dans le nombre maximal de films de la catégorie.

Solution

```sql
WITH ActorFilmCount AS (
    SELECT
        c.name AS category_name,
        a.actor_id,
        a.first_name || ' ' || a.last_name AS actor_name,
        COUNT(fa.film_id) AS film_count
    FROM
        actor a
    JOIN
        film_actor fa ON a.actor_id = fa.actor_id
    JOIN
        film f ON fa.film_id = f.film_id
    JOIN
        film_category fc ON f.film_id = fc.film_id
    JOIN
        category c ON fc.category_id = c.category_id
    GROUP BY
        c.name, a.actor_id, a.first_name, a.last_name
),

RankedActors AS (
    SELECT
        category_name,
        actor_name,
        film_count,
        DENSE_RANK() OVER (PARTITION BY category_name ORDER BY film_count DESC) AS rank
    FROM
        ActorFilmCount
)
SELECT
    category_name,
    actor_name,
    film_count
FROM
    RankedActors
WHERE
    rank = 1
ORDER BY
    category_name;
```

On peut se passer des fonctions fenêtres.

Solution

```sql
WITH ActorFilmCount AS (
    SELECT
        c.name AS category_name,
        a.actor_id,
        a.first_name || ' ' || a.last_name AS actor_name,
        COUNT(fa.film_id) AS film_count
    FROM
        actor a
    JOIN
        film_actor fa ON a.actor_id = fa.actor_id
    JOIN
        film f ON fa.film_id = f.film_id
    JOIN
        film_category fc ON f.film_id = fc.film_id
    JOIN
        category c ON fc.category_id = c.category_id
    GROUP BY
        c.name, a.actor_id, a.first_name, a.last_name
),


MaxPerCat AS (
    SELECT
        category_name,
        MAX(film_count) as m_film_count
    FROM
        ActorFilmCount
    GROUP BY
        category_name
),


SELECT
    ac.category_name,
    ac.actor_id,
    ac.actor_name,
    ac.film_count
FROM
    ActorFilmCount  ac
JOIN
    MaxPerCat  mpc
USING(category_name)
WHERE
    ac.film_count=mpc.m_film_count
ORDER BY
    ac.category_name ;
```

ou encore

Solution

```sql
WITH ActorFilmCount AS (
    SELECT
        c.name AS category_name,
        a.actor_id,
        a.first_name || ' ' || a.last_name AS actor_name,
        COUNT(fa.film_id) AS film_count
    FROM
        actor a
    JOIN
        film_actor fa ON a.actor_id = fa.actor_id
    JOIN
        film f ON fa.film_id = f.film_id
    JOIN
        film_category fc ON f.film_id = fc.film_id
    JOIN
        category c ON fc.category_id = c.category_id
    GROUP BY
        c.name, a.actor_id, a.first_name, a.last_name
)

SELECT
    afc.*
FROM
    ActorFilmCount afc
WHERE
    afc.film_count >= ALL (
        SELECT
            a.film_count
        FROM
            ActorFilmCount a
        WHERE
            a.category_name = afc.category_name
    ) ;
```

1. Lister sans doublons les paires d'acteurs qui n'ont jamais joué dans un même film.

Solution

```
WITH ActorPairs AS (
    -- Generate all possible actor pairs (excluding pairing an actor with themselves)
    SELECT
        a1.actor_id AS actor1_id,
        a2.actor_id AS actor2_id,
        a1.first_name || ' ' || a1.last_name AS actor1_name,
        a2.first_name || ' ' || a2.last_name AS actor2_name
    FROM
        actor a1
    JOIN
        actor a2 ON a1.actor_id < a2.actor_id  -- Ensure actor1_id < actor2_id to avoid duplic
),
ActorsTogether AS (
    -- Find all actor pairs who have acted together in at least one film
    SELECT
        fa1.actor_id AS actor1_id,
        fa2.actor_id AS actor2_id
    FROM
        film_actor fa1
    JOIN
        film_actor fa2 ON fa1.film_id = fa2.film_id
    WHERE
        fa1.actor_id < fa2.actor_id  -- Same condition to avoid duplicates
),
ActorsNotTogether AS (
    -- Left JOIN all possible pairs with the pairs that acted together
    SELECT
        ap.actor1_name,
        ap.actor2_name
    FROM
        ActorPairs ap
    LEFT JOIN
        ActorsTogether at ON ap.actor1_id = at.actor1_id AND ap.actor2_id = at.actor2_id
    WHERE
        at.actor1_id IS NULL  -- Only return pairs that do not exist in the ActorsTogether res
)
-- Final result
SELECT
    actor1_name,
    actor2_name
FROM
    ActorsNotTogether
ORDER BY
    actor1_name, actor2_name;
```

1. Lister les acteurs (prénom, nom) qui ont joué dans des films tournés dans au moins deux langues différentes (langue de tournage : `original_language_id`)

Solution

```sql
SELECT
    a.actor_id,
    a.first_name || ' ' || a.last_name AS actor_name,
    STRING_AGG(DISTINCT l.name, ', ') AS languages
FROM
    actor a
JOIN
    film_actor fa ON a.actor_id = fa.actor_id
JOIN
    film f ON fa.film_id = f.film_id
JOIN
    language l ON f.original_language_id = l.language_id
GROUP BY
    a.actor_id, a.first_name, a.last_name
HAVING
    COUNT(DISTINCT f.original_language_id) > 1
ORDER BY
    actor_name;
```

7. Pour chaque magasin (désigné par `store_id`), chaque langue, donnez le nombre de DVDs (physiques) disponibles dans cette langue, dans ce magasin.

Solution

```sql
SELECT
    s.store_id,
    l.name AS language_name,
    COUNT(i.inventory_id) AS inventory_count
FROM
    store s
JOIN
    inventory i ON s.store_id = i.store_id
JOIN
    film f ON i.film_id = f.film_id
JOIN
    language l ON f.language_id = l.language_id
WHERE
    f.language_id IS NOT NULL
GROUP BY
    s.store_id, l.name
ORDER BY
    s.store_id, l.name;
```