ANY, ALL, Vues

1 ANY et ALL

`ANY` et `ALL` sont deux opérateurs qu'on utilise avec des requêtes imbriquées dans la clause `WHERE`. Par exemple, `SELECT * FROM table WHERE col < ALL(requete)` sélectionnera les lignes de `table` telles que la valeur de `col` est plus petite que toutes les valeurs retournées par la requête `requete`. Par exemple,

```
SELECT * FROM world.country
WHERE population_country >= ALL(SELECT population_country FROM world.country)
```

retournera la liste des pays les plus peuplés. `SELECT * FROM table WHERE col < ANY(requete)` sélectionnera les lignes de `table` telles que la valeur de `col` est plus petite qu'au moins une des valeurs retournées par la requête `requete`.

1. Comment simuler `IN` avec un `ANY` ?

   Solution: Réponse : `colonne=ANY(requete)`

2. Comment simuler `colonne>=ALL(requete)` ou `colonne>=ANY(requete)` avec les opérateurs d'agrégations `MIN`, `MAX` ?

   Solution: Si la colonne `col` ne peut pas prendre la valeur `NULL`, `colonne>=ALL(SELECT col FROM t)` est équivalent à `colonne>=(SELECT MAX(col) FROM t`.
   Si la colonne `col` peut prendre la valeur `NULL`, et prend effectivement la valeur `NULL`, la condition `colonne>=ALL(SELECT col FROM t)` ne sera pas réalisée et le resultat sera une table vide.
   En revanche, `colonne>=ANY(SELECT col FROM t)` est équivalent à `colonne>=(SELECT MIN(col) FROM t`.

2 Vues

Les vues permettent de donner un nom à une requête afin de pouvoir l'appeler plus tard sans la réécrire à chaque fois. Une vue s'enregistre dans un schéma. Par exemple, dans la table `Sakila`, on pourrait créer une vue `FilmComedy` qui contient toutes les comédies de la table. On crée une vue avec `CREATE VIEW nom AS requete`. Étant donné que vous ne pouvez écrire que dans votre schéma personnel, il faudra nommer vos vues `entid.nom` où `entid` est votre identifiant ENT. Ainsi

```
CREATE VIEW entid.FilmComedy AS
WITH Comedy AS (SELECT film_id FROM film_category NATURAL JOIN category
                WHERE name='Comedy')
SELECT * FROM sakila.film
WHERE film_id IN (SELECT * FROM Comedy);
```

crée une vue dans votre schéma personnel qui renvoie les comédies de la table `sakila.film`. Désormais, si on veut sélectionner les comédies qui durent plus de deux heures, on pourra simplement écrire : SELECT * FROM entid.FilmComedy WHERE duration>=120. Remarquez la différence entre `WITH` et une vue. `WITH` nomme une requête temporairement, seulement à l'échelle de la requête courante tandis qu'une vue est enregistrée de façon permanente. Cependant, chaque fois que vous appelez votre vue, elle est réévaluée par le système de base de données.

1. Proposer une vue `NeverRented` qui liste les films n'ayant jamais été loués.

Solution:

```
CREATE VIEW entid.NeverRented AS (
WITH DejaLoue AS
(SELECT film_id FROM sakila.inventory NATURAL JOIN sakila.rental)
SELECT * FROM sakila.film WHERE film_id NOT IN (SELECT * FROM
  DejaLoue)
);
```

2. Proposer une vue `StatRented` qui liste les identifiants des films et le nombre de fois où ils ont été loués.

Solution:

```
CREATE VIEW entid.StatRented AS (
SELECT film_id, COUNT(rental_id)
FROM sakila.inventory NATURAL JOIN sakila.rental
GROUP BY film_id);
```

3 Manipuler les dates et les intervalles

Deux types de données courants en SQL sont les types `date` et `interval`. Plusieurs fonctions permettent de les manipuler :

- `EXTRACT(champ FROM date)` extrait une information de la date ou de l'intervalle indiqué. Par exemple, pour extraire l'année de `rental_date`, on écrira `EXTRACT(year FROM rental_date`.
- `NOW()` renvoie la date courante.
- On peut les comparer avec les opérateurs `=, <, >, max, min` etc.
- On peut créer une date avec la fonction `make_date(year, month, day)`. On peut aussi utiliser `CAST('yyyy-mm-dd' AS DATE)` pour transformer une chaîne de caractère en date.
- On peut additionner/soustraire les dates (cela donne des intervalles) ou des intervalles.

1. Combien de DVD ont-il été loués en Juin 2005 ?

Solution:

```
SELECT COUNT(rental_id)  FROM sakila.rental
WHERE EXTRACT(year from rental_date)=2005 AND
      EXTRACT(month from rental_date)=6;
```

2. Combien de jours se sont-ils écoulés depuis la dernière location enregistrée dans `sakila` ?

Solution:

```
SELECT EXTRACT(day FROM NOW()-MAX(rental_date)) FROM sakila.rental;
```

4 Proposer des requêtes pour répondre aux questions suivantes

1. Existe-t-il deux acteurs qui ont exactement le même prénom ? Indiquez prénom par prénom le nombre d'acteurs qui le portent, trié par fréquence décroissante.

Solution:

```
SELECT first_name, COUNT(actor_id) as n FROM sakila.actor
GROUP BY first_name
ORDER BY n DESC;
```

2. Déterminer les noms des acteurs qui jouent dans au moins trente films.

Solution:

```
WITH ActID AS (SELECT actor_id FROM sakila.film_actor
GROUP BY sakila.actor_id
HAVING COUNT(film_id) >= 30)
SELECT last_name FROM sakila.actor NATURAL JOIN  ActID;
```

3. Afficher le nom et le prénom des acteurs dont le nom commence par une lettre supérieure ou égale à `N`. Trier le résultat par nom.

Solution:

```
SELECT last_name,first_name FROM sakila.actor WHERE last_name>='N'
ORDER BY last_name;
```

4. Existe-t-il des acteurs qui jouent dans les mêmes films exactement ?

Solution:

```
WITH FilmDiff AS (SELECT a.actor_id, b.actor_id
FROM sakila.film_actor as a JOIN sakila.film_actor as b
ON (a.actor_id < b.actor_id and a.film_id != b.film_id))

(SELECT a.actor_id,b.actor_id FROM sakila.actor as a, sakila.actor as
    b WHERE a.actor_id < b.actor_id)
EXCEPT SELECT * FROM FilmDiff;
```

5. Existe-t-il des films sans acteurs ?

Solution:

```
SELECT film_id, title FROM sakila.film
WHERE film_id NOT IN (SELECT film_id FROM sakila.film_actor);
```

6. Calculer le nombre de DVD loués par mois.

Solution:

```
SELECT EXTRACT(year FROM rental_date) as year,
       EXTRACT(month FROM rental_date) as month,
       COUNT(rental_id)
FROM rental GROUP BY year,month;
```

7. Un client n'est autorisé à détenir que 3 DVDs simultanément. Vérifier qu'aucun client ne dépassait cette limite le 15 Juin 2005.

Solution:

```
WITH IsRenting AS (
SELECT customer_id,rental_id FROM sakila.rental
WHERE rental_date <= make_date(2005,06,15) and
      (return_date IS NULL or return_date >= make_date(2005,06,15)))
SELECT customer_id FROM IsRenting GROUP BY customer_id HAVING COUNT(
    rental_id)>3;
```

8. Quel film a-t-il été loué le plus longtemps en tout (sans compter les DVD non-rendus) ?

Solution:

```
WITH TimeRented AS ( SELECT film_id,SUM(return_date-rental_date) as
s FROM rental NATURAL JOIN inventory GROUP BY film_id)

SELECT film_id FROM TimeRented WHERE s = (SELECT max(s) FROM
TimeRented);
```

9. Combien y a-t-il de clients à Moscou ?

Solution:

```
SELECT COUNT(customer_id)
FROM sakila.customer NATURAL JOIN sakila.address NATURAL JOIN
      sakila.city
WHERE city='Moscow';
```

10. Pour chaque pays indiquez le nombre de clients.

Solution:

```
SELECT country,COUNT(customer_id)
FROM sakila.customer NATURAL JOIN sakila.address NATURAL JOIN
      sakila.city NATURAL JOIN sakila.country
GROUP BY country_id;
```

11. Pour chaque `customer_id`, quelle est la catégorie de film qu'il a le plus loués ?

Solution:

```
WITH NbRented AS (SELECT customer_id, category_id, COUNT(film_id) as
N FROM rental NATURAL JOIN inventory NATURAL JOIN film_category
GROUP BY customer_id, category_id)

SELECT customer_id, category_id FROM NbRented as R WHERE N = (SELECT
MAX(N) FROM NbRented WHERE customer_id = R.customer_id);
```

12. Quel est le nombre moyen de clients par pays ?

Solution:

```
WITH CustomerPerCountry AS
(SELECT country,COUNT(customer_id) AS N
FROM sakila.customer NATURAL JOIN sakila.address NATURAL JOIN
```

```
        sakila.city NATURAL JOIN sakila.country
    GROUP BY country_id)

    SELECT AVG(N) FROM CustomerPerCountry;
```

13. Lister les pays pour lesquels toutes les villes ont au moins un client.

Solution:

```
WITH CityWithNoCustomer AS
(SELECT city_id,country_id FROM sakila.city as V
WHERE NOT EXISTS
(SELECT * FROM customer NATURAL JOIN address WHERE city_id=V.city_id
  ))

SELECT country_id FROM sakila.country
WHERE country_id NOT IN (SELECT country_id FROM CityWithNoCustomer);
```

14. Déterminer la liste des films disponibles dans toutes les pays.

Solution:

L'agrégation est pratique dans ce cas. On compte le nombre de pays où le film est disponible et on vérifie que c'est le même nombre que le nombre de pays.

```
SELECT film_id FROM inventory NATURAL JOIN store
                NATURAL JOIN address
                NATURAL JOIN city
GROUP BY film_id
HAVING  COUNT(DISTINCT country_id) =
(SELECT COUNT(country_id) FROM sakila.country);
```

15. Créer un rapport qui montre le titre, la note, la catégorie, le prix, la longueur des films qui durent plus de trois heures triés par taux de location décroissants.

Devoir maison

Écrire dans votre schéma des fonctions pour répondre aux questions suivantes sur le schéma `sakila`.

1. Écrire une fonction qui prend en argument une chaine de caractères `jour` (date au format 'yyyy-mm-dd' et un entier `n` et renvoie la liste des DVD loués depuis plus de `n` jours à la date spécifiée par `jour`.

```
CREATE OR REPLACE FUNCTION entid.en_retard(jour CHARACTER(10), n INTEGER)
RETURNS TABLE(inventory_id integer)
LANGUAGE sql AS $$
-- TODO
$$ ;
```

2. Écrire une fonction qui renvoie le nombre de DVDs (instances d'`inventory`) déjà rendu par le client dont le nom et le prénom sont donnés en argument à la date définie par `jour`, date au format 'yyyy-mm-dd' (tant qu'un DVD n'est pas rendu, le champ return_date est NULL).

```
CREATE OR REPLACE FUNCTION entid.a_deja_rendu(prenom character varying(45),
                    nom character varying(45),
                    jour character(10))
RETURNS BIGINT LANGUAGE sql AS $$
```

```
-- TODO
$$ ;
```