Maintenir une base de donnée - 2

Les triggers et les tâches planifiées

Sommaire

- Les triggers
  - Définition
  - Exemple
  - Mise en œuvres
- Tâches planifié sur MySQL
  - Exemple : purger régulièrement une table
  - Exemple : création régulière de statistiques

Les triggers

Dans toutes base de données il est possible de déclencher des traitements sur des évènements liées à la manipulation des données : ajout, modification, et suppression.

Définition :

Les **TRIGGERS** sont des traitements qui peuvent être déclenché avant ou après chaque requêtes **INSERT**, **UPDATE** ou **DELETE**.

Les triggers

Exemple :

Pour accélérer le temps d'exécution des requêtes permettant de récupérer le nombre de production par pays, on décide de stocker ces donnée de manière permanente dans la base de données.

```
CREATE TABLE country(
country_id int PRIMARY KEY NOT NULL,
country varchar(50),
quant int);
```

```
UPDATE country
SET quant = (select count(title) from shows
    right join show_country
        on shows.show_id=show_country.show_id
    where country_id = 1
    group by country_id)
where country_id = 1;
```

Les triggers

```
DELIMITER $$
CREATE TRIGGER count_pays AFTER INSERT ON show_country
FOR EACH ROW

BEGIN
UPDATE country
SET quant = (select count(title) from shows
    right join show_country
    on shows.show_id=show_country.show_id
    where country_id = NEW.country_id)
where country_id = NEW.country_id;
END $$
DELIMITER ;
```

- On à la possibilité de déclencher le traitement avant ou après l'**INSERT** avec les paramètres **AFTER** ou **BEFORE**.
- On peut déclencher l'évènement l'or d'un **INSERT**, un **DELETE** ou un **UPDATE**.
- La commande **ON EACH ROW** indique le traitement est exécuté pour chaque enregistrement.
- Le mot clé **NEW/OLD** permet d'accéder au données en cour de traitement avant/après l'exécution de la requête.

Les triggers

- J’alimente m’a base de données avec mon code python après la mise en place de mon trigger.

```
select * from country;
```

	country_id	country	quant
▶	0	United States	2609
	1	India	838
	2	South Korea	162
	3	China	120
	4	United Kingdom	601

```
CALL nouveau_film(999997, 'Movie', 'Simplon dark knight',
    2019, 'NR', 180, 'Deuxième volet de la franchise Simplon, Un chevalier noir fait irruption...');
INSERT INTO show_country VALUES (0, 999997);
```

```
select * from country;
```

	country_id	country	quant
▶	0	United States	2610
	1	India	838
	2	South Korea	162
	3	China	120
	4	United Kingdom	601

Tâches planifiées

Il est possible sur MySQL de créer une tâche planifiée afin d'exécuter régulièrement une requête.

Dans un premier temps il faut vérifier si le gestionnaire d'événement de MySQL est désactivé. Si c’est le cas activé le.

```
SHOW VARIABLES WHERE VARIABLE_NAME = 'event_scheduler';
SET GLOBAL event_scheduler = 0;
```

Tâches planifiées

Exemple : purger régulièrement une table

Tout les vieux film de Netflix sont conservés dans une table historique mais ne doivent plus apparaitre dans le catalogue.

La requête suivante est une solution simple à mettre en place :

```
CREATE EVENT purge_vieux_films
ON SCHEDULE EVERY 1 YEAR
STARTS '2020-01-01 09:50:00'
DO
 DELETE LOW_PRIORITY FROM shows WHERE
 release_year < (YEAR(NOW()) - 5);
```

Tâches planifiées

Exemple : création régulière de statistiques

Pour accélérer l'affichage de statistique sur les acteurs, on peut les stockée en mémoire plutôt que d'avoir à les recalculer à chaque fois.

```
DELIMITER $$
CREATE EVENT even_stat_journaliere
ON SCHEDULE EVERY 1 DAY
STARTS '2020-01-01 09:50:00'
DO
BEGIN
    DROP TABLE IF EXISTS nb_film_par_acteurs;
    CREATE TABLE nb_film_par_acteurs as
        select cast, count(*) as nb from show_cast
            right join casting
                on show_cast.casting_id = casting.casting_id
            group by cast;
END $$
```