Maintenance de la base de données

Cohérence et déclanchement des traitement de données



Sommaire

- Respecter les contraintes d'intégrités fonctionnelles
 - Mise en places d'évènements l'ors de la création d'une table
 - Définition
 - Exemple
- Transaction
 - Définition
 - Syntaxe
 - Exemple
- Application : maintenir sa base de données

Respecter les contraintes d'intégrités fonctionnelles

Une mise à jour ou une suppression d'enregistrements peut parfois échouer si les contraintes posées sur la table ne sont plus respectées.

Solution:

On peut définir des évènements natif au SGBD, lors de la création ou de la modification de tables, qui vont déclencher des traitements afin de garantir les contraintes d'intégrité existantes.

Ces évènement sont :

- ON UPDATE
- ON DELETE

Ils sont précisés au niveau d'une clé étrangère afin de déterminer l'action qui doit être faite l'or de la suppression de la clé primaire parent.

Ils sont souvent apparenté à des TRIGGERS.

Respecter les contraintes d'intégrités fonctionnelles

Les actions possibles sont :

- CASCADE : si une clés primaire parent est modifiée ou supprimée, il en est de même pour les clés étrangère reliées.
- SET NULL/ST DEFAULT : Si une clé primaire parent est modifiée ou supprimée, les clés étrangère prennent une valeur NULL.
- **RESTRICT** : Si une clés primaire parent est modifiée ou supprimée un message d'erreur apparait.

Mise en places d'évènements l'ors de la création d'une table

```
CREATE TABLE shows(
show_id int PRIMARY KEY NOT NULL,
type varchar(10),
title varchar(255),
date_added date,
release_year int,
rating varchar(10),
duration int,
description text);
```

```
CREATE TABLE director(
director_id int PRIMARY KEY NOT NULL,
director varchar(50));

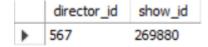
CREATE TABLE show_director(
director_id int,
show_id int,
FOREIGN KEY (director_id)
    REFERENCES director(director_id) ON DELETE CASCADE,
FOREIGN KEY (show_id)
    REFERENCES shows(show_id) ON DELETE CASCADE);
```

Mise en places d'évènements l'ors de la création d'une table

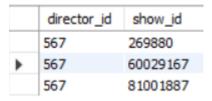


	show_id	type	title	date_added	release_year	rating	duration	description
•	269880	Movie	Bad Boys	2019-10-01	1995	R	119	In this fast-paced actioner, two Miami narcotics \dots

select * from show_director where show_id = 269880;

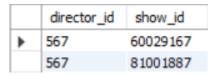


select * from show_director where director_id = 567;



Mise en places d'évènements l'ors de la création d'une table

```
SET SQL_SAFE_UPDATES = 0;
delete from shows where title like 'Bad Boys';
select * from show_director where director_id = 567;
```





D'un point de vues métier il est rare que la demande d'un utilisateur puisse être traité en une seul requête SQL.

Exemple : Problème du site voyage

Un utilisateur souhaite réserver un billet d'avion et une chambre d'hôtel à la fois, mais pas l'un sans l'autre.

Et c'est là que les **transactions** interviennent pour gérer ce type de problème.

Définition :

Une transaction est une suite d'opération qui font passer une base de données d'un état A à un état B.

Une transaction à plusieurs propriétés qui assurent la cohérence des données l'or de ce changement d'état :

L'atomicité

Cette propriétés assure que toute les modifications d'une transaction doivent être soit validées soit annulées (complète ou pas du tout).

La cohérence

L'or de l'état final de la transaction toute incohérence (contraintes d'intégrité, ROLLBACK, ...) entre les données doit entrainer une annulation de la transaction pour revenir à un état cohérent.

Isolée

L'état de la base de données au cour d'une transaction ne doit être visible que par cette transaction. Les modifications de deux transaction simultané ne doivent pas intervenir l'une sur l'autre.

• Durable

Les modification d'une transaction doivent être conservé même en cas de panne.

Moyen mnémotechnique : ACID

- Une transaction démarre par **START TRANSACTION**.
- Deux commandes peuvent modifier la fin de la transaction avec de comportement différent :
 - **COMMIT** : Les insertions, modifications, suppressions de données sont validées.
 - **ROLLBACK** : Les insertions, modifications, suppressions de données sont annulées.
- La propriété AUTOCOMMIT permet de gérer un COMMIT automatique ou manuel.

```
SET autocommit = 0;
SET autocommit = 1;
```

Exemple d'usage de transaction :

L'entreprise Netflix souhaite mettre en place une politique de qualité et ne faire apparaitre dans son catalogue de film en ligne seulement des film dont la date de production n'est pas plus veille que trois ans.

La mise en place d'une **TRANSACTION** permet de bloquer l'entrer d'un nouveau film si la condition sur l'année de production n'est pas respecté.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS nouveau film $$
CREATE PROCEDURE nouveau film (
   IN show id INT,
   IN type VARCHAR(10),
   IN title VARCHAR(255),
   IN release year INT,
   IN rating VARCHAR(10),
   IN duration INT,
   IN description TEXT)
/* Label de sortie */
sortie :
BEGIN
DECLARE temps INT;
/* Debut e la transaction */
START TRANSACTION;
/* Requête d'insertion dans la table INSCRIPTION */
INSERT INTO shows (show_id, type, title, date_added,
                    release year, rating, duration, description)
   VALUES (show_id, type, title, NOW(), release_year,
            rating, duration, description);
```

```
/* rècupère l'âge du coureur le jour de la course */
SET temps = year(NOW()) - release year;
/* Dans le cas ou la l'ancièneté du film est supèrieur à 3 ans */
IF (temps > 3) THEN
    /* Annulation de la transaction */
    ROLLBACK;
    /* Sortie de la procédure */
    LEAVE sortie;
END IF;
/* Validation e la transaction */
COMMIT;
END $$
DELIMITER ;
```

	show_id	type	title	date_added	release_year	rating	duration	description
•	999999	Movie	Simplon for ever	2020-12-13	2019	NR	180	La vie movementé de simplonien durant leur for

Application : maintenir sa base de données

Problème:

Si tout les films d'un directeur sont supprimés de la base, alors le nom du directeur en question subsiste dans la table directeur mais sans être affilié à aucun films.

La base de données doit être maintenue via une tâche automatisé pour éviter l'accumulation de données inutiles.

<u>Solution:</u>

- Créer une procédure SQL qui aura pour argument d'entré le nom d'un directeur et qui le supprimera de la table directeur si celui-ci n'a aucun film enregistré dans la table show.
- Créer un programme python qui, en utilisant la procédure, supprimera tout les noms de directeur n'ayant pas film.