Triggers: Déclencheurs MySQL

Nous utiliserons l'ensemble de données de la librairie(**bookstore**), à collecter auprès du formateur.

Créer un Trigger

```
CREATE TRIGGER [IF NOT EXISTS] trigger_name
    { BEFORE | AFTER } { INSERT | UPDATE | DELETE }
    ON tbl_name FOR EACH ROW
    trigger_body
```

> Code ci-haut est une version simplifiée de la syntaxe. Version complète ici

Obtenir tous les déclencheurs existants

```
SHOW TRIGGERS
```

Supprimer un Trigger

```
DROP TRIGGER [IF EXISTS] trigger_name
```

> Code ci-haut est une version simplifiée de la syntaxe. Version complète ici

Example 1

Un déclencheur pour réduire automatiquement la quantité de livres en stock à chaque nouvel achat

Créeation

```
DELIMITER //
CREATE TRIGGER trg_purchases_update_qty_on_purchase
AFTER INSERT ON purchases
FOR EACH ROW
BEGIN
    UPDATE books SET quantity = quantity - NEW.quantity WHERE id = NEW.book_id;
END //
DELIMITER ;
```

déclencheur

```
INSERT INTO purchases (book_id, quantity) VALUES(1, 10);
```

Example 2

```
CREATE TRIGGER trg_purchase_insert_fail_qty_gt_stock
BEFORE INSERT ON purchases FOR EACH ROW
BEGIN
        IF NEW.quantity > (SELECT quantity FROM books WHERE id = NEW.book_id)
    THEN
                 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Quantity in purchase greater than
    END IF;
END
```

In the example above, we use an *IF Statement* to verify that the purchase quantity does not exceeds the availbale quantity in stock. We then throw an exception using the SQLSTATE condition with a generic value for unhandled exception. The syntax for an IF statement is as follow:

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF
```

Syntax for an throwing an exception:

```
SIGNAL condition_value [SET signal_information_item];
```

> Code ci-haut est une version simplifiée de la syntaxe. Version complète ici