Rapport : les curseurs en MySQL

Module : Base de Données MySQL

Nom: Aya ouazzani akkar hammoudi

SOMMAIRE :

1) Le contexte:

Cet atelier s'inscrit dans le cadre de la gestion budgétaire pour des projets. L'objectif est de pratiquer l'utilisation des curseurs en MySQL afin de parcourir ligne par ligne les données d'une table, effectuer des calculs, mettre à jour les informations et générer des alertes.

les objectifs pédagogiques:

Créer et exécuter une procédure stockée utilisant un **curseur**.

Calculer automatiquement le **taux d'utilisation du budget** pour chaque projet.

Mettre à jour le **statut** des projets selon leur consommation.

Générer des **alertes dynamiques** en cas de dépassement.

Tracer toutes les opérations dans un **historique** pour consultation future.

Créer une **fonction personnalisée réutilisable** pour calculer le taux d'un projet spécifique.

2) Le code SQL :

Étape 1 – Création de la base et des tables

-- Création de la base

```
CREATE DATABASE IF NOT EXISTS gestion_projets;
USE gestion_projets;
```

-- Table principale

```sql
CREATE TABLE projets (
    id_projet INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100),
    budget DECIMAL(10,2),
    depenses DECIMAL(10,2),
    taux_utilisation DECIMAL(5,2),
    statut VARCHAR(20)
);

-- Table secondaire : alertes
CREATE TABLE alertes (
    id_alerte INT AUTO_INCREMENT PRIMARY KEY,
    id_projet INT,
    message VARCHAR(255),
    date_alerte DATETIME
);
-- Table historique : pour tracer les calculs
CREATE TABLE historique (
    id INT AUTO_INCREMENT PRIMARY KEY,
    id_projet INT,
    taux DECIMAL(5,2),
    date_calcul DATETIME
);
```

Étape 2 – Insertion de données de test :

```sql
INSERT INTO projets (nom, budget, depenses) VALUES
('Projet Alpha', 10000, 8500),
('Projet Beta', 5000, 6000),
('Projet Gamma', 8000, 2000);
SELECT * FROM projets;
```

Étape 3 – Création de la procédure principale avec curseur :

```sql
DELIMITER //

CREATE PROCEDURE calculer_taux_utilisation(IN taux_seuil DECIMAL(5,2))
BEGIN
    -- Déclaration des variables
    DECLARE v_id INT;
    DECLARE v_budget DECIMAL(10,2);
    DECLARE v_depenses DECIMAL(10,2);
    DECLARE v_taux DECIMAL(5,2);
    DECLARE fin BOOLEAN DEFAULT FALSE;

    -- Déclaration du curseur
    DECLARE cur_projets CURSOR FOR
    SELECT id_projet, budget, depenses FROM projets;
```

```
    -- Gestion de la fin du curseur
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;

    -- Ouverture du curseur
    OPEN cur_projets;

    -- Boucle de lecture
    boucle_projets: LOOP
        FETCH cur_projets INTO v_id, v_budget, v_depenses;

        -- Condition de fin
        IF fin THEN
            LEAVE boucle_projets;
        END IF;

         -- Calcul du taux d'utilisation
         SET v_taux = (v_depenses / v_budget) * 100;

         -- Mise à jour selon le taux
         IF v_taux > taux_seuil THEN
             UPDATE projets
             SET taux_utilisation = v_taux, statut = 'Depassement'
             WHERE id_projet = v_id;

            -- Création d'une alerte
            INSERT INTO alertes (id_projet, message, date_alerte)
            VALUES (v_id, CONCAT('Depassement du seuil ', taux_seuil, ' % → ', v_taux, ' %'),
NOW());
        ELSE
            UPDATE projets
            SET taux_utilisation = v_taux, statut = 'Normal'
            WHERE id_projet = v_id;
        END IF;

        -- Enregistrement dans l'historique
        INSERT INTO historique (id_projet, taux, date_calcul)
        VALUES (v_id, v_taux, NOW());
    END LOOP;

    -- Fermeture du curseur
    CLOSE cur_projets;
END //

DELIMITER ;
```

Étape 4 – Exécution et tests :

```
CALL calculer_taux_utilisation(90);
SELECT * FROM projets;
SELECT * FROM alertes;
SELECT * FROM historique;
```

Étape 6 – Créer une fonction réutilisable :

```sql
DELIMITER //

CREATE FUNCTION calcul_taux(id_p INT)
RETURNS DECIMAL(5,2)
DETERMINISTIC
BEGIN
    DECLARE v_budget DECIMAL(10,2);
    DECLARE v_depenses DECIMAL(10,2);
    DECLARE v_taux DECIMAL(5,2);

    SELECT budget, depenses INTO v_budget, v_depenses
    FROM projets WHERE id_projet = id_p;

    IF v_budget > 0 THEN
        SET v_taux = (v_depenses / v_budget) * 100;
    END IF;

    RETURN v_taux;
END //

DELIMITER ;
```

Test :

```sql
SELECT nom, calcul_taux(id_projet) AS taux_calcule
FROM projets;
```

Étape 7 :

Nettoyage automatique :

```sql
DELIMITER //

CREATE PROCEDURE nettoyer_alertes()
BEGIN
    DELETE FROM alertes
    WHERE date_alerte < DATE_SUB(NOW(), INTERVAL 30 DAY);
END //

DELIMITER ;
```

Créer une vue globale :

```sql
CREATE VIEW vue_projets_alertes AS
SELECT p.nom, p.budget, p.depenses, p.taux_utilisation, p.statut,
a.message, a.date_alerte
FROM projets p
LEFT JOIN alertes a ON p.id_projet = a.id_projet;
```

Affichage global :

```sql
SELECT * FROM vue_projets_alertes;
```

3) Des captures d'écran prouvant l'exécution des requêtes :

Étape 1 – Création de la base et des tables

-- Création de la base

-- Table principale

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0007 seconde(s).)

```
CREATE TABLE projets ( id_projet INT AUTO_INCREMENT PRIMARY KEY, nom VARCHAR(100), budget DECIMAL(10,2), depenses DECIMAL(10,2), taux_utilisation DECIMAL(5,2), statut VARCHAR(20) );
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0007 seconde(s).)

```
-- Table secondaire : alertes CREATE TABLE alertes ( id_alerte INT AUTO_INCREMENT PRIMARY KEY, id_projet INT, message VARCHAR(255), date_alerte DATETIME );
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0008 seconde(s).)

```
-- Table historique : pour tracer les calculs CREATE TABLE historique ( id INT AUTO_INCREMENT PRIMARY KEY, id_projet INT, taux DECIMAL(5,2), date_calcul DATETIME );
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

Étape 2 – Insertion de données de test :

3 lignes insérées.
Identifiant de la ligne insérée : 3 (traitement en 0,0010 seconde(s).)

```
INSERT INTO projets (nom, budget, depenses) VALUES ('Projet Alpha', 10000, 8500), ('Projet Beta', 5000, 6000), ('Projet Gamma', 8000, 2000);
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

Affichage des lignes 0 - 2 (total de 3, traitement en 0,0008 seconde(s).)

```
SELECT * FROM projets;
```

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

	id_projet	nom	budget	depenses	taux_utilisation	statut
Éditer Copier Supprimer	1	Projet Alpha	10000.00	8500.00	NULL	NULL
Éditer Copier Supprimer	2	Projet Beta	5000.00	6000.00	NULL	NULL
Éditer Copier Supprimer	3	Projet Gamma	8000.00	2000.00	NULL	NULL

Tout cocher *Avec la sélection :* Éditer Copier Supprimer Exporter

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Étape 3 – Création de la procédure principale avec curseur :

```
CREATE PROCEDURE calculer_taux_utilisation(IN taux_seuil DECIMAL(5,2)) BEGIN -- Déclaration des variables DECLARE v_id INT; DECLARE v_budget DECIMAL(10,2); DECLARE v_depenses DECIMAL(10,2); DECLARE v_taux DECIMAL(5,2); DECLARE fin BOOLEAN DEFAULT FALSE; -- Déclaration du curseur DECLARE cur_projets CURSOR FOR SELECT id_projet, budget, depenses FROM projets; -- Gestion de la fin du curseur DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE; -- Ouverture du curseur OPEN cur_projets; -- Boucle de lecture boucle_projets:
```

```
Déclaration du curseur DECLARE cur_projets CURSOR FOR SELECT id_projet, budget, depenses FROM projets; -- Gestion de la fin du curseur DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE; -- Ouverture du curseur OPEN cur_projets; -- Boucle de lecture boucle_projets: LOOP FETCH cur_projets INTO v_id, v_budget, v_depenses; -- Condition de fin IF fin THEN LEAVE boucle_projets; END IF; -- Calcul du taux d'utilisation SET v_taux = (v_depenses / v_budget) * 100; -- Mise à jour selon le taux IF v_taux > taux_seuil THEN UPDATE projets SET tau[...]
```

Étape 4 – Exécution et tests :

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0304 seconde(s).)

```
CALL calculer_taux_utilisation(90);
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

Affichage des lignes 0 - 2 (total de 3, traitement en 0,0007 seconde(s).)

```
SELECT * FROM projets;
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

	id_projet	nom	budget	depenses	taux_utilisation	statut
Éditer Copier Supprimer	1	Projet Alpha	10000.00	8500.00	85.00	Normal
Éditer Copier Supprimer	2	Projet Beta	5000.00	6000.00	120.00	Depassement
Éditer Copier Supprimer	3	Projet Gamma	8000.00	2000.00	25.00	Normal

Étape 5 – Comprendre le rôle du curseur:

Élément	Rôle	Exemple SQL
CURSOR	Permet de lire chaque ligne du résultat d'une requête une par une	DECLARE cur_projets CURSOR FOR SELECT id_projet, budget, depenses FROM projets;
FETCH	Récupère les valeurs de la ligne courante dans des variables	FETCH cur_projets INTO v_id, v_budget, v_depenses;
HANDLER	Gère la fin du curseur pour éviter les erreurs quand il n'y a plus de lignes	DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;
LOOP	Exécute les opérations pour chaque ligne	boucle_projets: LOOP ... END LOOP;
LEAVE	Quitte la boucle proprement lorsque toutes les lignes ont été lues	IF fin THEN LEAVE boucle_projets; END IF;

Étape 6 – Créer une fonction réutilisable :

Afficher la zone SQL

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0237 seconde(s).)

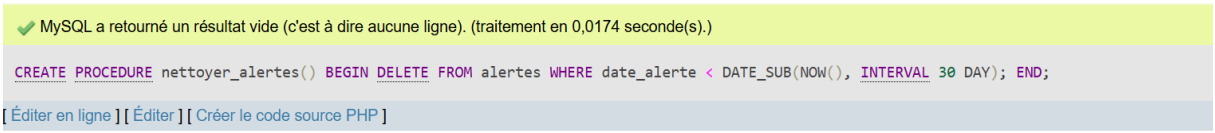
```
CREATE FUNCTION calcul_taux(id_p INT) RETURNS DECIMAL(5,2) DETERMINISTIC BEGIN DECLARE v_budget DECIMAL(10,2); DECLARE v_depenses
DECIMAL(10,2); DECLARE v_taux DECIMAL(5,2); SELECT budget, depenses INTO v_budget, v_depenses FROM projets WHERE id_projet = id_p; IF
v_budget > 0 THEN SET v_taux = (v_depenses / v_budget) * 100; END IF; RETURN v_taux; END;
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

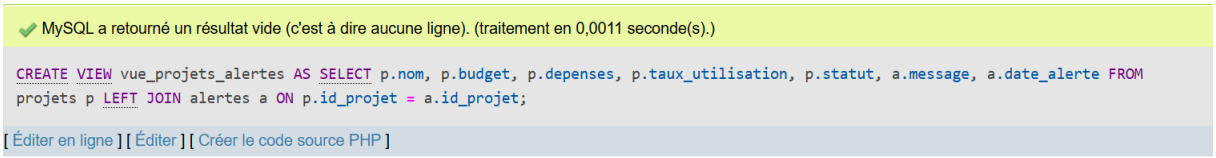
Test :

Étape 7 :

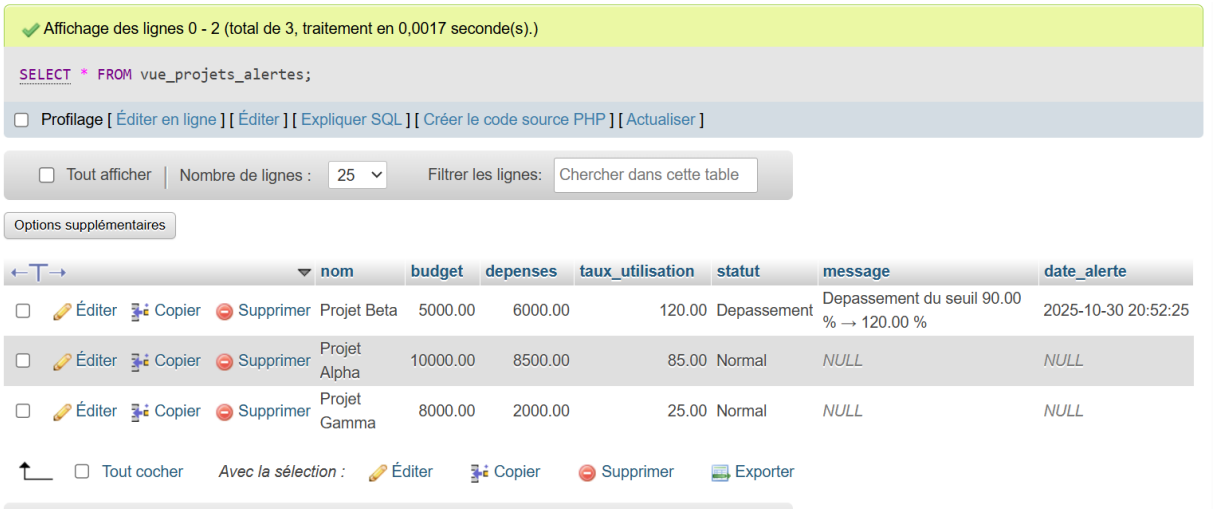
Nettoyage automatique :



Créer une vue globale :



Affichage global :



4) Une analyse des résultats:

- **Projet Alpha** : 85% → statut Normal, aucune alerte.
- **Projet Beta** : 120% → dépassement du seuil 90%, statut Depassement, alerte créée.
- **Projet Gamma** : 25% → statut Normal, aucune alerte.