La gestion des erreurs

Constatations

- Lorsque on travail avec Mysql on exécute des requêtes SQL simples à l'aide des commandes connues (SELECT, INSERT,UPDATE, …) ou même des programme qui contiennent ces instructions SQL. Et on remarque que suite à des mauvaises manipulation, des erreurs qui sont affichés et qui bloquent l'exécution de ces instructions.

Exemples d'erreurs mysql

- ***Exemple*** 1:

```
delimiter |
Create procedure maprocedure(IN ID int , IN NM varchar(20))
begin

INSERT INTO EMPLOYES (`N EMPLOYE`, NOM) VALUES(ID , NM);

end |
delimiter ;
```

- ***Exécution:***

```
Call maprocedure(1, 'DD' )
```

L'erreur produite est la suivante:

Error Code: 1062. Duplicate entry '1' for key

- ***Exécution:***

```
Call maprocedure(null, 'DD' )
```

l'erreur produite est la suivante:

Error Code: 1048. Column 'N employe' cannot be null

Gestion des erreurs en MySQL

- En MYSQL comme dans d'autres langages, on a la possibilité de gérer ces erreurs. Qu'on appel dans d'autres langages des ***exception*** mais en MYSQL on les appelle le gestionnaire d'erreur ou le **handler**.
- Afin d'éviter qu'un programme ne s'arrête dès la première erreur suite à une instruction SQL, il est indispensable de prévoir **les cas d'erreurs** et d'associer à chacun de ces cas la programmation d'une **exception** (**handler** dans le vocabulaire de MySQL).
- Les exceptions peuvent être gérées dans un sous-programme (fonction ou procédure cataloguée) ou un déclencheur.

Le concept des exceptions en MySQL

- Une exception est détectée si elle est prévue dans un **handler** au cours de l'exécution d'un **bloc** (entre **BEGIN** et **END**).
- Une fois levée, elle fait **continuer** (ou **sortir** du bloc) le programme après avoir réalisé une ou plusieurs instructions que le programmeur aura explicitement spécifiées.

Syntaxe globale d'un handler

- voici la syntaxe à utiliser pour créer un gestionnaire d'erreur :

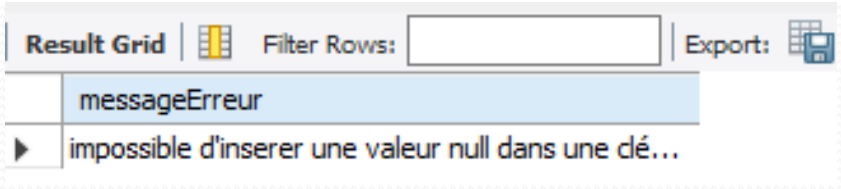
```
DECLARE { EXIT | CONTINUE } HANDLER FOR {
numero_erreur | { SQLSTATE identifiant_erreur } | condition }
instruction ou bloc d'instructions
```

- Un gestionnaire d'erreur définit une instruction (une seule !), ou un bloc d'instructions (**BEGIN** ... **END**), qui va être exécuté en cas d'erreur correspondant au gestionnaire.
- Tous les gestionnaires d'erreur doivent être déclarés au même endroit : après la déclaration des variables locales, mais avant les instructions de la procédure.
- Un gestionnaire peut, soit provoquer l'arrêt de la procédure (**EXIT**), soit faire reprendre la procédure après avoir géré l'erreur (**CONTINUE**).
- On peut identifier le **type d'erreur** que le gestionnaire va reconnaître de trois manières différentes : un **numéro d'erreur**, un **identifiant**, **ou une CONDITION**.

Exemple 1: Exception en utilisant le numéro d'erreur

```
delimiter |
create procedure pr1()
begin
declare coder int default 0;
     begin
       declare exit handler for 1048 set coder=1;
       insert into employes(`n employe`,nom) values(null,'Bahhar');
     end;
if coder=1 then
select 'impossible d''inserer une valeur null dans une clé primaire !' as messageErreur;
end if;
end |
delimiter ;
```

On exécute avec : **call pr1()**
Le résultat obtenu est :



Numéros des erreurs et identifiant d'erreurs

Code MySQL	SQLSATATE	Description
1048	23000	La colonne x ne peut pas être NULL
1169	23000	Violation de contrainte d'unicité
1216	23000	Violation de clé secondaire : insertion ou modification impossible (table avec la clé secondaire)
1217	23000	Violation de clé secondaire :suppression ou modification impossible (table avec la référence de la clé secondaire)
1172	42000	Plusieurs lignes de résultats alors qu'on ne peut en avoir qu'une seule
1242	42000	La sous-requête retourne plusieurs lignes de résultats alors qu'on ne peut en avoir qu'une seule

Pour plus d'informations sur ces numéros d'erreurs vous pouvez visiter le lien suivant: https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-reference-error-sqlstates.html

Exception en utilisant un identifiant d'erreur sqlstate

```
delimiter |
create procedure pr2()
begin
declare coder int default 0;
    begin
        declare exit handler for sqlstate '23000' set coder=1;
        insert into employes(`n employe`, nom) values (null,'Bahhar');
    end ;
    if coder=1 then
    select 'inpossible d''inserer une valeur null dans une clé primaire' as messageErreur;
    end if;

end |
delimiter ;
```

Après l'exécution avec **call pr2() on obtient le même résultat que le précédent exemple.**

Exception avec l'utilisation d'une CONDITION

- Avec un numéro d'erreur MySQL et un identifiant d'état SQL, il existe une troisième manière d'identifier les erreurs reconnues par un gestionnaire : une CONDITION.

- Une CONDITION est en fait simplement un **nom donné à un numéro d'erreur MySQL ou à un identifiant d'état SQL**. Cela vous permet de travailler avec des erreurs plus claires.
- Voici la syntaxe à utiliser pour nommer une erreur. Il s'agit à nouveau d'un **DECLARE**.
- Les déclarations de CONDITION doivent se trouver avant les déclarations de gestionnaires.

```
DECLARE nom_erreur CONDITION FOR { SQLSTATE
identifiant_SQL | numero_erreur_MySQL };
```

Exemple d'une exception avec l'utilisation d'une CONDITION

- Exemple:

```
delimiter |
create procedure pr2()
begin
declare coder int default 0;

declare probleme CONDITION for sqlstate '23000' ;

begin
declare exit handler for probleme set coder=1 ;
insert into employes(`n employe`, nom) values (null,'Bahhar');
end;

    if coder=1 then
    select 'inpossible d''inserer une valeur null dans une clé primaire' as messageErreur;
    end if;

end |
delimiter ;
```

Conditions prédéfinies

- Conditions prédéfinies Il existe trois conditions prédéfinies dans MySQL :

  - **SQLWARNING** : tous les identifiants SQL commençant par '01', c'est-à-dire les avertissements ;
  - **NOT FOUND** : tous les identifiants SQL commençant par '02', et que nous verrons plus loin avec les curseurs ;
  - **SQLEXCEPTION** : tous les identifiants SQL ne commençant ni par '00', ni par '01', ni par '02', donc les erreurs.

Exception avec l'utilisation de NOT FOUND

```
delimiter |
CREATE PROCEDURE Exemple (IN p_nom VARCHAR (200))
BEGIN
DECLARE flagNOTFOUND BOOLEAN DEFAULT 0 ;
DECLARE var1 VARCHAR(20) ;
    BEGIN
        DECLARE EXIT HANDLER FOR NOT FOUND SET flagNOTFOUND = -1;
        SELECT nom INTO var1 FROM clients WHERE nom = p_nom ;
        SELECT CONCAT ('Le seul client avec le nom ', p_nom, ' est ', var1) AS 'Resultat ps_exc_not_found_Exemple1';
    END ;
IF flagNOTFOUND=-1 THEN
SELECT CONCAT ('y a pas de client aven le nom ', p_nom) AS 'Resultat ps_exc_not_found_Exemple1' ;
END IF;
END;
END |
delimiter ;
CALL Exemple('Dalaj');
```

Exception avec **Continue** en utilisant **SQLEXCEPTION**

```
delimiter |
create procedure pro3()
begin
declare coder int default 0;
declare x int default 0;
begin
declare continue  handler for sqlexception set coder=1;
insert into employes(`n employe`,nom) values(null,'Bahhar');
set x=1;
end;
if coder=1 then
select 'impossible d''inserer une valeur null dans une clé primaire !' as messageErreur;
select x;
end if;
end |
delimiter ;

call pro3();    #x=1 dans le résultat
```

Déclarer plusieurs gestionnaires

- Un gestionnaire peut reconnaître plusieurs types d'erreurs différents. Par ailleurs, il est possible de déclarer plusieurs gestionnaires dans un même programme et celui détecté va être exécuté.