4. Les bases du langage de programmation

Le langage procédural de MySQL est une extension de SQL, car il permet de faire cohabiter les habituelles structures de contrôle avec des instructions SQL. Les avantages sont :

- la modularité : réutilisabilité d'un sous-programme
- la portabilité : un sous-programme est indépendant du système d'exploitation
- la sécurité : le sous-programme s'exécute dans un environnement à priori sécurisé

Structure d'un bloc

```
BEGIN
[ DECLARE déclaration ]
… code
END
```

Casse, identificateurs, commentaires

- insensible à la casse
- identificateur composé de caractères ∈ **['a'..'z', 'A'..'Z', '0'..'9', '_']**
- commentaire monoligne : « **- -** »
- commentaire multilignes commence par « **/*** » et finit par « ***/** »

Variables

```
DECLARE nomvariable1 [ ,nomvariable2 … ] typeMySQL
        [ DEFAULT expression ] ;
```

Affectation

```
SET variable := expression
```

la directive INTO d'une requête(pour extraire des données)

```
SELECT … INTO variable FROM …
```

Choix des noms de variables

- nommer toutes les variables en utilisant le préfixe « **v_** »

Opérateurs

- identiques à ceux utilisés dans les requêtes MySQL.

Variables de session (variables globales)

```
SET @variable = expression
```

Structures conditionnelles

```
IF condition THEN
    instructions;
END IF;
```

```
IF condition1 THEN
    instructions1;
ELSE
    Instructions2;
END IF;
```

```
IF condition1 THEN
    instructions1;
ELSEIF condition2
THEN
    instructions2;
ELSE
    Instructions3;
END IF;
```

Structures CASE

```
CASE variable
WHEN expr1 THEN instructions1;
WHEN expr2 THEN instructions2;
…
WHEN exprN THEN instructionsN;
[ ELSE instructionsN+1; ]
END CASE;
```

```
CASE
WHEN cond1 THEN instructions1;
WHEN cond2 THEN instructions2;
…
WHEN condN THEN instructionsN;
[ ELSE instructionsN+1; ]
END CASE;
```

Structure TANT QUE

```
WHILE condition DO
    instructions
END WHILE ;
```

Structure REPETER

```
REPEAT
    instructions
UNTIL condition
END REPEAT ;
```

5. La programmation avancée

- procédures stockées
- curseurs
- exceptions

5.1. Les procédures stockées

- déclaration des variables
- instructions
- éventuellement des exceptions

Création d'une procédure stockée

```
DELIMITER $$
CREATE PROCEDURE [nomBase.]nomprocédure (
        [ [ IN | OUT | INOUT ] param typeMySQL
        [ , [ IN | OUT | INOUT ] param2 typeMySQL ] ] … )
BEGIN
   [ DECLARE … ; ]
   bloc d'instructions
END;
$$
DELIMITER ;
```

DELIMITER : délimiteur de bloc de commandes différent de « ; » (symbole utilisé obligatoirement en fin de chaque déclaration et instruction du langage procédural)

Appel d'une procédure stockée

```
CALL [ nomBase.]nomprocédure
                ( [ paramètre [ , paramètre2 … ] ] ) ;
```

Suppression d'une procédure stockée

```
DROP PROCEDURE [ IF EXISTS ] [nomBase.]nomprocédure ;
```

La commande ALTER PROCEDURE est inutile. En effet, elle ne permet pas la modification des paramètres, du code.

Exercices

Ecrire les procédures stockées SQL-MySQL :

E511 : procédure **sp_caisse_market (↓ achat : entier ; ↓ règlement : entier)** ***Durée : 45 min.***

Vous disposez d'une caisse contenant des billets de 10 € et 5 € ainsi que des pièces de 2 € et 1 € .
Vous devez afficher la monnaie à rendre sur le règlement d'un achat en espèces.
Le nombre de pièces dans la caisse n'est pas limité.

Exemples

CALL sp_caisse_market(112,170);

billet de 10	billet de 5	pièce de 2	pièce de 1	résultat
6	1	1	1	5 billets de 10€, 1 billet de 5€, 1 pièce de 2€, 1 pièce de 1€

CALL sp_caisse_market(16,30);

billet de 10	billet de 5	pièce de 2	pièce de 1	résultat
1	0	2	0	1 billet de 10€, 2 pièces de 2€,

CALL sp_caisse_market(16,15);

billet de 10	billet de 5	pièce de 2	pièce de 1	résultat
NULL	NULL	NULL	NULL	vous n'avez pas assez d'argent, vous déposez vos achats et vous partez !

CALL sp_caisse_market(24,24);

billet de 10	billet de 5	pièce de 2	pièce de 1	résultat
0	0	0	0	Le compte est bon.

E512 : procédure **sp_palindrome (↓ phrase : chaine(200), ↑ resultat : booléen)** ***Durée : 30 min.***

La procédure affecte resultat à vrai si le contenu de phrase est un palindrome.
Un palindrome est un mot ou groupe de mots pouvant être lu indifféremment de gauche à droite ou de droite à gauche en conservant le même sens.

Exemples

CALL sp_palidrome(" Oh, cela te perd répéta l'écho ! " ,@resultat);

SELECT @resultat;

résultat
1

CALL sp_palidrome('bonjour',@resultat);

SELECT @resultat;

résultat
0