Les variables utilisateurs[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-variables-utilisateurs)

Il est possible de déclarer des variables de session propres à chaque utilisateur.

Ces variables globales commencent par le caractère @. Il existe des variables systèmes (qui sont globales aussi), et qui commencent par @@ (par exemple @@global.max\_connections).

Pour les déclarer, on utilise le mot clé SET ou encore SELECT accompagné de l'opérateur d'assignation :=.

In [6]:

%%sql

set @var = 12;

\* mysql+pymysql://notebook:\*\*\*@localhost/information\_schema?charset=utf8

0 rows affected.

Out[6]:

[]

In [7]:

%%sql

SELECT @var;

\* mysql+pymysql://notebook:\*\*\*@localhost/information\_schema?charset=utf8

1 rows affected.

Out[7]:

|  |
| --- |
| **@var** |
| 12 |

In [10]:

%%sql

SELECT @var2:= 15;

\* mysql+pymysql://notebook:\*\*\*@localhost/information\_schema?charset=utf8

1 rows affected.

Out[10]:

|  |
| --- |
| **@var2:= 15** |
| 15 |

La programmation sous MySQL[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#La-programmation-sous-MySQL)

Il est possible de créer des procédures et des fonctions.

Les procédures sont des blocs d'instructions qui ne renvoient pas de valeurs. Elles sont appellées avec le mot clé CALL.

Les fonctions quant à elles renvoient des valeurs et peuvent être appelées dans une requête (instruction SELECT par exemple).

Pour pouvoir déclarer (ou créer) des fonctions ou des procédures, il faut en priorité changer le délimiteur de fin d'instruction. En effet, le délimiteur par défaut est ;, et les instructions que contiennent une fonction ou une procédure se terminent par ce délimiteur.  
Pour cela, on utilise l'instruction suivante :

DELIMITER $$

Le délimiteur de fin d'instruction devient donc la chaîne $$.

Une fonction ou une procédure est créée grâce au mot clé CREATE.

DROP PROCEDURE IF EXISTS pr\_example;

DELIMITER $$

CREATE PROCEDURE example()

BEGIN

-- instructions

SELECT 'La procédure example a été appelée' AS message;

END $$

DELIMITER ;

Les paramètres d'une procédure peuvent être donnés :

* En lecture seule (IN) : la variable ne peut pas être modifiée par la procédure. Lors de l'appel, le système crée une copie locale, qui a pour portée la procédure.
* en modification seule (OUT) : la variable n'est destinée qu'à recevoir des valeurs. Si on tente de la lire dans la procédure, cette dernière a la valeur NULL. La modification de la valeur persiste après l'appel de la procédure.
* en lecture et modification (INOUT).

Définition d'une fonction[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#D%C3%A9finition-d'une-fonction)

La syntaxe est la suivante :

CREATE FUNCTION fct\_inc(liste\_paramètres) RETURNS type\_retour

[caractéristiques]

BEGIN

-- instructions

RETURN valeur\_retour;

END;

Les fonctions, contrairement aux procédure, ne prennent que des paramères en lecture seule, qui ne peuvent pas être modifiés par le corps de la fonction. On ne précise donc pas le type de paramètre (**IN**, OUT, INOUT).

Les caractéristiques des fonctions[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-caract%C3%A9ristiques-des-fonctions)

LANGUAGE SQL | [NOT] DETERMINISTIC | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA } | SQL SECURITY { DEFINER | INVOKER }

* DETERMINISTIC : indique que la valeur renvoyée par la fonction est toujours la même lorsque les paramètres d'appels sont les mêmes.

Les variables[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-variables)

Les variables locales d'une procédure ou d'une fonction doivent être déclarées (dans le corps de la fonction entre les mot-clés BEGIN et END ):

DECLARE somme INT DEFAULT 0;

SET somme = somme + 1 ;

Les types de données[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-types-de-donn%C3%A9es)

* INT :
* CHAR(lg) : défini une chaîne de caractère de longueur fixe (lg est la longueur à indiquer)
* VARCHAR(lg) : défini

Affectation de variables[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Affectation-de-variables)

Le mot clé SET sert à affecter les variables. Il est aussi possible d'utiliser la clause SELECT INTO :

DECLARE nb\_emp INT DEFAULT 0;

SELECT COUNT(\*) INTO nb\_emp FROM employees;

La requête ne doit renvoyer qu'une seule ligne de résultat.

Les structures de contrôle[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-structures-de-contr%C3%B4le)

Les structures conditionnelles[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-structures-conditionnelles)

On peut utiliser le IF tout seul :

IF condition THEN

-- instruction(s)

END IF;

On peut aussi utiliser un bloc ELSE qui est executé si la condition du IF est fausse.

IF condition THEN

-- instruction(s)

ELSE

-- autre(s) instruction(s)

END IF;

Enfin, on peut utiliser des conditions ELSEIF qui ne sont executées que si les conditions précédentes sont fausses et que la condition du ELSEIF est vérifiée.

IF condition THEN

-- instruction(s)

ELSEIF autres\_conditions THEN

-- autre(s) instruction(s)

--...

ELSE

-- instruction(s)

END IF;

CASE WHEN THEN DEFAULT END CASE

CASE[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#CASE)

Syntaxe :

CASE valeur\_à\_tester

WHEN valeur1 THEN --Instructions

WHEN valeur2 THEN -- Instructions

[ELSE --Instructions du else]

END CASE;

Au cas où l'on a un test de valeur non prévue et que lo'n souhaite avoir une instruction ELSE vide, vous pouvez créer un bloc vide dans le ELSE. Sinon, le système renverra une erreur.

CASE valeur\_à\_tester

WHEN valeur1 THEN --Instructions

WHEN valeur2 THEN -- Instructions

ELSE

BEGIN

END;

END CASE;

Les curseurs[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-curseurs)

Ils sont utilisés dans des fonctions, procédures ou encore triggers.

Ils permettent le parcours itératif de ligne de résultat de requête.

Utilisation[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Utilisation)

Pré-requis : les variables suivantes sont nécessaires à l'utilisation du curseur présenté après :

DECLARE fin INT default 0;

DECLARE name VARCHAR(50);

DECLARE first\_name VARCHAR(50);

DELACRE birth DATE;

Tout d'abord, les curseurs doivent être déclarés :

DECLARE nom\_curseur CURSOR FOR

SELECT nom, prenom, date\_naissance

FROM adherent

Une fois le curseur déclaré, il faut définir la condition d'arrêt, qui se fait sur une erreur. On affecte une variable qui nous servira de condition d'arrêt lors du parcours séquentiel des enregistements.

DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = 1;

Maintenant, le curseur est prêt à être ouvert :

OPEN nom\_curseur;

On peut maintenant parcourir séquentiellement les enregistrements. Les enregistrements sont parcourus dans l'ordre renvoyé par le select lors de la déclaration du curseur. Le parcours se fait obligatoirement du premier au dernier enregistrement : il n'est pas possible de placer le curseur où l'on veut. Par ailleurs, sous Mysql les curseurs sont en lecture seule. On ne peut pas modifier les données. En revanche, si les données sont modifiées ailleurs, le curseur lira les valeur modifiées. Il n'y a donc pas de verrou de ligne en lecture lors des manipulations de curseurs.

Exemple de boucle de parcours :

REPEAT

FETCH nom, prenom, date\_naissance INTO name, first\_name, birth ;

-- Manipulation : j'insère dans une table les valeurs concatenées de nom, prenom, date

INSERT INTO tbl\_id\_adh values (CONCAT(name, first\_name, birth));

UNTIL fin END REPEAT;

On peut enfin fermer le curseur :

CLOSE nom\_curseur;

Avant l'ouverture du curseur, on peut vérifier l'existence de la table tbl\_id\_adh ou sino la créer :

CREATE TABLE IF NOT EXISTS tbl\_id\_adh(id varchar(256) primary key);

Les triggers[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-triggers)

Les triggers sont des déclencheurs, réagissant à des évenements suivants :

* INSERT
* UPDATE
* DELETE Les triggers peuvent être déclenchés avant (BEFORE) ou après (AFTER) l'évenement en question.

Syntaxe :

CREATE TRIGGER mon\_trigger {BEFORE|AFTER} {INSERT|UPDATE|DELETE} ON nom\_table FOR EACH ROW

BEGIN

-- Traitement

END;

Dans le corps du trigger, on va avoir à notre disposition 2 pointeurs :

* OLD : contient la ligne de la table avant modification
* NEW : contient les nouvelles valeurs.

On peut accéder aux valeurs avec la notation pointée : IF OLD.nom = NEW.nom THEN ....

Exemple :[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Exemple-:)

Le règlement de la bibliothèque précise que le montant de cotisation ne peut jamais dépasser 100 €.

DELIMITER $$

CREATE TRIGGER trop\_cher BEFORE INSERT ON cotisation FOR EACH ROW

BEGIN

IF NEW.montant > 100 THEN

SET NEW.montant = 100;

END IF;

END $$

DELIMITER ;

Les erreurs et exceptions[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-erreurs-et-exceptions)

Les gérer[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-g%C3%A9rer)

Les erreurs sont gérer par un HANDLER. La syntaxe est la suivante :

DECLARE action HANDLER FOR valeur\_de\_condition instruction;

Action peut avoir les valeurs suivante :

* EXIT: le bloc contenant la déclaration du gestionnaire est interrompu.
* CONTINUE : l'exécution du bloc BEGIN ... END continue.

valeur\_de\_condition peut être :

* un code d'erreur MySQL
* une valeur SQLSTATE, ou SQLWARNING, ou NOTFOUND (pour les curseurs) ou SQLEXCEPTION.

Un petit exemple générique :

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET error = 1;

La variable error doit être déclarée au préalable.

DECLARE EXIT HANDLER FOR 1062

BEGIN

SELECT 'Clé primaire dupliquée' as message\_erreur;

END

1062 est le code d'erreur qui correspond à une clé primaire dupliquée.

Les générer[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#Les-g%C3%A9n%C3%A9rer)

Générer des erreurs[¶](https://moodle.ynov.com/mod/page/view.php?id=14211#G%C3%A9n%C3%A9rer-des-erreurs)

La génération d'erreur se fait à l'aide de l'instruction SIGNAL; Syntaxe :

SIGNAL SQLSTATE | nom\_condition;

nom\_condition doit avoir été déclarée au préalable.

Pour propager une exception, on utilise l'instruction RESIGNAL.

In [ ]: