

## Gestion de données

- Procédures stockées
- Les fonctions
- Les triggers
- Les curseurs
- MongoDB (Nosql)

## Méthode Agile

## Instructions de base de la Programmation procédurale dans MySQL

### Déclaration de variables

On peut déclarer une variable de deux façons dans MySQL.

La première méthode consiste à utiliser l'opérateur SET.

```
SET @nbr = 3, @y = 3 + 2, @s = @x % @y ;  
Declare x int
```

On peut également consister une variable à partir d'une requête.

```
SELECT @nb:= COUNT(*) FROM `ma_table` ;
```

**Note** : dans ce cas là, on préfixe l'opérateur = de : (deux points).

Ou encore

```
select COUNT(*)  
  
into @nb_total  
  
FROM ville;  
  
SELECT @nb_total;
```

### Instruction de condition

Syntaxes:

La structure IF ... THEN ... ELSE ... END IF; **ne fonctionne que dans les procédures stockées** (contenant plusieurs requêtes). Pour gérer une condition en dehors d'elles, on peut utiliser: IF(condition,

siVraie, siFausse);.

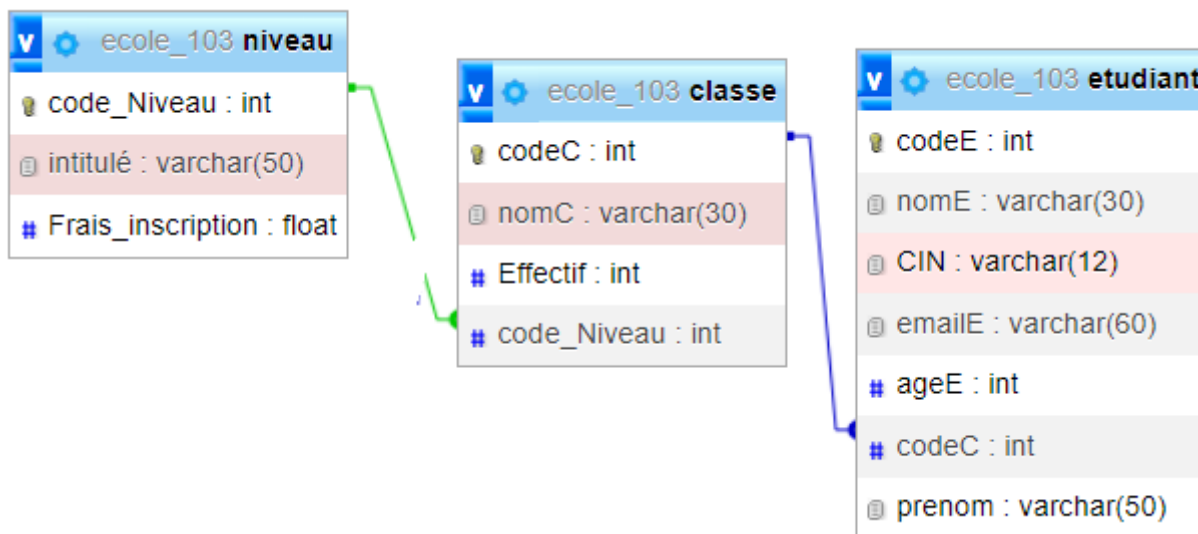
Exp 1:

```

DELIMITER $$
create procedure proc1()
begin
    set @x=10;
    IF @x > 0 THEN
        select concat(@x , ' est positif');
    ELSE
        select concat(@x , ' est negatif');
    end IF;
end$$

```

Soit la base de données

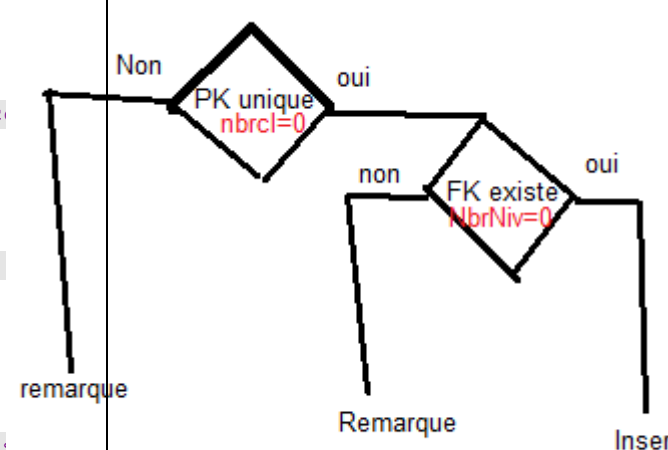


exp 2: Lister les classes

|  |             |
|--|-------------|
| <pre> Delimiter \$\$ create procedure pr1() BEGIN     declare x int;     set x=0;     select * from classe; END\$\$ </pre> | Call pr1(); |
|--|-------------|

exp 3: Ajouter les classes

|  |                            |
|--|----------------------------|
| <pre> Delimiter \$\$ drop procedure pr3; \$\$ create procedure pr3(cc int, nc varchar(30), eff int, cn int) BEGIN     insert into classe values (cc, nc, eff, cn); end; </pre>   |                            |
| <pre> create procedure pr3(cc int, nc varchar(30), eff int, cn int) BEGIN     declare nbr int;     select count(*) into nbr from classe where codeC=cc;     if nbr=0 then </pre> | Vérifier que PK est unique |

|   |   |
|---|---|
| <pre> insert into classe values(cc,nc,eff , cn); ELSE select concat('la classe ', cc, ' Existe déjà') as 'remarque'; end if; end\$\$ </pre>   |   |
| <pre> create procedure pr333(cc int, nc varchar(30),eff int, cn int) BEGIN declare nbrcl int; declare nbrNiv int; select count(*) into nbrcl from classe where codeC=cc; if nbrcl=0 then select count(*) into nbrNiv from niveau where code_Niveau =cn; if nbrNiv=0 THEN select concat('Code niveau ',cn, ' n existe pas 'remarque'; ELSE insert into classe values(cc,nc,eff , cn); end if; ELSE select concat('la classe du code : ', cc, ' Exis déjà') as 'remarque'; end if; </pre> | <p>Vérifier que FK existe</p>  <pre> graph TD     A{PK unique<br/>nbrcl=0} -- Non --&gt; B[remarque]     A -- oui --&gt; C{FK existe<br/>NbrNiv=0}     C -- non --&gt; D[Remarque]     C -- oui --&gt; E[Insert] </pre> |

Créer une procédure pour supprimer une classe

|   |  |
|---|--|
| <pre> create procedure pr4(cc int) BEGIN declare nbrcl int; select count(*) into nbrcl from classe where codeC=cc; if nbrcl=0 then select concat('la classe du code : ', cc, ' N''Existe pas') as 'remarque'; ELSE delete from classe where codeC=cc; end if; end; </pre>   |  |
| <pre> Delimiter \$\$ create procedure pr44(cc int) BEGIN declare nbrcl int; DECLARE nbrEt int; select count(*) into nbrcl from classe where codeC=cc; if nbrcl=0 then select concat('la classe du code : ', cc, ' N''Existe pas') as 'remarque'; ELSE select count(*) into nbrEt from etudiant where codeC=cc; if nbrEt=0 THEN delete from classe where codeC=cc; ELSE select concat('La classe : ', cc , ' n''est pas vide') as 'remarque'; end if; end if; end\$\$ </pre> | <p>Vérifier que la classe ,ne contient pas des étudiants</p> |

|  |  |
|--|--|
|  |  |
|--|--|

Créer une procédure pour modifier les données une classe :

|   |   |
|---|---|
| <pre>create procedure pr5(cc int, nc varchar(30) , eff int , cn int) BEGIN update classe set nomC=nc , Effectif=eff , code_Niveau=cn where codeC=cc; end;</pre>   |   |
| <pre>DELIMITER \$\$ create procedure pr55(cc int, nc varchar(30) , eff int , cn int) BEGIN  declare nbrCl int; declare nbrniv int; select count(*) into nbrCl from classe where codeC=cc; if nbrcl=0 THEN     select concat('La classe numéro : ' ,cc , ' n"exist pas' ) as 'Remarque'; ELSE     select count(*) into nbrNiv from niveau where code_Niveau=cn;     if nbrNiv=0 THEN         select concat('niveau avec le code : ' ,cn , 'n"existe pas' ) as 'remarque';     ELSE         update classe set nomC=nc , Effectif=eff , code_Niveau=cn         where codeC=cc;     end if; end if; end\$\$</pre> | <p>Vérifier que la classe (PK) existe, pour pouvoir la modifier</p> <p>Vérifier que le niveau (FK) existe</p> |
|   |   |
|   |   |

Pour appeler la procédure:

Call proc1()

Exercice :

Modifier la procédure stockée de façon à ce qu'elle possède x comme paramètre et non comme variable locale.

TP1:

Créer une table ville( codev , nomv , NBRHabitant)

Créer une procédure stockée, qui reçoit le code de la ville en paramètre, puis affiche le nombre d'habitants et une remarque

Petite ville si le nombre d'habitant est <2000

Grande ville si le nombre d'habitant est >=2000

```

delimiter $$

create procedure proc3( c int)

begin

set @nbr=0;

set @r='';

select NBRHabitant

into @nbr

from ville

where codev=c;

if @nbr<2000 then

    set @r= 'est une petite ville';

else

    set @r= 'est une grande ville';

end if;

select concat(@nbr,@r);

end$$

```

TP2:

Sur la même table ville( codev , nomv , NBRHabitant)

Créer une procédure stockée, qui reçoit le code de la ville en paramètre, puis affiche le code , le nom, le nombre d'habitants et une remarque :

Petite ville si le nombre d'habitant est <20000

Grande ville si le nombre d'habitant est >=2000

```

DROP PROCEDURE `if3`//# MySQL a retourné un résultat vide (aucune ligne).

```

```

# MySQL a retourné un résultat vide (aucune ligne).

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `if3`(x int)

begin

select nbrhabitant ,nomv

```

```
into @nbr, @nv

from ville where codev= x;

IF @nbr > 2000 THEN

    select concat( x , ' ', @nv, ' ', ' grande ville');

ELSE

    select concat( x , ' ', @nv, ' ', ' petite ville');

end IF;

end
```

Exp2 :

```
DELIMITER $$
create procedure pro_if(v int)
begin
set @x=' ';
IF @v=1 THEN
    set @x='Lundi';
ELSEIF v=2 THEN
    set @x='MArdi';
ELSEIF v=3 THEN
    set @x='Mercredi';
ELSEIF v=4 THEN
    set @x='jeudi';
ELSEIF v=5 THEN
    set @x='Vendredi';
ELSEIF v=6 THEN
    set @x='Samedi';
ELSEIF v=7 THEN
    set @x='Dimanche';
ELSE
    set @x='erruer';
END if;
SELECT @v,@x;
END$$
```

```

DELIMITER $$

create procedure case1(v int)

begin

select v,

case

    when v=1 then 'lundi'

    when v=2 then 'Mardi'

    when v=3 then 'Mercredi'

    when v=4 then 'Jeudi'

    when v=5 then 'vendredi'

    when v=6 then 'Smedi'

    when v=7 then 'dimanche'

    Else 'erreur'

end;

END$$

```

#### Délimiteur

MySQL utilise un caractère comme délimiteur pour séparer ses requêtes, par défaut ';'. Quand on crée des procédures stockées avec plusieurs requêtes, on en crée en fait une seule : CREATE de la procédure. Toutefois, si elles sont séparées par ';', il faut demander à MySQL de les ignorer pour estimer la fin du CREATE

L'instruction case :

TP3 :

Créer une procédure stockée qui reçoit le mois en paramètre puis affiche le nom du mois

#### L'instruction case :

**SELECT CASE WHEN** condition **THEN** siVraie **ELSE** siFausse **END**;

Exemple : SELECT CASE WHEN '-1 < 0' THEN 0 ELSE 1 END; renvoie 0.

Exemple avec plusieurs conditions :

**CASE v**

```

WHEN 2 THEN SELECT v;
WHEN 3 THEN SELECT 0;
ELSE
  BEGIN
  END;
END CASE;

```

Exp1 :

```

set @v=2;
SELECT @v,
CASE @v
  WHEN 1 THEN 'Lundi'
  WHEN 2 THEN 'MArdi'
  WHEN 3 THEN 'Mercredi'
  WHEN 4 THEN 'jeudi'
  WHEN 5 THEN 'Vendredi'
  WHEN 6 THEN 'Samedi'
  WHEN 7 THEN 'Dimanche'
  ELSE 'erruer'
END;

```

TP4:

Créer une procédure stockée qui reçoit le jour en paramètre, puis affiche le nom du jour

TP5 :

Créer une procédure stockée qui reçoit le mois en paramètre, puis affiche le nom du mois

**TP6 :**

Créer la table étudiant (cin, nom prenom, moyenne)

Créer la procédure stockée qui affiche les étudiants triés par moyenne, avec une décision :

Éliminé si moyenne < 8

Redoublant si moyenne entre 8 et 9

Racheté si moyenne entre 9 et 10

Réussi si moyenne ≥ 10

NB : utiliser l'instruction case when

TP7:

Créer une table ville( codev , nomv , NBRHabitant)

Créer une procédure stockée, qui reçoit le code de la ville en paramètre, puis affiche le nombre d'habitants et une remarque

Petite ville si le nombre d'habitant est < 20000

Grande ville si le nombre d'habitant est ≥ 20000

TP8:

Sur la même table ville( codev , nomv , NBRHabitant)



Créer une procédure stockée, qui reçoit le code de la ville en paramètre, puis affiche le code , le nom, le nombre d'habitants et une remarque :

Petite ville si le nombre d'habitant est <20000

Grande ville si le nombre d'habitant est >=2000

