**Les procédures stockées**

1. Définition et déclaration

Une procédure stockée est une entité SQL indépendante possédant une partie déclaration, une partie traitement (ordres SQL) et échange avec son environnement un ensemble de paramètres d’entrée / sortie.

A l’instar des autres langages, on peut réaliser plusieurs appels de la procédure stockée.

Syntaxe :

|  |
| --- |
| delimiter $  create procedure nomProcedure ( liste d’arguments )  begin  # déclaration  # traitement  end $  delimiter ; |

Les procédures stockées ne sont pas associées à une seule table comme les triggers. On peut écrire autant de procédures que nous voulons et peut les appeler indéfiniment.

La liste des arguments peut être :

* paramètres d’entrée : IN : ce sont les données envoyées à la procédure pour réaliser le traitement.
* paramètres de sortie : OUT: ce sont les résultats fournis par la procédure à son environnement.
* paramètres d’entrée / sortie INOUT : ce sont des données reçues par la procédure, modifiées par celle-ci et renvoyée à l’environnement.

Donc la signature de la procédure devient :

|  |
| --- |
| create procedure nomProcedure (IN parametre1 type, …  OUT parametre2 type, …  INOUT parametre3 type , … ) |

Pour appeler une procédure, on doit mentionner son nom suivi par la liste des paramètres effectifs :

|  |
| --- |
| call nomProcedure (liste des paramètres effectifs ) ; |

Pour supprimer la procédure, on utilise drop procedure :

|  |
| --- |
| drop procedure nomProcedure ; |

1. Exemple de traitements avec des procédures

Soit la base de données suivante composée de trois tables :

|  |
| --- |
| drop database if exists stock\_Promo\_250\_23;  create database stock\_promo\_250\_23;  use stock\_promo\_250\_23;  create table categorie (  idcategorie int (3) not null auto\_increment,  libelle varchar(50),  primary key (idcategorie)  );  create table fournisseur (  idfournisseur int (3) not null auto\_increment,  nom varchar(50),  adresse varchar(50),  primary key (idfournisseur)  );  create table produit (  idproduit int (3) not null auto\_increment,  designation varchar(50),  prix float,  qte int (3),  tva float ,  idcategorie int(3) not null,  idfournisseur int(3) not null,  primary key (idproduit),  foreign key (idcategorie) references categorie (idcategorie),  foreign key (idfournisseur) references fournisseur (idfournisseur)  ); |

On réalise les insertions suivantes dans les trois tables :

|  |
| --- |
| insert into categorie values (null, "Alimentaire"),  (null, "Santé"), (null, "Jardinage"), (null, "Beauté");  insert into fournisseur values (null, "Carrefour", "Paris"),  (null, "Auchan", "Marseille"), (null, "Cora", "Lille");  insert into produit values (null, "Lait", 2, 12, 20, 1, 1),  (null, "Sucre", 1.10, 23, 20, 1, 2),  (null, "Pelle", 3.90, 34, 5.5, 3, 3),  (null, "Parfum", 48.67,4, 7.5, 4, 2 ); |

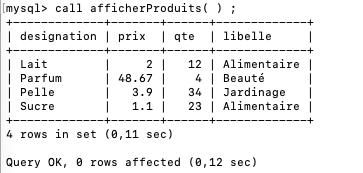
**Exemple 1 : Procédure sans paramètres**

Ecrire une procédure qui affiche la liste des produits (désignation, prix, qte et le libellé de la catégorie)

|  |
| --- |
| delimiter $  create procedure afficherProduits ( )  Begin  select p.designation, p.prix, p.qte, c.libelle  from produit p , categorie c  where p.idcategorie = c.idcategorie  order by p.designation ;  End $  delimiter ; |

Appel de la procédure :

|  |
| --- |
| call afficherProduits( ) ; |



Modifier la procédure pour afficher une deuxième liste avec les produits et les fournisseurs (désignation, prix, qte, nom du fournisseur).

* On supprime la procédure afficherProduits

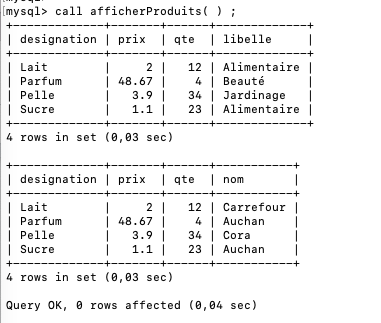
|  |
| --- |
| drop procedure afficherProduits ; |

* ON ré-écrit la procédure :

|  |
| --- |
| delimiter $  create procedure afficherProduits ( )  Begin  select p.designation, p.prix, p.qte, c.libelle  from produit p , categorie c  where p.idcategorie = c.idcategorie  order by p.designation ;    select p.designation, p.prix, p.qte, f.nom  from produit p , fournisseur f  where p.idfournisseur = f.idfournisseur  order by p.designation ;  End $  delimiter ; |

Appel de la nouvelle procédure :

|  |
| --- |
| call afficherProduits( ) ; |



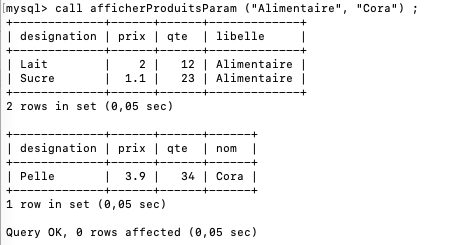
**Exemple 2 : Procédure avec des paramètres d’entrée**

Modifier la procédure afficherProduits en afficherProduitsParam pour qu’elle puisse recevoir deux entrées : la catégorie et le fournisseur.

|  |
| --- |
| delimiter $  create procedure afficherProduitsParam ( IN p\_libelle varchar(50), IN p\_nom varchar(50) )  Begin  select p.designation, p.prix, p.qte, c.libelle  from produit p , categorie c  where p.idcategorie = c.idcategorie  **and c.libelle = p\_libelle**  order by p.designation ;    select p.designation, p.prix, p.qte, f.nom  from produit p , fournisseur f  where p.idfournisseur = f.idfournisseur  **and f.nom = p\_nom**  order by p.designation ;  End $  delimiter ; |

Appel de la procédure avec deux paramètres effectifs : “Alimentaire” et “Cora”

|  |
| --- |
| call afficherProduitsParam ("Alimentaire", "Cora") ; |



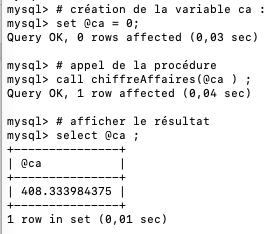
* **Exemple : procédure avec des paramètre de sortie**

Ecrire une procédure stockée qui permet de calculer le total du chiffre d’affaires : somme des produits (prix\*qte) en appliquant le taux de TVA. Ce total est renvoyé en résultat.

|  |
| --- |
| delimiter $  create procedure chiffreAffaires (OUT ca float)  Begin  select sum(prix \* qte \* ( 1+ tva/100 ) ) into ca from produit ;  End $  delimiter ; |

Appel de la procédure :

|  |
| --- |
| # création de la variable ca :  set @ca = 0;  # appel de la procédure  call chiffreAffaires(@ca ) ;  # afficher le résultat  select @ca ; |

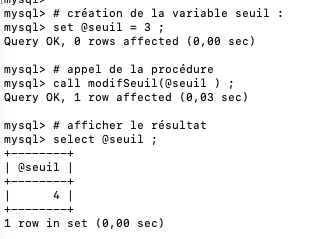


**Exemple 3 : Procédure avec paramètres d’entrée / sortie**

Ecrire une procédure modifSeuil qui reçoit un seuil de quantité et qui renvoie en sortie dans cette variable le nombre de produits avant une quantité plus grande que ce seuil.

|  |
| --- |
| delimiter $  create procedure modifSeuil ( INOUT seuil int(3))  Begin  select count(\*) into seuil  from produit  where qte > seuil ;  End $  delimiter ; |

|  |
| --- |
| # création de la variable seuil :  set @seuil = 3 ;  # appel de la procédure  call modifSeuil(@seuil ) ;  # afficher le résultat  select @seuil ; |



1. **L’héritage en SQL avec les procédures stockées**

Soit la base de données :

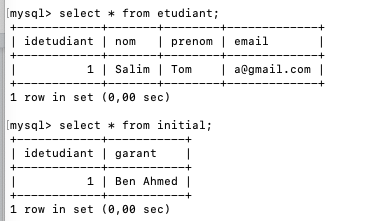
|  |
| --- |
| drop database if exists cfa\_heritage\_23 ;  create database cfa\_heritage\_23;  use cfa\_heritage\_23;  create table etudiant (  idetudiant int(3) not null auto\_increment,  nom varchar(50),  prenom varchar(50),  email varchar(50),  primary key (idetudiant)  );  create table initial (  idetudiant int(3) not null,  garant varchar(50),  primary key (idetudiant),  foreign key (idetudiant) references etudiant (idetudiant)  );  create table alternant (  idetudiant int(3) not null,  entreprise varchar(50),  salaire float (10.2),  primary key (idetudiant),  foreign key (idetudiant) references etudiant (idetudiant)  ); |

**Ecrire les deux procédures d’insertion d’un étudiant en initial et d’un étudiant en alternance en exploitant l’héritage.**

|  |
| --- |
| delimiter $  create procedure insertInitial (IN p\_nom varchar(50), IN p\_prenom varchar(50), p\_email varchar(50), IN p\_garant varchar(50) )  Begin  Declare p\_idetudiant int(3);    insert into etudiant values (null, p\_nom, p\_prenom, p\_email );  select idetudiant into p\_idetudiant  from etudiant  where nom = p\_nom and prenom=p\_prenom and email =p\_email ;  insert into initial values (p\_idetudiant, p\_garant);  End $  delimiter ; |

Appel de la procédure insertInitial

|  |
| --- |
| call insertInitial ("Salim", "Tom", "a@gmail.com", "Ben Ahmed"); |

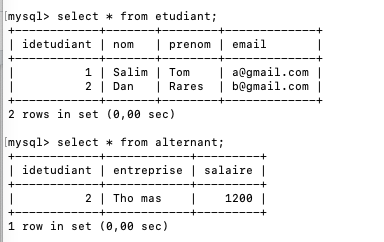


Réalisation de la deuxième procédure :

|  |
| --- |
| delimiter $  create procedure insertAlternant (IN p\_nom varchar(50), IN p\_prenom varchar(50), p\_email varchar(50), IN p\_entreprise varchar(50) , IN p\_salaire float)  Begin  Declare p\_idetudiant int(3);    insert into etudiant values (null, p\_nom, p\_prenom, p\_email );  select idetudiant into p\_idetudiant  from etudiant  where nom = p\_nom and prenom=p\_prenom and email =p\_email ;  insert into alternant values (p\_idetudiant, p\_entreprise,p\_salaire );  End $  delimiter ; |

Appel de la procédure insertInitial

|  |
| --- |
| call insertAlternant ("Dan", "Rares", "b@gmail.com", "Tho mas", 1200 ); |



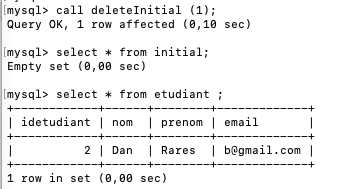
Réalisation des deux procédures de suppression d’un initial et d’un alternant :

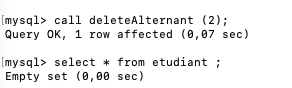
|  |
| --- |
| delimiter $  create procedure deleteInitial (IN p\_idetudiant int(3) )  Begin  delete from initial where idetudiant = p\_idetudiant ;  delete from etudiant where idetudiant = p\_idetudiant;  End $  delimiter ; |

|  |
| --- |
| delimiter $  create procedure deleteAlternant (IN p\_idetudiant int(3) )  Begin  delete from alternant where idetudiant = p\_idetudiant ;  delete from etudiant where idetudiant = p\_idetudiant;  End $  delimiter ; |

Appel des procédures :

|  |
| --- |
| call deleteInitial (1);  call deleteAlternant (2) ; |





**Les procédures stockées REVISION**

use cfa\_scolarite\_23;

DROP procedure if EXISTS afficheUser;

delimiter $

create procedure afficheUser()

begin

select iduser, nom, prenom, email, mdp, role from user;

end $

delimiter ;

CALL afficheUser();

use cfa\_scolarite\_23;

DROP procedure if EXISTS afficheUser;

delimiter $

create procedure afficheUser(IN unnom varchar(50))

begin

select iduser, nom, prenom, email, mdp, role from user where nom = unnom;

end $

delimiter ;

CALL afficheUser('Dan');

use cfa\_scolarite\_23;

DROP procedure if EXISTS afficheUser;

delimiter $

create procedure afficheUser(IN unnom varchar(50), OUT reponse varchar(255))

begin

select prenom into reponse from user where nom = unnom;

end $

delimiter ;

# création de la variable :

set @reponse = '';

CALL afficheUser('Dan', @reponse);

select @reponse;