

Nantes Université - IUT - BUT Informatique

**Rapport de SAE**

SQL Dans Un Langage De Programmation

eq\_02\_01

*OULMAS Thomas, ROUX Célestin, MOREAU–THOMAS Nils, VAN LAERE Pierric*

2ème année de BUT Informatique - SAE du 1er semestre

# 

# 

# Présentation

## Le sujet

Une société souhaite une refonte de son système d’information (SI) et l’automatisation

d’une partie du SI, avec principalement la création d’un site (ou application) de présentation et de ventes de produits en ligne. Le site devra être hébergé sur des serveurs. La société sera

responsable des serveurs et du déploiement de l’application sur ces derniers. L’ensemble devra

répondre aux contraintes de sécurité, de performance et aux aspects juridiques sur la protection

des données.

## Numéro de l’équipe

eq\_02\_01

## Membres de l’équipes

OULMAS Thomas : Étude sur la structure de la Base de Données

ROUX Célestin : Rédaction des procédures

VAN LAERE Pierric : Étude sur la structure de la Base de Données

MOREAU–THOMAS Nils : Rédaction des procédures

## Le client

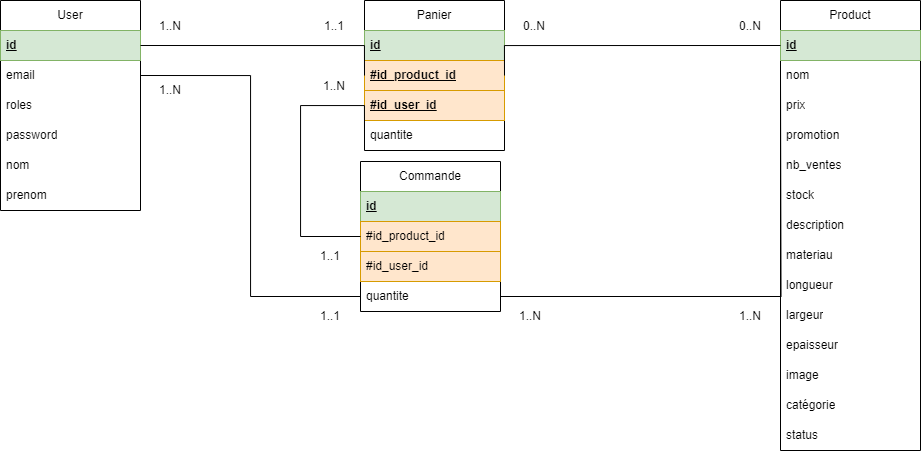
Le client de notre site se nomme Antoine, il possède une graveuse laser avec laquelle il grave des pièces sur mesures en bois. Mais il vend aussi des pièces uniques qu’il produit en parallèle des demandes de ses clients.

Ces pièces sont des dessous de verres, des boules de noël, des portes-clés et autres.

Il possède :

* un instagram : @antoinefactory
* une adresse mail : antoinefactory@gmail.com

# Les tables et leurs relations

**

*Disponible ici :*

[*https://drive.google.com/drive/folders/1yne7hV8fWDcLLMmfrpJ2FY6\_dixnO-3l?usp=sharing*](https://drive.google.com/drive/folders/1yne7hV8fWDcLLMmfrpJ2FY6_dixnO-3l?usp=sharing)

# Les procédures stockées

## Note

Notre site utilise mariaDB.

Il est important de noter que ces procédures ont été écrites à l’aide d’une documentation en ligne car la syntaxe diffère d’Oracle et certaines choses (comme le “LOOP”) n’ont pas été vu en cours.

Ces procédures ci-dessous n’ont pas été utilisées dans le site, puisque le framework Symfony les effectue déjà par le biais d'entités, qui une fois modifiées, modifient les ligne associées dans la base de données, et la logique de certaines ont déjà été implémentées en php.

Le fichier dans son entièreté est disponible ici : <https://drive.google.com/file/d/1C1WhqQkPJFTqTHmUTpnq50Rd3xDMfps-/view?usp=drive_link>

## Suppression d’un utilisateur

| /\*procédure n1\*/  CREATE OR REPLACE PROCEDURE supprimer\_user(IN user\_id INT)  BEGIN  DECLARE cnt INT;  SELECT COUNT(\*) INTO cnt FROM commande WHERE id\_user\_id = user\_id;  IF cnt > 0 THEN  SIGNAL SQLSTATE '45001'  SET MESSAGE\_TEXT = 'L utilisateur possede des commandes en cours';  ELSE  DELETE FROM panier WHERE id\_user\_id = user\_id;  DELETE FROM commande WHERE id\_user\_id = user\_id;  DELETE FROM user WHERE id = user\_id;  SELECT 'L utilisateur est supprimé';  END IF;  END;  /\*insert test1\*/  INSERT INTO user VALUES(999,'user@test.fr ','["ROLE\_USER"]','$2y$13$Km3pXs/NW6NFMr9JrdqZHOKij1SQJyhRctUECm2RAOEgZJa1AUPYW','Letest','Utilisateur');  /\*test1\*/  PREPARE stmt1 FROM 'CALL supprimer\_user(?)';  SET @a = 999;  EXECUTE stmt1 USING @a; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un utilisateur est supprimer

* si jamais il est dans la table commande, alors la suppression ne se fait pas
* sinon toutes les lignes contenant son id sont supprimer

## Suppression d’un articles

| /\*procédure n2\*/  CREATE OR REPLACE PROCEDURE supprimer\_article(IN product\_id INT)  BEGIN  DECLARE cnt INT;  SELECT COUNT(\*) INTO cnt FROM commande WHERE id\_product\_id = product\_id;  IF cnt > 0 THEN  SIGNAL SQLSTATE '45002'  SET MESSAGE\_TEXT = 'L article est contenu dans des commandes en cours';  ELSE  DELETE FROM panier WHERE id\_product\_id = product\_id;  DELETE FROM commande WHERE id\_product\_id = product\_id;  DELETE FROM product WHERE id = product\_id;  SELECT 'L article est supprimé';  END IF;  END;  /\*insert test2\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,1,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  /\*test2\*/  PREPARE stmt2 FROM 'CALL supprimer\_article(?)';  SET @a = 999;  EXECUTE stmt2 USING @a; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un article est supprimer

* si jamais il est dans la table commande, alors la suppression ne se fait pas
* sinon toutes les lignes contenant son id sont supprimer

## Ajout d’un article dans panier

| CREATE OR REPLACE PROCEDURE ajout\_panier(IN p\_product\_id INT, IN p\_user\_id INT, IN p\_quantite INT)  BEGIN  DECLARE pan\_item\_count INT;  /\*regarde si l article est déjà dans le panier de user\*/  SELECT COUNT(\*) INTO pan\_item\_count FROM panier WHERE id\_product\_id = p\_product\_id AND id\_user\_id = p\_user\_id;  IF pan\_item\_count > 0 THEN  /\*l article est déjà dans le panier de user\*/  UPDATE panier  SET quantite = quantite + p\_quantite  WHERE id\_product\_id = p\_product\_id AND id\_user\_id = p\_user\_id;  ELSE  /\*l article n’est pas déjà dans le panier de user\*/  INSERT INTO panier (id\_product\_id, id\_user\_id, quantite)  VALUES (p\_product\_id, p\_user\_id, p\_quantite );  END IF;  END;  /\*insert test3\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,1,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  INSERT INTO user VALUES(999,'user@test.fr ','["ROLE\_USER"]','$2y$13$Km3pXs/NW6NFMr9JrdqZHOKij1SQJyhRctUECm2RAOEgZJa1AUPYW','Letest','Utilisateur');  /\*test3\*/  PREPARE stmt3 FROM 'CALL ajout\_panier(?,?,?)';  SET @a = 999;  SET @b = 999;  SET @c = 5;  EXECUTE stmt3 USING @a,@b,@c; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un utilisateur ajoute au panier un article

* si jamais l’article est déjà dans son panier, alors la quantité dans le panier sera celle déjà dans le panier + celle que l'on ajoute
* sinon une ligne se crée avec l’article et l’utilisateur qui l’a commandé

## Passage de la commande d’un utilisateur

| CREATE PROCEDURE passage\_commande(IN p\_user\_id INT, IN adr VARCHAR, IN tel VARCHAR)  BEGIN  DECLARE product\_id, pan\_quantite, available\_stock INT;  DECLARE done BOOLEAN DEFAULT FALSE;  DECLARE pan\_cursor CURSOR FOR  SELECT p.id\_product\_id, p.quantite, pr.stock  FROM panier p  JOIN product pr ON p.id\_product\_id = pr.id  WHERE p.id\_user\_id = p\_user\_id;  DECLARE CONTINUE HANDLER FOR NOT FOUND  SET done = TRUE;  OPEN pan\_cursor;  pan\_loop: LOOP  FETCH pan\_cursor INTO product\_id, pan\_quantite, available\_stock;  IF done THEN  LEAVE pan\_loop;  END IF;  IF pan\_quantite > available\_stock THEN  SIGNAL SQLSTATE '45003' SET MESSAGE\_TEXT = 'La quantité commandée n est pas en stock';  ELSE  -- Ajout dans la table commande  INSERT INTO commande (id\_user\_id, id\_product\_id, adresse, telephone, quantite)  VALUES (p\_user\_id, product\_id, adr, tel, pan\_quantite);    -- Retire de la table panier  DELETE FROM panier WHERE p\_user\_id = p\_user\_id AND p\_product\_id=product\_id and;  -- Update product pour montrer la vente  UPDATE product  SET stock = stock - pan\_quantite,  nb\_ventes = nb\_ventes + pan\_quantite  WHERE id = product\_id;  END IF;  END LOOP;  CLOSE pan\_cursor;  END;  /\*insert test4\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,1,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  INSERT INTO user VALUES(999,'user@test.fr ','["ROLE\_USER"]','$2y$13$Km3pXs/NW6NFMr9JrdqZHOKij1SQJyhRctUECm2RAOEgZJa1AUPYW','Letest','Utilisateur');  PREPARE stmt4 FROM 'CALL ajout\_panier(?,?,?)';  SET @a = 999;  SET @b = 999;  SET @c = 5;  EXECUTE stmt4 USING @a,@b,@c;  /\*test4\*/  PREPARE stmt5 FROM 'CALL passage\_commande(?)';  SET @d = 999;  EXECUTE stmt5 USING @d; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un utilisateur passe un commande, on regarde chaque article de son panier

* si jamais la quantité dans le panier excède celle disponible alors l’article n’est pas commandé
* sinon il est commandé et le nombre de vente augmente + le stock disponible diminue

# Les triggers stockées

## La mise à jour du statut d’un article

Nous avons réalisé deux triggers (pour l’update et l’insert) pour le changement dynamique du statut d’un article :

| CREATE OR REPLACE TRIGGER update\_product\_status\_before\_insert  BEFORE INSERT ON product  FOR EACH ROW  BEGIN    IF NEW.stock > 0 AND NEW.stock < 11 THEN  SET NEW.status = 'low';  ELSEIF NEW.stock = 0 THEN  SET NEW.status = 'out';  ELSEIF (NEW.status = 'out' OR NEW.status='low') AND NEW.stock > 10 THEN  SET NEW.status = '';  END IF;  END;  CREATE OR REPLACE TRIGGER update\_product\_status\_before\_update  BEFORE UPDATE ON product  FOR EACH ROW  BEGIN  IF NEW.stock > 0 AND NEW.stock < 11 THEN  SET NEW.status = 'low';  ELSEIF NEW.stock = 0 THEN  SET NEW.status = 'out';  ELSEIF (OLD.status = 'out' OR OLD.status='low') AND NEW.stock > 10 THEN  SET NEW.status = '';  END IF;  END;  /\*insert test5\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,0,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  /\*le statut doit être out\*/  /\*update test5\*/  UPDATE INTO product SET stock=5 WHERE id = 999  /\*le statut doit être low\*/ |
| --- |

Le trigger en cas d’update d’un article fonctionne comme ceci :

On regarde chaque ligne qui sera modifié

* si le stock est entre 0 et 10, le statut devient “low” (bientôt épuisé)
* sinon si le stock est à 0, le statut devient “out” (épuisé)
* sinon si l’article était épuisé ou bientôt épuisé et que le nouveau stock est supérieur à 10, alors son statut est normal (soit “”)

Le trigger en cas d’insert d’un article fonctionne comme ceci :

On regarde chaque ligne qui sera modifié

* si le stock est entre 0 et 10, le statut devient “low” (bientôt épuisé)
* sinon si le stock est à 0, le statut devient “out” (épuisé)
* sinon si l’article est épuisé ou bientôt épuisé et que le stock est supérieur à 10, alors son statut est normal (soit “”)

Le différences entre les deux triggers c’est que l’insert ne prend pas en compte “l’ancien statut” mais celui actuelle, au cas où le client crée un produit déclaré comme épuisé mais avec du stock, alors le trigger corrige son erreur