Rapport SQL

Enzo Bertel

Felix Pautrel

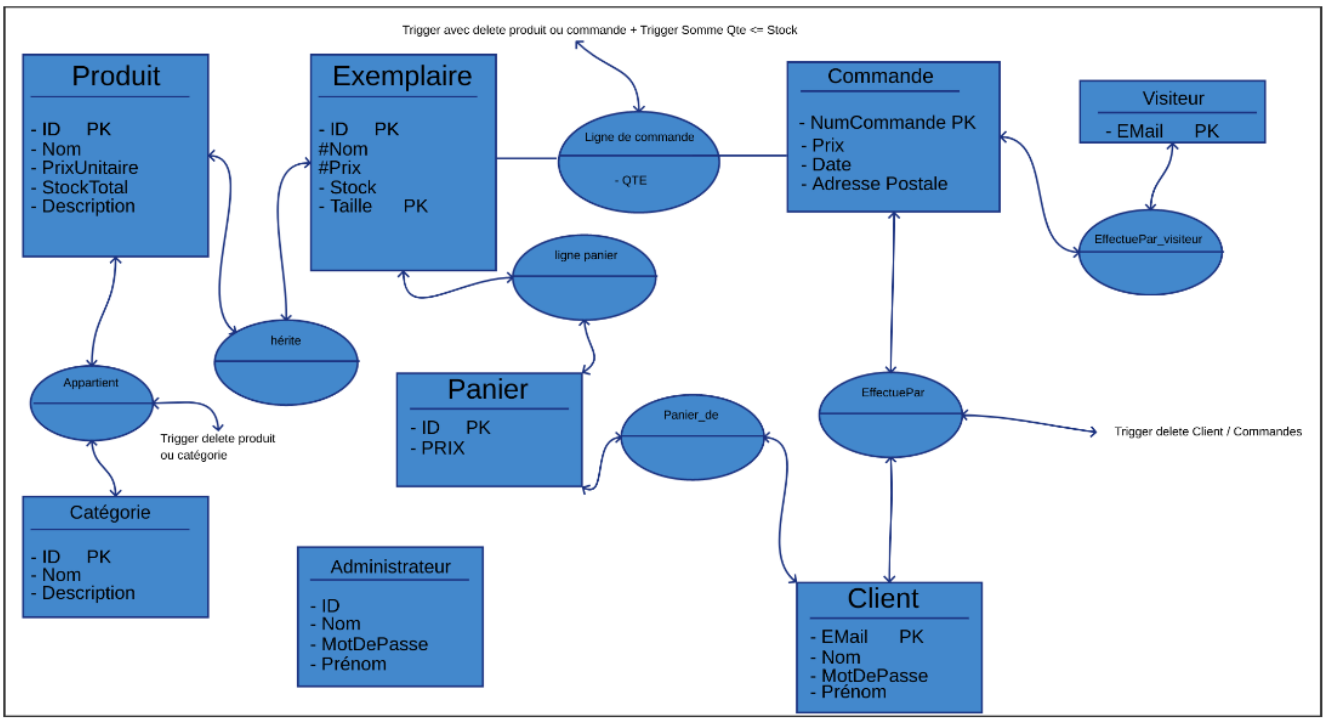
Luc Ndzamba

Arnaud Leboeuf

Alan Pierre Dit Hubert

Création des tables

SQL_tables



```sql
drop table if exists administrateur;
drop table if exists example;
drop table if exists messenger_messages;
drop table if exists appartient;
drop table if exists lignecommande;
drop table if exists effectue_par_client;
drop table if exists panier_de;
drop table if exists lignepanier;
drop table if exists visiteur;
drop table if exists categorie;
drop table if exists administrateur;
drop table if exists client ;
drop table if exists product;
drop table if exists panier;
drop table if exists commande;

create table product (
    id int primary key not null,
    nom varchar(25) ,
    prixunitaire decimal(10,2),
    stocktotal int,
    description text,
    imageurl varchar(500)
);

alter table product add fulltext index fulltext_index (nom, description);

create table commande (
    id int primary key not null,
    prix decimal(10,2),
    date_demande varchar(10),
    adresse_postale varchar(50)
);


create table lignecommande (
    id_product int,
    numcommande int,
    quantite int,
    foreign key (id_product) references product(id),
    foreign key (numcommande) references commande(id),
    primary key(id_product,numcommande)
);

create table categorie (
    id int primary key not null,
    nom varchar(20) unique,
```

```
    theme varchar(20),
    description varchar(100)
);

alter table categorie add fulltext index fulltext_index1 (nom,theme,
description);

create table appartient (
    id_product int,
    id_categorie int,
    foreign key (id_product) references product(id) ,
    foreign key (id_categorie) references categorie(id),
    primary key (id_categorie,id_product)
);

create table client (
    id int primary key not null,
    email varchar(50) unique,
    nom varchar(20),
    prenom varchar(20),
    motdepasse varchar(100)
);

insert into client
values(0,"test@gmail.com","test","test",'$2y$10$vvai6nrtl8evogwbgvzhje5888iyeqb
wsmebdkmcc2ikiaqbtefdo');

create table visiteur (
    id int primary key not null,
    email varchar(50) unique
);

create table administrateur (
   id int primary key not null,
   nom varchar(20),
   prenom varchar(20),
   motdepasse varchar(100)
);

insert into administrateur values(0,'adminsupreme','00','adminpass');

create table panier (
    id int primary key,
    prix int
);

create table effectue_par_client (
    id_commande int,
```

```
    id_client int,
    foreign key (id_client) references client(id),
    foreign key (id_commande) references commande(id),
    primary key (id_client,id_commande)
);

create table lignepanier(
    id_product int,
    id_panier int,
    quantite int,
    foreign key (id_product) references product(id),
    foreign key (id_panier) references panier(id),
    primary key (id_product,id_panier)
);

create table panier_de (
    id_client int,
    id_panier int,
    foreign key (id_panier) references panier(id),
    foreign key (id_client) references client(id),
    primary key (id_client,id_panier)
);
```

PRODUCT

Description

Cette table contient l'ensemble des produits du site.

Membres

- id (int primary key not null) : identifiant unique du produit
- nom (varchar(25)) : nom du produit
- prixunitaire (decimal(10,2)) : prix unitaire du produit
- stocktotal (int) : stock total disponible du produit
- description (text) : description détaillée du produit
- imageurl (varchar(500)) : URL de l'image du produit

Primary key

- id int primary key not null

COMMANDE

Description

Cette table contient les informations sur les commandes passées sur le site.

Membres

- id (int primary key not null) : identifiant unique de la commande
- prix (decimal(10,2)) : prix total de tous les produits commandés
- date_demande (varchar(10)) : date de la commande
- adresse_postale (varchar(50)) : adresse postale de livraison de la commande

Primary key

- id int primary key not null

LIGNECOMMANDE

Description

Table est liéé à COMMANDE. Pour chaque produit dans COMMANDE, il existe une LIGNECOMMANDE qui contient les informations sur le produit, comme la quantité de ce-dernier.

Membres

- id_product (int) : identifiant unique du produit dans la commande
- numcommande (int) : identifiant (non unique) lié à id de la table COMMANDE
- quantite (int) : quantité du produit incluse dans la commande

Primary key

- primary key(id_product,numcommande)

Foreign Keys

- foreign key (id_product) references product(id)
- foreign key (numcommande) references commande(id)

CATEGORIE

Description

Cette table contient les catégories des produits disponibles sur le site. Un produit peut appartenir a 0, 1 ou plusieurs catégories en même temps.

Membres

- id (int primary key not null) : identifiant unique de la catégorie
- nom (varchar(20) unique) : nom de la catégorie
- theme (varchar(20)) : thème de la catégorie
- description (varchar(100)) : description détaillée de la catégorie

Primary key

- id int primary key not null

APPARTIENT

Description

Cette Table permet de lier des produits et des catégories.

Membres

- id_product (int) : identifiant unique du produit
- id_categorie (int) : identifiant unique de la catégorie

Primary key

- primary key (id_categorie,id_product)

Foreign Keys

- foreign key (id_product) references product(id)
- foreign key (id_categorie) references categorie(id)

CLIENT

Description

La table CLIENT contient les informations sur les clients enregistrés sur le site.

Membres

- id (int primary key not null) : Chaque client a un identifiant unique 'id'
- email (varchar(50) unique) : Chaque client a un un email unique 'email'
- nom (varchar(20)) : nom du client
- prenom (varchar(20)) : prenom du client
- motdepasse (varchar(100)) : mot de passe (hashé) du client

Primary key

- id int primary key not null

VISITEUR

Description

Cette table correspond à un utilisateur non identifié sur le site.

Membres

- id (int primary key not null) : identifiant unique et clé primaire du visiteur
- email (varchar(50) unique) : identifiant unique du visiteur utilisé pour la commande

Primary key

- id int primary key not null

ADMINISTRATEUR

Description

La table ADMINISTRATEUR contient les informations sur les administrateurs enregistrés sur le site. Les administateurs sont disctincts des clients, ils ne peuvent pas commander de produits et ne sont pas lier à un panier. Ils ont des accès spéciaux aux pages admin.

Membres

- id (int primary key not null) : Chaque administrateur a un identifiant unique 'id'
- nom (varchar(20)) : nom de l'admin
- prenom (varchar(20)) : prenom de l'admin

- motdepasse (varchar(100)) : mot de passe (hashé) de l'admin

Primary key

- id int primary key not null

PANIER

Description

Chaque PANIER est lié à un client

Membres

- id (int primary key) : id unique du panier
- prix (int) : prix total des produits contenus dans le panier

Primary key

- id int primary key

EFFECTUE_PAR_CLIENT

Description

Cette table fait une correspondance entre une commande et un client

Membres

- id_commande (int) : identifiant de la commande, fait partie de la clé primaire
- id_client (int) : identifiant du client, fait partie de la clé primaire

Primary key

- primary key (id_client,id_commande)

Foreign Keys

- foreign key (id_client) references client(id)
- foreign key (id_commande) references commande(id)

LIGNEPANIER

Description

Table est liéé à PANIER. Pour chaque produit dans PANIER, il existe une LIGNEPANIER qui contient les informations sur le produit, comme la quantité de ce-dernier.

Membres

- id_product (int) : id unique du produit
- id_panier (int) : id unique du panier, lié à id de PANIER
- quantite (int) : quantité du produit

Primary key

- primary key (id_product,id_panier)

Foreign Keys

- foreign key (id_product) references product(id)
- foreign key (id_panier) references panier(id)

PANIER_DE

Description

Table liant le client à son panier

Membres

- id_client (int) : id unique du client
- id_panier (int) : id unique du panier

Primary key

- primary key (id_client,id_panier)

Foreign Keys

- foreign key (id_panier) references panier(id)
- foreign key (id_client) references client(id)

Création de procédures

AjoutCommande

```
drop procedure if exists ajoute_produit_commande;
delimiter &&
create procedure ajoute_produit_commande(le_numcommande int, le_idexemplaire
int,le_qte int)
begin
insert into lignecommande values (le_idexemplaire, le_numcommande,le_qte);
commit;
end &&
delimiter ;
```

Description

Cette prodécure permet d'ajouter un produit dans une commande d'un client avec ses attributs.

Membres

- le_numcommande (int) :
- le_idexemplaire (int) :
- le_qte (int) :

AjoutPanier

```
drop procedure if exists ajoute_produit_panier;

delimiter &&
create procedure ajoute_produit_panier(le_idpanier int, le_idexemplaire
int,le_qte int)
begin
declare id_panier int;
declare id_exemp int;

select id_product, id_panier into id_exemp, id_panier from lignepanier where
id_product = le_idexemplaire and id_panier = le_idpanier;

if id_panier = null then
    insert into lignepanier values (le_idexemplaire,le_idpanier,le_qte);
else
    update lignepanier set quantite = quantite+ le_qte where id_product =
le_idexemplaire and id_panier = le_idpanier;

end if;
end &&
delimiter ;
```

Description

Cette procédure ajoute un produit dans le panier d'un client. Le panier est créé s'il n'existe pas sinon le produit y est directement ajouté.

Membres

- le_idpanier (int) :
- le_idexemplaire (int) :
- le_qte (int) :

AjouterDansCategorie

```
drop procedure if exists productintocategorie;
delimiter &&
create procedure productintocategorie(le_id_product int,le_id_cat int)
begin
    select id_product into @id from appartient where id_categorie = le_id_cat
and id_product = le_id_product ;

    if @id is null then
        insert into appartient values (le_id_product,le_id_cat);
    end if;

end &&
delimiter ;
```

Description

Cette procédure ajoute un produit dans une catégorie s'il n'y est pas déjà. La procédure est réservé aux administrateurs.

Membres

- le_id_product (int) :
- le_id_cat (int) :

CategorieFromProduct

```
drop procedure if exists cat_fromprod;
delimiter &&
create procedure cat_fromprod(le_id int)
begin
drop table if exists categorie_temp;
    create table categorie_temp(
        id int primary key not null
    );

    insert into categorie_temp (id)
    select id_categorie from appartient where id_product = le_id;

end &&
delimiter ;
```

Description

Cette procédure créer une table temporaire contenant tous les identifiants des catégories du produit donné en paramètre afin de les récupérer dans le code après.

Membres

- le_id (int) :

CreerPanier

```
drop procedure if exists creer_panier;
delimiter &&

create procedure creer_panier(le_idpanier int ,le_id int)
begin
insert into panier values (le_idpanier,0);

insert into panier_de values (le_id,le_idpanier);

end &&
delimiter ;
```

Description

Cette procédure permet de créer un panier appartenant à un client.

Membres

- le_idpanier (int) :
- le_id (int) :

CreerProduit

```
drop procedure if exists creer_produit;
delimiter &&
create procedure creer_produit(le_id int, le_nom varchar,le_prix
decimal,le_stock int,le_descr text,le_imgurl varchar)
begin
insert into product values (le_id, le_nom,le_prix,le_stock,le_descr,le_imgurl);
commit;
end &&
delimiter ;
```

Description

Cette procédure permets de créer un produit, elle est réservé aux administrateurs.

Membres

- le_id (int) :
- le_nom (varchar) :
- le_prix (decimal) :
- le_stock (int) :
- le_descr (text) :
- le_imgurl (varchar) :

ProduitsFromCategorie

```
drop procedure if exists productfromcategorie;
delimiter &&
create procedure productfromcategorie(le_id int)
begin
drop table if exists product_temp;

    create table product_temp(
        id int primary key not null
    );


    insert into product_temp (id)
    select id_product from appartient where id_categorie = le_id;

end &&
delimiter ;
```

Description

Cette procédure récupère tous les produits d'une catégorie via une table temporaire qui sera utilisé plus tard dans le code.

Membres

- le_id (int) :

ProduitsFromPanier

```
drop procedure if exists productfrompanier;
delimiter &&
create procedure productfrompanier(le_id int)
begin
drop table if exists product_temp;

    create table product_temp(
        id int primary key not null,
        quantite int
    );


    insert into product_temp (id,quantite)
    select id_product,quantite from lignepanier where id_panier = le_id;

end &&
delimiter ;
```

Description

Cette procédure récupère les produits d'un panier d'un client en les insérant dans une table temporaire qui sera utilisé plus tard dans le code.

Membres

- le_id (int) :

ProduitsFromSearchTerme

```
drop procedure if exists productfromsearchterm;
delimiter &&
create procedure productfromsearchterm(le_searchterm text)
begin
drop table if exists product_temp;

    create table product_temp(id int primary key not null);


    insert into product_temp (id)
    select id from product
          where match(nom, description) against (le_searchterm in boolean
mode);
    insert into product_temp (id)
    select id_product from appartient where id_product not in (select id from
product_temp) and id_categorie in (select id from categorie
          where match(nom, description, theme) against (le_searchterm in
boolean mode));

end &&
delimiter ;
```

Description

Cette procédure récupère les produits contenant le paramètre "le_searchterm" dans son nom ou sa description et les produits qui n'on pas déjà été ajouté qui ont une catégorie qui contient "le_searchterm" dans son nom, sa description ou son theme.

Membres

- le_searchterm (text) :

Création de triggers

TiggerStockQte

```
drop trigger if exists stock_qte;

delimiter &&

create or replace trigger stock_qte before insert on lignecommande
for each row
begin
    declare quantite_total int;
    declare stock_prod int;

    select sum(quantite)
    into quantite_total
    from lignecommande where id_product = new.id_product;

    select stocktotal into stock_prod from product where id = new.id_product;

    if quantite_total + new.quantite > stock then
        signal sqlstate '45001' set message_text = "la quantite demandée est
supérieure au stock disponible";
    end if;
end &&

delimiter ;
```

Description

Ce trigger vérifie avant l'ajout d'un produit dans la commande d'un client si la quantité choisis de produits est supérieure au stock disponible.

TriggerAjoutProduitCommande

```
drop trigger if exists ajout_commande;

delimiter &&

create or replace trigger ajout_commande after insert on lignecommande
for each row
begin
    declare prixproduit int;

    select prixunitaire into prixproduit from product where id =
new.id_product;
    update commande set prix = prix + (new.quantite* prixproduit) where
commande.id = new.numcommande;

    update product set stocktotal = stocktotal - new.quantite where id =
new.id_product;
end &&

delimiter ;
```

Description

Ce trigger mets à jour la commande d'un client après qu'il y ait ajouté un article.

TriggerAjoutProduitPanier

```
drop trigger if exists ajout_panier;

delimiter &&

create or replace trigger ajout_panier after insert on lignepanier
for each row
begin
    declare prixproduit int;

    select prixunitaire into prixproduit from product where id =
new.id_product;
    update panier set prix = prix + (new.quantite* prixproduit) where id =
new.id_panier;
end &&

delimiter ;
```

Description

Ce trigger mets à jour le panier d'un client après qu'il y ait ajouté un article.

TriggerDelPanier

```
delimiter &&

create or replace trigger delpanier after delete on panier for each row
begin
    delete from lignepanier where id_panier =old.id ;
    delete from panier_de where id_panier = old.id ;
end &&

delimiter ;
```

Description

Ce trigger supprime les produits des tables liés au panier après suppression dans ce dernier.

TriggerQteStock

```
drop trigger if exists qte_stock;

delimiter &&


create or replace trigger qte_stock after update on product
for each row
begin
    declare quantite_total int;

    select sum(quantite)
    into quantite_total
    from lignecommande where id_product = new.id;

    if quantite_total > new.stocktotal then
        signal sqlstate '45001' set message_text = "la quantite demandée est
supérieure au stock disponible";
    end if;
end &&

delimiter ;
```

Description

Ce trigger vérifie si la quantité disponible d'un produit est supérieure à la quantité demandé à chaque modification de stock de ce dernier.

Trigger_Del_product

```
drop trigger if exists del_prod;

delimiter &&

create or replace trigger del_prod before delete on product for each row
begin
    declare quant_comm int;
    declare quant_pan int;
    declare id_comm int;
    declare prix_prod int;
    declare id_pan int;

    delete from appartient where id_product = old.id;

    select quantite,numcommande into quant_comm,id_comm from lignecommande
where id_product = old.id;
    select prixunitaire into prix_prod from product where id = old.id;
    select quantite,id_panier into quant_pan,id_pan from lignepanier where
id_product = old.id;

    delete from lignepanier where id_product = old.id;
    delete from lignecommande where id_product = old.id;

    if id_comm is not null then
        update commande set prix = prix - (prix_prod*quant_comm) where id_comm
= numcommande;
    end if;

    if id_pan is not null then
        update panier set prix = prix - (prix_prod*quant_pan) where id_pan =
old.id;
    end if;
end &&

delimiter ;
```

Description

Ce trigger supprime les produits dans les tables liés à l a table produit si un produit est supprimé de cette table. Cela mets également à jour les commandes et paniers des clients.

Trigger_del_categorie

```
delimiter &&

create or replace trigger del_categorie after delete on categorie
for each row begin
    delete from appartient where id_categorie = old.id;
end &&
delimiter ;
```

Description

Ce trigger supprime les correspondances entre un produit et une catégorie quand elle est supprimé.

Trigger_del_client

```
delimiter &&

create or replace trigger del_client after delete on client
for each row
begin
    delete from panier_de where id_client =old.id;
end &&

delimiter ;
```

Description

Ce trigger supprime le panier d'un client quand il est supprimé.

Trigger_del_commande

```
delimiter &&

create or replace trigger del_commande after delete on commande for each row
begin
    delete from lignecommande where numcommande =old.id ;
    delete from effectue_par_client where id_commande =old.id;
end &&

delimiter ;
```

Description

Ce trigger supprime un produit de la ligne de commande d'un client quand il le supprime de sa commande.

Trigger_del_panier_de

```
drop trigger if exists del_panier_de;

delimiter &&

create or replace trigger del_panier_de after delete on panier_de for each row
begin
    delete from panier  where id =  old.id_panier;
end &&

delimiter;
```

Description

Ce trigger supprime le panier dans la table panier quand le panier correspondant dans la table panier_de est supprimé.